Artificial Intelligence Approaches to Music Composition

Richard Fox and Adil Khan
Department of Computer Science
Northern Kentucky University, Highland Heights, KY 41099

Abstract – *Artificial Intelligence (AI) techniques have been successfully applied to a wide range of problems that perform problem solving such as diagnosis, decision making and optimization problems. However, any AI algorithm applied to a creative problem requires some mechanism to substitute for the creative spark found in the human, as the computer has no creative capacity. Randomness, as supplied by a random number generator, cannot be the sole mechanism to bring about a creative composition. This paper examines three AI approaches applied to music composition. Specifically, the paper introduces MAGMA, a knowledge-based system that uses three different AI algorithms to generate music. Sample songs generated by MAGMA are compared.*

Keywords: music composition, Markov models, routine planning, genetic algorithms, artificial intelligence

1 Introduction

Artificial Intelligence (AI) research primarily produces problem solving systems that are aids in human decision making and identification. AI is often applied to problems such as diagnosis, design, and optimization. These problems often have specific solutions so that the AI system results can be tested against similar solutions as generated by human experts. AI research has also been applied to problems involving creativity such as composition of poetry or music. In such cases, output must be judged through subjective standards. However, when AI is used to solve creative problems, the AI algorithm must substitute something in place of the creative spark that humans seem to have. Randomness by itself will not be sufficient.

Similar to other areas of AI research, AI applied to music composition varies greatly by both the AI method(s) employed and the specific problem being solved. At one end of the music composition spectrum is the use of AI to generate a musical accompaniment in real time to musical performance [1, 2, 3]. At the other end are AI systems that compose music from scratch [4, 5, 6]. Also in the mix are AI systems which are trained based on previously composed pieces of music (e.g., classical music compositions of Bach and Beethoven), which then generate similar sounding compositions [7, 8]. AI-based music composition should not be confused with other forms of computer-generated music such as the use of MIDI technology or digital samplers and synthesizers.

Music composition in humans is performed in several different ways. Some people “hear” music in their mind and transcribe it onto sheet music. Others plan out the music through a deliberative process. Still others create music spontaneously through “jamming”. Or, musicians might combine these approaches. We might view these three different approaches as the application of music theory, the application of planning, and the application of randomness with pruning of music that “doesn’t work” as determined by some fitness evaluation. In AI, we might implement these three approaches using knowledge-based reasoning coupled with a stochastic approach, routine planning and genetic algorithms respectively. In the most recent AI research into music composition, the primary method utilized is the genetic algorithm (or some variation). Other approaches include neural networks, fractal geometry, and stochastic approaches using Markov chains.

There are many works that have applied genetic algorithms to music generation, including for instance GenDash [4] which models each measure of a song as a different population to evolve, or the work of Donnelly and Sheppard [5] which evolves four-part harmonies and their rhythm. CONGA [6], unlike the previous two mentioned systems, uses humans to perform the fitness analysis of each piece of music, thus drawing upon more than music theory to evaluate how listenable a piece of music is. AMUSE is used to generate and evolve improvised melodies given a harmonic context [9] and BlueJam employs a combination of heuristics and genetic algorithms [10].

There are numerous examples of stochastic approaches to music generation. For instance, the Stochos system uses eight different types of stochastic functions [11]. Bell applied Markov chains to control pitch, duration and chords of music that was generated by genetic algorithm [12].

The research presented here consists of three different AI techniques applied to the music composition problem. Specifically, this paper describes a system called MAGMA, the Multi-AlGorithmic Music Arranger, which utilizes three AI algorithms to generate music: stochastic generation via Markov chains, routine planning, and genetic algorithms.

The paper examines these AI-based algorithms in section 2. Next, the paper introduces MAGMA and describes how these algorithms are used to implement music creation. In section 4, two different sets of song excerpts are examined to compare the capabilities of the three different algorithms. Section 5 offers some conclusions and future work. The research reported here is a work in progress. The goal of this research is not to indicate that any single approach is better than another but to demonstrate how these approaches can be utilized to compose music.

2 Background

This section introduces the three algorithms that are used in the MAGMA system to compose music. These algorithms are described independently of music composition, saving those details for section 3.

A Markov chain is a state transition diagram whose links (edges) are annotated with probabilities [13]. The entire Markov chain represents a statistical model. The Markov chain can then be used to generate a possible sequence of events and the probability that the sequence will arise. Figure 1 illustrates a simple Markov chain of daily weather patterns. Given the Markov chain, one could generate the probability of a sequence of daily weather patterns. For instance, the probability given that today is sunny of the next three days being sunny, sunny, snowy, would be .4 * .4 * .2 whereas the probability of the next three days being rainy, rainy, sunny would be .4 * .6 * .3.

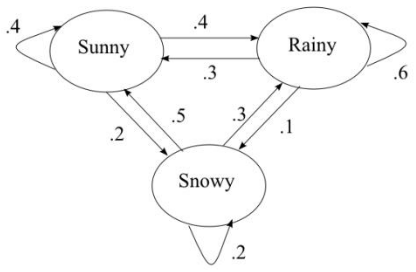


Figure 1: A Markov Chain

The statistical model can be easily generated from a database of events. For instance, the data from figure 1 could come from several months of weather data. Although the Markov chain is not specific to AI, it has found a number of uses within AI. Most commonly, a variation called a hidden Markov model (HMM) adds prior probabilities for each state and evidential probabilities of the likelihood that a given node is reached given the data. The HMM can be used to generate the most likely explanation for the appearance of some data. HMMs have found tremendous success in speech recognition while Markov chains have a common application to generating music.

Routine planning (or routine design) [14] captures the prototypical sequence of problem solving activities that a domain expert might undertake in planning or designing an artifact. It is a knowledge-based approach that describes the solution to a problem using the routine knowledge that the domain expert will compile through years of experience in solving variations of the problem. Routine design/planning has been used to solve a number of routine problems from air cylinder design to air force mission planning to nutritional meal planning.

The actual knowledge of the knowledge base breaks into several categories. First is a hierarchy of the components involved in the routine design or planning problem. This represents plan decomposition. The idea is that to construct the given artifact (whether a physical object or an abstract object) requires designing each of the components and subcomponents in turn. Second, to design any given (sub)component, the expert utilizes plan steps specific to that component. There may be many different plan steps for each component. The selection of the appropriate plan step is based on pattern-matching knowledge that identifies the plan step with the greatest chance of providing success in the designed artifact meeting user specifications, along with decisions already made on the design of other components. As component-level interactions may nullify a partial design, redesign steps are available, for instance decreasing a component's size because it is too large to fit within another component. Lastly, failure handling knowledge can be applied in the case that the designed artifact does not completely fulfill its intended function or meet all user specifications.

Figure 2 illustrates plan decomposition of routine design/planning along with plan steps available to design each (sub)component. In this figure, the overall artifact to be designed consists of three components, two of which have subcomponents.

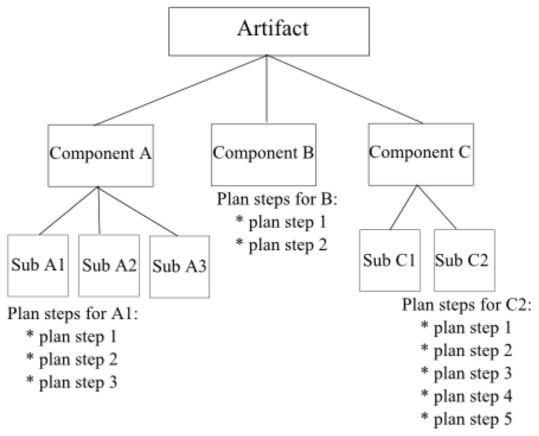


Figure 2: Routine Design Plan Decomposition

The genetic algorithm (GA) uses a form of natural selection to evolve a better solution [15]. The GA models the domain using a string of symbols, often called a chromosome, along with a fitness function. The GA starts with a base set of chromosomes. This population is treated as parents. The parents generate children through a series of genetic operations (mutation, inversion and crossover) that manipulate select parent chromosomes. In mutation, one or more values in a chromosome are randomly changed. For inversion, a sequence of chromosome values are reversed or otherwise rearranged. Crossover swaps portions of the chromosomes found in two parents so that two variations are generated as children. The result of these operations is a new population of children making up the next generation. These child chromosomes are evaluated using a fitness function. The fitness function models aspects of the domain so that the resulting value indicates how good the particular child is. A selection mechanism is then used to select the children who should survive into the next generation. Those children become the parents and the cycle continues. Selection algorithms include selecting the highest evaluated children, randomly selecting children, using rankings as probabilities for selection, and looking for diversity among the children being selected. The GA strategy is applied over and over for some number of iterations (generations) or until a child has been found whose fitness exceeds some desired threshold.

There are many open questions when solving problems with GAs. These include how one models the domain with a chromosome (which features are modeled? what values are permitted?), which genetic operations are applied and how often/to what extent, what fitness function should be used, what selection mechanism(s) is applied, how many children are generated, and how many children are selected to become parents. GAs have been applied to a wide variety of problems including optimization problems, planning/design problems, and a variations of GAs are often applied to automatic generation of program code. Section 3 elaborates on how these three algorithms can be applied to music composition.

3 MAGMA

MAGMA (Multi-AlGorithmic Music Arranger) is an experimental AI system currently under construction to compose music, specifically pop songs. The system uses a stochastic algorithm based on Markov chains, a routine planning algorithm, and a genetic algorithm to generate songs. The system takes user specifications as input and generates a song in the form of a MIDI file.

The user specifications determine what type of song the user is interested in. There are five preferences, each rated on a 5-point scale. These preferences are as follows.

- Transition determines the size of a transition from chord to chord or note to note. The transition value can be thought of as the maximum "step size" between any two chords or notes. A higher transition makes the song sound more "volatile" and might cause a song to have a more dissonant sound while a small transition might create a boring song.
- Repetition dictates how likely chords and notes might repeat before transitioning to other chords or notes. The lower the repetition, the more "creative" a song might sound. Repetition is also applied to song structure so that a lower repetition would lead to song components that are not repeated as often.
- Variety applies to chords, notes and song structure, impacting how many chords/notes/song components are generated. A higher variety leads to a song that has more parts and more diversity within those parts. The lower the variety, the simpler the song.
- Range applies to the chords and notes, similar to repetition and variety, but in this case it influences the chords and notes over the entire song. Range also controls the number of octaves that might be applied as well as the instruments selected for the MIDI file.
- Mood impacts the key and the tempo of the song. A more somber mood may cause a minor key and a slower tempo. A more upbeat mood would often result in a major key and a faster tempo. An intermediate mood may cause minor and major keys for different song components such as a minor key for the verse and a major key for the chorus. The mood also impacts the instruments selected.

The user also specifies which of the three algorithms to utilize. MAGMA will generate a single song using the selected algorithm (stochastic approach, planning approach, genetic algorithm approach). The output of MAGMA is a MIDI file of a song that might vary in duration from 2 minutes to 5 minutes. See figure 3 for an overview of the system's architecture.

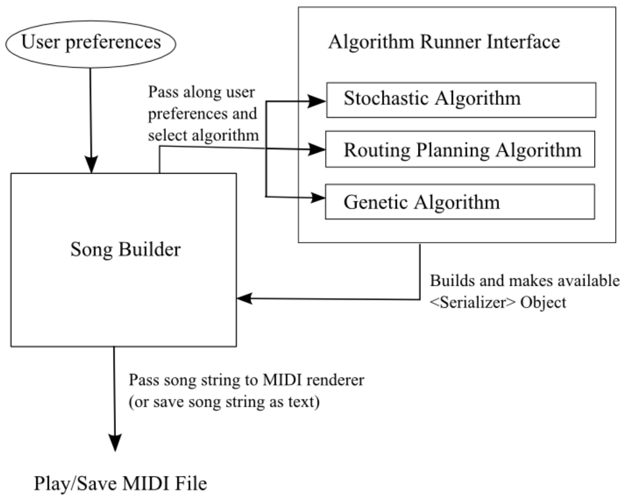

User preferences
Algorithm Runner Interface
Pass along user preferences and select algorithm
Stochastic Algorithm
Routing Planning Algorithm
Genetic Algorithm
Song Builder
Builds and makes available <Serializer> Object
Pass song string to MIDI renderer (or save song string as text)
Play/Save MIDI File

Figure 3: MAGMA's Architecture

Each of the three algorithms generates a song through planning decomposition, as shown in figure 4. First, the song's structure is generated. Song structures are based on a few components: introductions (I), verses (V), choruses (C), bridges (B), solo sections (S) and outros (O). A simple song may have a structure of I-V-C-V-C-O and a more complex song might have a more elaborate structure of I-V-V-C-V-B-C-S-C-O. In some cases, an intro and/or outro may match a verse and in other cases, the intro, outro and verse may all differ. Additionally, a song may contain a modulation either between or within components, for instance by having two repeated choruses shift from the key of D to the key of E.

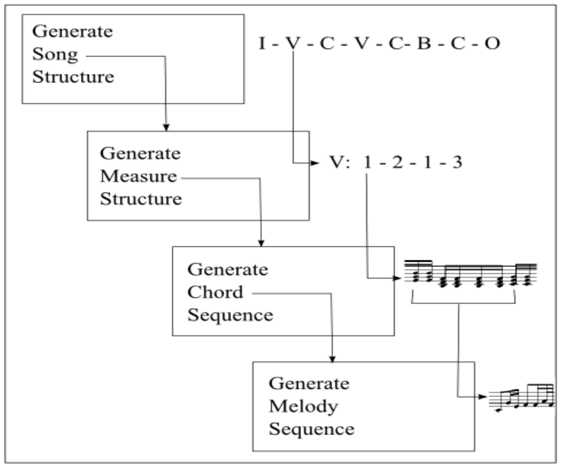


Figure 4: Four Steps to Generate a Song

Given the song structure, or song components, the next step is to generate the structure of each of those components. This structure is based on the number of and types of measures. For instance, a verse might consist of 4 or 8 measures. These measures may all be identical, or they may alternate between different chord sequences. A 4-measure component may follow a pattern like 1-1-1-2 or 1-2-1-3 or 1-1-1-1 or even 1-2-3-4. Obviously, repetition and variety impact the generation of the component's structure.

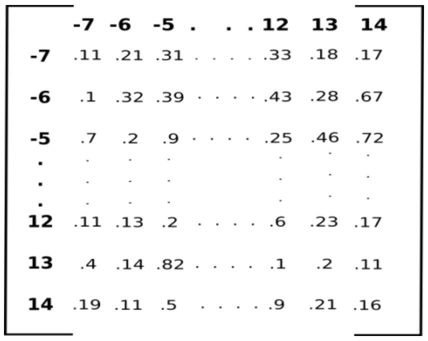
Now MAGMA must generate the actual chords for each of the measures. If a verse has a pattern of 1-1-1-2, then MAGMA must generate two chord sequences for the verse. The measure will comprise some number of "beats". In 4/4 timing for instance, a measure might consist of 4 quarter notes or 8 eighth notes or 1 whole note. The variety will help dictate whether all chords have the same duration or whether some chords will be longer and/or shorter than others. A more diverse measure might consist of a half note followed by a quarter note followed by two eighth notes.

Given the chords, MAGMA now generates a melody sequence over those chords. The melody is generated independently of the chord sequence but must match the chord sequence's duration. The notes generated are based on the key that makes up the component (e.g., if the verse is in the key of Am, the notes generated must match Am). In this way, the notes are not necessarily based on the chord that the notes are played over.

All chord and melody data are generated using Nashville notation so that they are key agnostic (using numbers in place of notes). This permits easy translation of a note or chord sequence based on the generated key. The key is generated randomly, but is impacted by the user's preference for mood. Selections for tempo and musical instruments (for the MIDI file) also combine mood and randomness.

Each of the three algorithms (stochastic, planning, genetic algorithm) performs these four plan decomposition steps in different ways. However, all three of the algorithms perform these steps in the order given in figure 4.

For the stochastic algorithm, each of the four steps is handled through separate Markov chains, each of which was automatically generated using a simple parsing algorithm from a collection of pop songs. For instance, the chord transition probabilities were generated by parsing the chords of numerous songs. The parser provides the Markov chains as transition probabilities matrixes, as shown in figure 5. MAGMA utilizes one matrix for song structure, one for measure structure, one for chord generation and one for melody generation. There are additional matrices for phrase transition (e.g., chorus to verse chord transitions), and chord duration. For each generated song, the probabilities of the matrices are altered based on the user specifications to better match the user's interests.



	-7	-6	-5	. . .	12	13	14
-7	.11	.21	.31	. . . .	.33	.18	.17
-6	.1	.32	.39	. . . .	.43	.28	.67
-5	.7	.2	.9	. . . .	.25	.46	.72
.	.	.	.		.	.	.
.	.	.	.		.	.	.
.	.	.	.		.	.	.
12	.11	.13	.2	. . . .	.6	.23	.17
13	.4	.14	.82	. . . .	.1	.2	.11
14	.19	.11	.5	. . . .	.9	.21	.16

Figure 5: Partial Transition Matrix for Notes/Chords

The planning approach has numerous plan steps that define specific sequences of song components (the song structure), measures, chord sequences and note sequences. The song structure is generated by selecting the sequence that most closely matches the user specification. Next, for each song component, its structure is generated in terms of the pattern of measures. Again, the planning system has a number of enumerated lists such as 1-2-1-2 or 1-1-2-1-1-2-1-3, and sequence selected is the one that most closely matches the user's specifications. Given a single measure, now the planning approach selects a chord sequence that again most closely matches the user specifications. Similarly, notes are then generated.

Below are three examples of song structure and the user preferences that each would best match. For instance, the first component would proscribe a song that consists of the pattern Introduction, Verse, Chorus, Verse, Chorus and Outro. It has a repetition value of 2 and a variety value of 3. The third example has a repetition value of 1 and a variety value of 5.

- `{R=2,V=3}=I|V|C|V|C|O`
- `{R=2,V=4}=I|V|C|V|C|B|C|O`
- `{R=1,V=5}=I|V|C|B|O`

Chord sequence and melody sequence plans include the pitch (the chord or the note) and the rhythm. What follows are examples of chord sequences. The numbers indicate the chord to play where 1 equals the key (e.g., a key of D would mean that a 2 is an E chord and 5 is an A chord). A 0 indicates a rest, or no chord/note played. The letter following the number represents the duration of the chord or note with w meaning a whole note, h being a half note, q being a quarter note, i being an eighth note, s being a sixteenth note, etc. The two lists below are examples of chord sequences and note/melody sequences respectively.

- `{R=1,T=1,V=2,H=3}=4q|4q|5q|5q`
- `{R=1,T=1,V=2,H=2}=5h|6h`
- `{R=1,T=1,V=1,H=1}=5w`

- `{R=2,V=2,T=2,H=4}=0q|0i|3i|3i|5i|5q`
- `{R=1,V=2,T=1,H=2}=2qqq|-3q`
- `{R=3,V=3,T=1,H=2}=3q|3q|2q|1q`
- `{R=4,V=1,T=1,H=3}=-1q|1qi|0i|-1q`

To further illustrate, a song with the key of A might generate a measure whose chord sequence is A-A-A-G-G-G-G-E and consist solely of eighth notes expressed as 1e|1e|1e|7e|7e|7e|7e|5e. Another example might be a song with a key of Cmaj with a measure whose chord sequence is D (half note), E (3 quarter notes) F (half note). In the latter case, this would be expressed as 2w|3qqq|4h.

In order to ensure that the melody generated matches to some extent the chords generated, three additional mechanisms are applied. First, the duration of a measure, as generated during the chord sequence phase, is used to "chop off" a melody. That is, if a selected melody plan step is too lengthy, its duration is simply truncated upon reaching the end of the chord sequence. Second, the melody must match the key of the song component (e.g., the key of the verse). This may or may not precisely match the chords and so additional work may be required to perform "failure handling" to ensure the notes and chords work together with respect to music theory. Finally, transition rules are applied to alter the sequence of chords as they move from one song component to another (e.g., verse to chorus transition).

The Genetic Algorithm approach employs four types of chromosomes, one each for the song structure, component structure, chord sequences and note sequences. Five fitness functions are utilized, one per type of user preference. The fitness of a chromosome is based on how closely it matches the given user preference. The five fitness function values are combined using a weighted average.

The song structure is generated with an initial chromosome size based on the user preference. If the preference is for low variety and low repetition for example, the size will be smaller than a preference of high repetition and high variety.

For each song component (verse, chorus, etc.) a measure sequence is evolved. The chromosome's size will depend on user preferences. A longer chromosome will be generated for high repetition and variety, while a shorter chromosome for low repetition and variety. At this level, each member of the chromosome represents a measure. So if the final chromosome is evolved to 1,1,2,1,3,2, then three distinct measures make up that song component.

Given a single measure, the next step is to generate its chord sequence. The chord sequence is influenced by user preference of repetition, variety and transition. Repetition and variety will also influence the size of the chromosome (chord sequence).

Once the chord sequence has been generated, a melody is evolved for each chord in that sequence. The chromosome comprising the melody is influenced by repetition, variety and transition. If the duration of the melody is larger than the underlying chord then it is trimmed to match the chord duration. If it is shorter, then the last note is padded to match the duration of the chord. The process of chord and then melody generation is repeated for every measure of the song.

During each phase of the genetic algorithm, every chromosome is scored by the fitness function, generating a real number between 0 and 1. This value is the combined weighting of the fitness functions that encode the user specifications applied for that portion of the song composition. For instance, music theory, diversity, repetition and transition are all applied for chord and note sequences while range, mood, diversity, transition and repetition are used to generate the song's structure.

A chromosome for a chord or melody sequence is denoted using Nashville notation which facilitates easy computation of transitions, distinct notes, and so forth. As an example, a chromosome storing the sequence 1, -3, 1, 5, 7, 9, 12 has almost no repetition but contains a high degree of transition. Negative numbers in the sequence represent lower octaves while numbers greater than 7 represent higher octaves. The use of the Nashville notation also simplifies music theory computations irrelevant of the key.

For each portion of song generation by the genetic algorithm (refer back to figure 4), the initial population size is 10 chromosomes. The initial population is created randomly. For each generation, four parents are selected from the base population, two of which have the highest fitness and two of which are the most diverse from the two parents. The four parents produce six children by crossover, and four additional children through mutation. For song structure and song component generation, 10-15 iterations are performed. For chord and melody sequence generation, 100 iterations are performed. The rationale being that structure is more easily evolved than the chord and note sequences.

4 Examples

This section examines and compares two sets of songs generated by MAGMA. For the first set of songs, the user specified high transition, high variety, highly rhythmic, low repetition and high range. Excerpts from the stochastic, planning and genetic algorithm portions of MAGMA are shown in figures 6, 7 and 8 respectively. The excerpts shown in the figures are typical of the whole song from which they were excerpted.

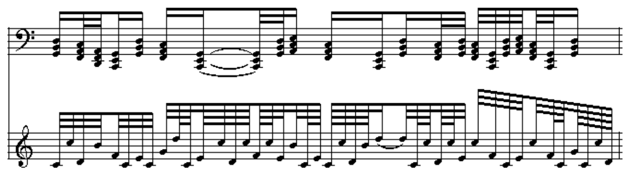


Figure 6: Excerpt from Stochastic Approach

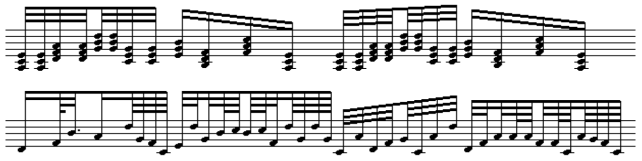


Figure 7: Excerpt from Planning Approach

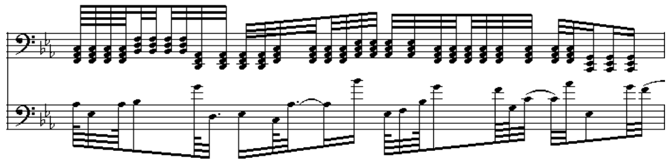


Figure 8: Excerpt from Genetic Algorithm Approach

Notice in figure 6 that the melody of the stochastic approach consists of rapidly changing notes where the transition of change (step size) between notes is being quite large. This leads to a chaotic, uneven sound. The melody generated by the genetic algorithm (figure 8) is sparser than the other two songs leading perhaps to a more listenable melody. However, the chord sequences from the genetic algorithm have a rhythmic pattern which makes the song less listenable (see how the chords are played continuously in the measure). The song generated from the planning approach (figure 7) offers a compromise between these two extremes. The melody is neither chaotic nor with large step sizes, and the chord structure, while not as diverse as the genetic algorithm approach also does not suffer from a lack of rhythm. In the opinion of the authors, the song generated from the planning algorithm is far more listenable.

The second set of songs was generated using specifications of low transition, low variety, less rhythmic, high repetition and low range. The low range had a large impact on all three generated songs in that they are at a much slower tempo. In the case of the genetic algorithm output, the song was placed in a minor key giving it a somber sound. Excerpts from the three songs as generated by the stochastic approach, planning approach and genetic algorithm approach are shown in figures 9, 10 and 11 respectively.

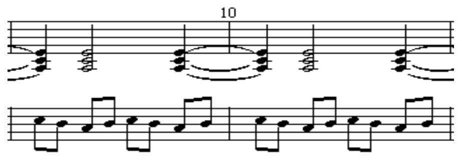


Figure 9: Excerpt from Stochastic Approach

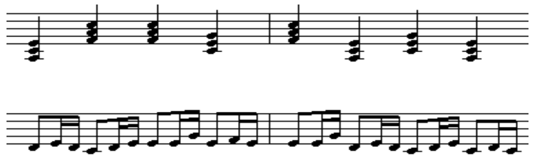


Figure 10: Excerpt from Planning Approach

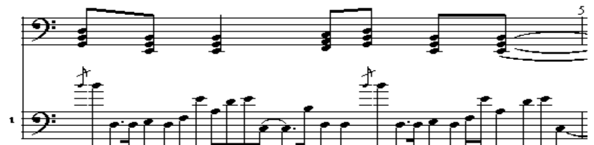


Figure 11: Excerpt from Genetic Algorithm Approach

The song generated by the stochastic approach is quite simple to the point of being boring. As can be seen in figure 9, the melody is highly repetitive. In this excerpt, the chords remain the same throughout this passage of music. The song generated from the planning approach is slightly more interesting while remaining basic. Only the genetic algorithm produced a piece of music that has some variability. And yet, the large step sizes between notes cause this song to lack coherence.

This brief look at the results of the three algorithms cannot lead to any definitive conclusions. However, it should be apparent that the stochastic algorithm is at a disadvantage because it does not apply any explicit strategy to either follow music theory or compositional strategies that make a song listenable. The main detractor of the songs generated by the genetic algorithm is the overly random nature of the notes. The planning approach does not suffer from either of these problems but may lack in originality because it is impacted the least by randomness.

5 Conclusions

MAGMA, the Multi-AlGorithmic Music Arranger, applies three artificial intelligence algorithms to compose pop songs: a stochastic approach using Markov chains, a routine planning approach and a genetic algorithm approach. MAGMA receives user specifications and applies the algorithms to generate songs. Song generation starts with the development of the song's structure. This structure reflects the component parts that will make up the song, such as verses and choruses. Each component is itself decomposed into measures. A component might be made up of several distinct measures such as a pattern of 1-2-1-2. Each measure is then designed by generating both chord sequences and notes (a melody). The generation of song structure, measures, chords and notes is all handled by one of the AI algorithms.

MAGMA is a proof-of-concept system being constructed as part of a master's thesis in computer science. The goal is to demonstrate how these approaches can be applied for music composition, and to compare and contrast the music generated from these algorithms. MAGMA generates songs as MIDI files so that people can both listen to the music and view the music in staff notation. While the analysis provided in this paper can point out flaws with generated music, a survey of listeners can provide a subjective comparison as well. This is intended as future work.

While the point of this paper is not to compare the three approaches, the examples shown do demonstrate a drawback of a strictly stochastic approach to music composition in that there is little to no ability to apply music theory or principles that lead to listenable music. The genetic algorithm approach has an advantage over the planning approach in that it can provide music with a large variability and randomness. The planning approach is perhaps a good middle ground. Ultimately, it is hoped that MAGMA can combine the strengths of the three algorithms to produce the most listenable music.

References

[1] Simon, I., Morris, D., and Basu, S (2008). "MySong: automatic accompaniment generation for vocal melodies." Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems. p. 727-734, ACM.

[2] Assayag, G., Bloch, G, Chemillier, M., Cont, A., and Dubnov, S. (2006). "Omax brothers: a dynamic yopology of agents for improvization learning." Proceedings of the 1st ACM workshop on Audio and music computing multimedia, p. 125-132, ACM.

[3] Blackwell, T (2007). "Swarming and music." Evolutionary Computer Music, p. 194-217, Springer.

[4] Waschka, R (2007). "Composing with Genetic Algorithms: GenDash." Evolutionary Computer Music, p. 117-136, Springer.

[5] Donnelly, P., and Sheppard, J (2011). "Evolving four-part harmony using genetic algorithms." Applications of Evolutionary Computation, p. 273-282, Springer.

[6] Tokui, Nao, and Hitoshi Iba. "Music composition with interactive evolutionary computation." Proceedings of the 3rd International Conference on Generative Art. Vol. 17. No. 2. 2000, Generative Design Lab.

[7] P. Hamel and D. Eck. Learning features from music audio with deep belief networks. In 11th International Society for Music Information Retrieval Conference (ISMIR 2010), 2010.

[8] Zhang, Q., and Miranda, E (2006). "Evolving musical performance profiles using genetic algorithms with structural fitness." Proceedings of the 8th annual conference on Genetic and evolutionary computation, p. 1833-1840, ACM.

[9] Özcan, E., and Erçal, T (2008). "A genetic algorithm for generating improvised music." Artificial Evolution, p. 266-277, Springer.

[10] Rowe, C (2008). "BlueJam: A heuristic-based approach to evolutionary music generation." Technical report, parallaxed.net/bluejam.

[11] Bokesoy, S., and Pape G (2003). "Stochos: Software for Real-Time Synthesis of Stochastic Music." Computer Music Journal 27.3, p. 33-43, MIT Press.

[12] Bell, C (2011). "Algorithmic music composition using dynamic Markov chains and genetic algorithms." Journal of Computing Sciences in Colleges 27.2, 99-107, CCSC.

[13] Hazewinkel, M. ed. (2001), "Markov chain", Encyclopedia of Mathematics, Springer.

[14] Brinkop, A., Laudwein, N., and Maasen R (1995). "Routine Design for Mechanical Engineering." AI Magazine, p 74-85, AAAI.

[15] Holland, J (1992). Adaptation in Natural and Artificial Systems. Cambridge, MIT Press.