AutoChorusCreator : Four-Part Chorus Generator with Musical Feature Control, Using Search Spaces Constructed from Rules of Music Theory

Benjamin Evans[†1] **Satoru Fukayama**[‡2] **Masataka Goto**[‡3] **Nagisa Munekata**[†4] **Tetsuo Ono**[†5]

[†] Hokkaido University, Japan

[‡] National Institute of Advanced Industrial Science and Technology (AIST), Japan

[1]benjamin[at]complex.ist.hokudai.ac.jp [2]s.fukayama[at]aist.go.jp [3]m.goto[at]aist.go.jp [4]munekata[at]complex.ist.hokudai.ac.jp [5]tono[at]ist.hokudai.ac.jp

ABSTRACT

This paper describes AutoChorusCreator(ACC), a system capable of producing, in real-time, a variety of four-part harmonies from lead sheet music. Current algorithms for generating four-part harmony have established a high standard in producing results following rules of harmony theories. However, it is still a challenging task to increase variation in the output. Detailed constraints for describing musical variation tend to complicate the rules and methods used to search for a solution. Reducing constraints to gain degrees of freedom in variation often lead to generating outputs which do not follow the rules of harmony theories. Our system ACC is based on a novel approach of generating four-part harmony with variations by incorporating two algorithms, statistical rule application and dynamic programming. This dual implementation enables the system to gain the positive aspects of both algorithms. Evaluations indicate that ACC is capable of generating four-part harmony arrangements of lead-music in real-time. We also confirmed that ACC achieved generating outputs with variations without neglecting to fulfil rules of harmony theories.

1. INTRODUCTION

Automatic composition has captivated the minds of both musicians and scientists for decades and many approaches have already been attempted in the field of information science [1, 2, 3]. Some of these include constraint satisfaction [4, 5, 6], example based approaches [7], genetic algorithms [8, 9], probabilistic modelling [10, 11] and rule based applications [12]. Recently, technologies originally from the field of music information retrieval (MIR) are also being used to support people who create musical works [13, 14].

Harmony is an important element in many music styles, especially in those of classical music. Emura describes an academic process of musical composition as 1) choosing a simple cadence of chords, 2) deciding a melody line which follows the structure of the sequence of chords, 3) replacing some of those chords with other replaceable chords and 4) adding harmonic interest to the piece by adding non-harmonic tones [15]. Harmonisation and re-harmonisation are important aspects in each of these steps. This fact makes the appropriate implementation of harmony theories crucial in developing automatic composition systems that incorporate aspects of tonality as those found in classical music.

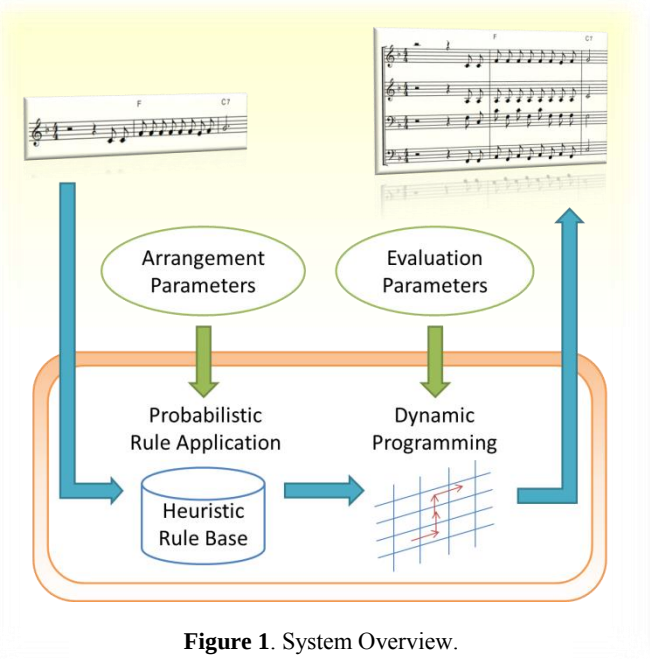

Figure 1. System Overview.

A particular task often dealt with in the study of automatic harmonisation is that of harmonising a classical four-part chorale from a single melody line. Allan used a data set of chorale harmonisations to train Hidden Markov Models to create four-part harmony [16]. Suzuki also used probabilistic models for automatic four-part harmonisation [17], comparing system outputs for when chord data was and was not used.

Rule-based approaches have also been exploited for four-part harmonisation. Ebcioğlu developed a rule-based system with over 270 rules and used a logic programming language for harmonising four-part chorales [12]. Phon-Amnuaisuk also created a rule-based system and compared it with a genetic algorithm system which had the same explicit rules of harmonisation implemented in it [8]. Biles used genetic algorithms in creating jazz improvisations [9]. MacCallum also used genetic algorithms

and incorporated mass consumer evaluations to investigate the composition of music through Darwinian processes [18].

While many four-part harmonisation systems acquire a high level of musical quality, it is still a challenging task to produce a variety of outputs from a single input. Harmonisation systems produce chorales that fulfil many rules of music theory. Statistical models have been reported to learn different rules of the theory of harmony as implicit representations in the system itself, while rule-base systems are successfully producing complicated arrangements that follow specific styles. However, the variation in output of these systems is often limited, leaving us an area within the field of automatic harmonisation with potential still to be exploited.

When aiming to gain diversity in harmonisation, we find neither increasing nor decreasing rules is the optimal solution. If a system extends its set of rules to incorporate rules that describe new styles of music, rule formats and search methods can become complicated. This will result in the system becoming less efficient than it originally was, often extending the execution time also. Reducing rules to gain extra degrees of freedom will result in a reduction of musical quality in the output when held up to scrutiny against theories of harmony. Both extending and reducing rule sets are still a challenging solution to adding diversity to automatic harmonisation. A new mechanism is needed if the system is to produce diverse harmonisations effectively.

The rest of this paper explains how we achieved that diversity through the combined implementation of a rule-based application and dynamic programming. In section 2, we describe some characteristics of a practical automatic harmonisation system and the technology requirements to produce its characteristics. We also explain the details of the system we implemented, listing some examples of the rules used in each algorithm. In section 3, we note the evaluations we made of the system before concluding in section 4 with a summary of this work and a discussion on areas that need future development.

2. METHOD

2.1 System Concept

A robust harmonisation system with practical applications would have to include the following characteristics.

1. Is able to produce a variety of outputs, preferably according to user-specified styles or characteristics.

2. Is able to be executed in real-time.

3. Is able to produce "good quality" harmony – harmony that follows basic principles of style/structure or that can be musically understood uniformly by a general audience.

These are some fundamental aspects a system must simultaneously achieve if it is going to be used in practical applications of musical composition. While it is relatively easy to achieve one or two of these characteristics in a harmonisation system, we have found the simultaneous attainment of all three to be more difficult. One reason for this would be that a typical harmonisation system will usually apply only a single algorithm (e.g. rule application, algorithmic search, pattern matching) to complete a given task. Implementing a single algorithm leads to a system which is an expert for one task, but less effective for another.

We have developed "AutoChorusCreator (ACC)," a harmonisation system that incorporates multiple algorithms into the harmonisation process. By implementing multiple algorithms, ACC benefits from the positive aspects of each algorithm while covering their weaknesses with another. ACC simultaneously achieves all three requirements listed above, thus providing a robust yet flexible harmoniser that can be used in practical contexts of music composition.

2.2 System Overview

ACC is an automatic four-part choral music harmoniser. Users input data of lead music (sheet music with one melody line and chord notations) from which the system creates a score designed for a four-part voice ensemble. ACC is implemented in Java and uses MusicXML for input and output data format.

ACC consists of two modules, each using different algorithms. The first module creates an initial arrangement from the input data using statistical applications of a heuristic rule set. This initial arrangement is then passed on to the second module, which uses dynamic programming to search for the optimal output out of those similar to the initial arrangement. Each rule in the heuristic rule set in the first module is applied to the input music according to a probabilistic model with parameters alterable by the user.

The evaluation functions in the second module mostly consist of rules derived from classical theories of harmony, but also include functions to evaluate other features of music that do not relate to harmony. Each function is weighted by a parameter and is applied to a piece of music to produce a cumulated evaluation score which is then used in the search process. By altering the parameters in each module, ACC is able of producing multiple arrangements of chorale music from a given input data. We have listed a diagram of the system overview in Figure 1.

Below we will describe in detail how we have integrated these two algorithms to produce a composition system that meets the three criteria we discussed in the section above.

2.2.1 Output Diversity

Users have diverse preferences regarding music even within the same music genre, and so a practical harmonisation system must also incorporate diversity into its system output. Harmonic variety not only makes a system interesting, but also creates a system potentially acceptable by a wider range of users.

Harmonisation systems often focus on finding optimal chord progressions that match a given melody, but spend little effort finding the best combination of notes within each chord. Although the decision of chord progressions is an important procedure, it is only a small part of the harmonisation process and a limited aspect of output diversity. Chord progressions determine the space from which the note in each part must be chosen for, but it is other musical qualities (e.g. movement in a single part, reflection of musical intensity in the melody to other parts, technical interest matching each performer's physical ability) that determine the final output. These musical qualities we will call "musical features."

The two algorithms implemented in ACC assist each other in achieving diverse musical features in output arrangements. As will be discussed in section 2.2.3, dynamic programming was incorporated as the second module to assist in finding an optimal output within a reasonable amount of time. Although dynamic programming is an efficient search algorithm, it will only find one output from a given input. To create various outputs from even the same input, ACC first creates an initial arrangement using statistical rule application and then passes this on as the seed for the search algorithm in the dynamic programming phase. Each heuristic rule is applied to portions of the song at random, producing a different initial arrangement for the dynamic programming module to work on each time the programme is executed.

We compiled a set of heuristic rules from observations of pre-composed pieces of music to use in the first algorithm. The compiled set of ten rules was far from being a complete implementation of musical arrangement procedures. We did find, however, that this limited set of rules was still adequate in examining whether the two-algorithm implementation of the harmonisation process could produce various arrangements from the same piece of music.

The rules were implemented so that they conducted simple structural alterations in each accompaniment part. ACC first chooses for each part a note contained in the chord listed in each measure. If there are no chord notations in a given measure, then the previous chord notation is used. Once the first note for each measure has been chosen, ACC applies rules such as those listed below to produce more notes to fill each measure.

- If two consecutive notes in the same part have the same pitch, move the later note to another pitch within the same chord (Figure 2.)

- If the distance between two consecutive notes in a part is a third, then halve the length of the first note and add a second note of the same length. The second, new note is raised to the next note in the key creating a "stepping" effect (Figure 3.)

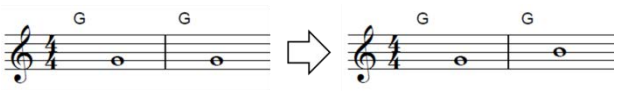

Figure 2. Example of Heuristic Rule (1).

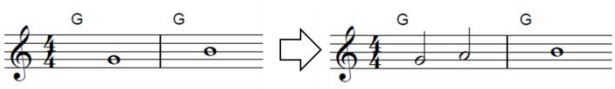

Figure 3. Example of Heuristic Rule (2).

Each rule *n* is only applied to an applicable passage of music at the probability p_n ($0 \leq p_n \leq 1$). This allows for a diverse range of initial arrangements which are then passed on to the dynamic programming module. Notice these rules are designed to produce structural diversity only and do not directly represent any particular music theory.

2.2.2 Search Area Reduction

The computational time of harmonisation systems increase exponentially with the length of the music given as input. This makes search area reductions essential in keeping system execution time minimal. Search area reductions can be implemented as computational techniques borrowed from studies of search algorithms, or as active search area reductions in the algorithm itself using knowledge of the music being composed. This musical knowledge can be constructed manually from music theories or learned automatically from corpuses of pre-existing compositions through machine learning.

The second module of ACC consists of a dynamic programming algorithm. Dynamic programming is a computational algorithm which can be applied to problems where the main problem is subdivided into multiple sub-problems, and the accumulation of optimal solutions from each sub-problem composes the optimal solution for the main problem [19]. Dynamic programming has been used in other music composition systems also, such as Fukayam's system *Orpheus* [20]. In our system, we have taken the task of harmonising the whole song and broken it down into a group of the subtask "finding the optimal combination of notes played simultaneously across four parts at each point in the song." By finding the optimal combination of notes at each moment of the song, our system will find a desirable solution to the harmonisation task.

To evaluate each set of four notes, we have chosen to include the theory of harmony of classical music as will be discussed in section 2.2.3. The classical theory of harmony does not only evaluate each group of notes on their own, though. Harmony theories classify groups of pitches into "chords." Evaluation is conducted on the structure of each individual chord and also its relation to the chords surrounding it. In the current version of ACC, we have implemented the rules which look at the structure of each chord and the relation to its preceding chord only.

Rule A 1 : The only note you may exclude from a chord is the 5th note of the chord. You may also exclude the tonic note if the chord is a dominant fifth/ninth chord.

$$f_1 = \begin{cases} \dfrac{n_{abbr.}}{n_{chord} - 2} : if(chord = 7th\ chord\ or\ 9th\ chord) \\ \dfrac{n_{abbr.}}{n_{chord} - 1} : else \end{cases}$$

$n_{abbr.}$: Number of abbreviated notes in the chord that must not be abbreviated

n_{chord} : Total number of notes that comprise the chord

p_{9th} : The pitch number of the 9th note, in MIDI notation

p_{chord} : The pitch number of the tonic note, in MIDI notation

Rule A 3 : The 9th note of a dominant ninth chord must be played at least an octave and a second (14 semitones) above the tonic note.

$$f_3 = \frac{|min(p_{9th} - p_{tonic} - 14, 0)|}{(p_{max,p_{tonic}} + 14) - p_{min,p_{9th}}}$$

$p_{max,p_{tonic}}$: The highest possible pitch of the part with the tonic note, in MIDI notation

$p_{min,p_{9th}}$: The lowest possible pitch of the part with the 9th note, in MIDI notation

Figure 4. Examples of continuous evaluation functions based on classical rules of harmony from [21], used in the dynamic programming module.

The dynamic programming module takes the initial arrangement composed in the rule-based module and conducts a search by altering each note by d semitones in either direction. The distance d therefore is the key factor in determining the extent to which dynamic programming conducts its search. The number of possible combinations of notes for an entire song can be annotated as

$$(\mathbf{2}\boldsymbol{d} + \mathbf{1})^{\mathbf{3}\times\boldsymbol{n}} \qquad \mathbf{(1)}$$

where d denotes the number of semitones a note can be moved in one direction and n is the number of chords (combination of four notes) to be analysed in a song.

As can be seen from equation 1, the computational time will increase exponentially as the length of the song increases. This is why we have limited the search to those songs similar to the initial song composed by the first module.

2.2.3 Evaluation Using Music Theory and Musical Features

In one way or another, every harmonisation system includes an evaluation module. This could be implemented in the search process so that the system makes an implicit evaluation at every decision. It could also be implemented at the end of the composition cycle as an explicit evaluation function. The former would evaluate local combinations of note while the latter could take into account the global structure of the entire musical piece. The evaluation process could also be excluded from the programmed system and given to the user as a manual operation. This evaluation would reflect the user impressions of the composed harmony. In general, the evaluation, and therefore the definition of "good harmony," will contain three aspects; local structure, global structure and user impression. We have added the evaluation procedure into our system as weighted functions used in the search algorithm of the second module.

Having formulated theories of musical structure is a particular characteristic of classical and popular Western music. While many theories for each of the individual elements of these music, such as melody, harmony and rhythm, have been formulated over time, the theory of harmony is the most refined and formalised of them all.

A difficulty found with applying rules of harmonisation in a computer programme is that some combinations of notes can only be attained if there is a correct sequence of notes leading up to, and after, that point in the music. By unintentionally choosing one different note combination somewhere in the song, a system could limit itself from ever reaching a particular combination of notes elsewhere in the music. One must keep in mind search algorithms need flexibility to search multiple combinations of notes throughout the whole song to find various output note sequences.

We have implemented both functions made from rules of harmony and functions that describe musical structure (musical features) in the search algorithm of the second module. The list of functions regarding the theory of harmony was compiled from *Geidai Wasei* [21], the text book on the theory of harmony of classical music most used by educational institutes in Japan [22]. *Geidai Wasei*, like other formalised theories of harmony, consists of a set of rules prohibiting specific combinations of pitches (e.g. seventh notes of a chord must not be doubled in a chord) and chord progressions (e.g. no parallel fifths). The easiest implementation of this type of rule is in the form of a Boolean function which determines whether the rule is broken at a certain point in the music or not. This, however, does not allow for discrimination between two songs which both break a certain rule but in different ways (e.g. two songs may have a doubled seventh note, but in fact one song may have the seventh note tripled, not just doubled). Therefore, when possible, we have implemented the rules with a continuous evaluation function to give a more discriminative preference ordering to any two pieces of music. See Figure 4 for examples of these functions. A total of 13 rules (A1-A5, B1-B3 and C1-C5) from *Geidai Wasei* were implemented as evaluation functions in the second algorithm of ACC.

Along with rules of formal musical theory, we have also incorporated rules regarding other musical features, such as the structure of each part and the number of notes in each bar of music. Many rules from the theory of har-

mony exclude specific combination of notes from the final output. However, there are very few rules which recommend a preferable combination of notes. We incorporated functions that evaluate musical features, not just harmony, so that we could manipulate the final output through explicit knowledge, rather than allowing the system to make random decisions from arrangements with similar harmonic evaluations.

Musical features evaluated in the dynamic programming algorithm included those such as "amount of pitch movement in each part" and "vocal range". Users input their preferred amount of each musical feature, such as "minimal pitch movement" or "vocal range no larger than one and a half octaves, starting from middle C." Each function then evaluates how different a musical arrangement differs from the user's preference, and gives a corresponding score. The evaluation result of each function is weighted and added to provide one final score which represents that arrangement. That score is then used in searching for an optimal arrangement as the final output. Altering the weights on each function affects the amount of influence each represented musical feature has on the search process. Adjusting these weights gives the user some control over the ultimate output produced from a given input.

3. RESULT

3.1 System Performance

In this section, we describe the performance of our system when using the two system modules individually and combined. We have used a popular children's song "If You're Happy and You Know It" as the example input, as shown in each of the figures below.

In Figure 5, we show an example output when using dynamic programming only. Each part tends to move in a monotonous way, as can be seen in the up-down movements of the tenor and bass parts. This is because when using dynamic programming only, each part starts with the same note right through the measure. The same rules of harmony are applied to every set of chord, forming similar movements through each measure.

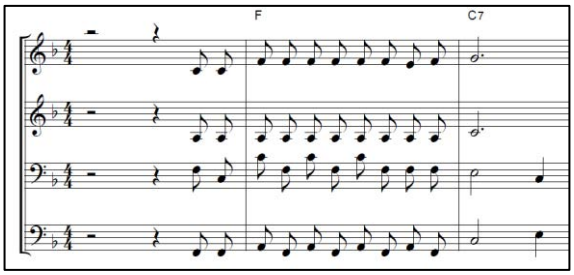

Figure 5. System output made using the dynamic programming module only.

Figure 6 shows an arrangement composed using the rule-base module only. Music made using heuristic rules only are prone to go against basic rules of harmony (e.g. parallel movements between parts) as they are not implemented explicitly in the rule base. Only using this module limits the diversity of output arrangements as they are solely dependant on the set of rules provided in the system. The amount of explicit knowledge represented in the rule set alone determines the variety in the output.

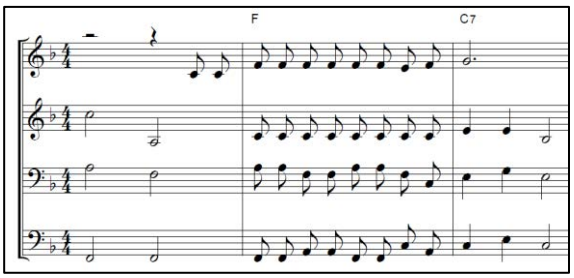

Figure 6. System output made using the rule-base application module only.

As seen in Figure 7, a combined application of both the heuristic rule-base module and dynamic programming module produces an arrangement of music not attainable by just either of the algorithms. Each voice is altered to better fulfil rules of harmony and other features implemented in the dynamic programming module. The consistent movement observed in the tenor and bass parts of Figure 5 (dynamic programming only) has been altered to a more varied arrangement.

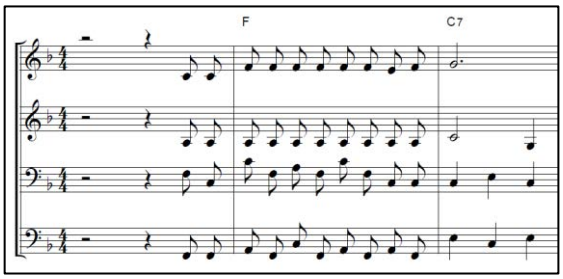

Figure 7. System output using both rule-base application and dynamic programming modules.

Figure 8 shows another example of using both modules but with some of the parameters altered. Compared to the output of Figure 7, this output has less movement in the accompaniment parts, providing an easier arrangement for amateur performers to use. Notice, though, how there is still movement in the tenor and bass parts at the end of measure two. The system has not simply allocated the same note to each part throughout the measure. This small variation in arrangement is caused by the movement in the melody line and dynamic programming searching for the best combination of notes in the last two chords of that measure, yet with as minimal movement as possible.

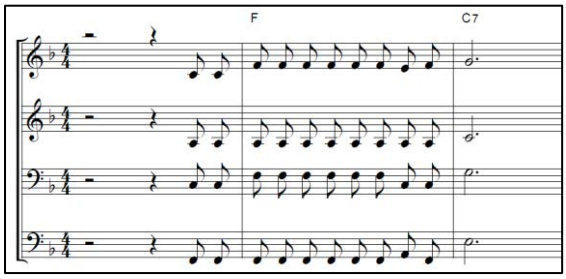

Figure 8. System output using both rule-base application and dynamic programming modules with function parameters adjusted from those used in Figure 7.

As shown above, our system can produce a variety of different arrangements from the same input, based on the parameter settings of the dynamic programming module. Currently, the tuning of parameters is done manually by users. In future, these could be learned through machine learning and applied based on different composer/genre style etc. to reflect user preferences.

We excluded the usage of non-harmonic tones in the above examples to better show the differences between different outputs. The usage of non-harmonic tones is also supported in ACC, allowing for passing notes and further arrangement diversity as shown in Figure 9. However, there are not many rules implemented in the current version of the system regarding the placement of non-harmonic tones, resulting in relatively poor arrangement quality. This we plan on improving in future versions of ACC.

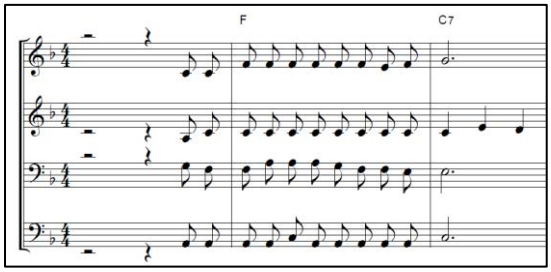

Figure 9. System output made using rule-base application and dynamic programming, with usage of non-harmonic tones.

3.2 Application

Although we have implemented rules from theories of classical music in ACC, we do not limit the scope of its application to the genre of classical music only. In fact, one of the reasons we chose lead music, not classical melodies or choral music, as the input data format, was because we recognise this type of sheet music is fairly easy to obtain for many popular and traditional genres of songs through online stores etc.

In Figure 10, we show an excerpt from a completed arrangement of music made using our system. ACC was used on a pop song found in the RWC Music Database [23, 24]. The sheet music used for input consisted of 104 measures of melody and chord signatures. Dynamic search was conducted using the distance $d=4$ for neighbouring pitch distance.

The system execution time is reasonable for real-time applications. In this song's case, ACC took roughly 60 seconds to compose the accompaniment parts and create the output data. This is much shorter than the total duration of the original song, which is 4 minutes and 49 seconds. ACC is capable of composing choral arrangements much faster than the speed a song is played at. As the search only looks at the relation of chords with those proceeding it, the output could be produced a bar at a time, or even in time to the music. This indicates our algorithm enables real-time accompaniment composition that could be played simultaneously with the original input. This could be used for situations such as impromptu ensemble sessions where the appropriate sheet music cannot be made ready on time, where members of the ensemble are not confidant with arranging their own music or where each member wants a different level/style of accompaniment to each other but still perform together.

ACC could be applied as a learning tool for composers and performers to use when learning how to compose musical arrangements themselves. Simple compositions produced by ACC could be used as a starting block for these musicians whom are not used to arranging music themselves. ACC would provide a musical arrangement with specified features for musicians to then work off when composing their own music. As the musician gains composition skills and confidence, parameters in ACC could be tuned to produce less ornamented compositions for the musician to work off, or even more ornamented compositions for the musician to learn from.

As the example in Figure 10 shows, ACC is capable of producing choral arrangements which both follows basic rules of classical music theory and also sounds interesting to the listener, even with the many different types of chord symbols. Musical features can be adjusted individually for each player in the group of musicians, and yet a uniformed sound can be maintained in the ensemble as a whole.

4. CONCLUSIONS

In this paper, we have presented a novel four-part choral music harmoniser capable of producing, in real-time, multiple outputs from the same input, with different musical features. The AutoChorusCreator utilises the benefits of using both heuristic rule-base application and dynamic programming. ACC produces harmonised choral music from lead music which both follows principals of harmony and sounds interesting. ACC works well with popular music, indicating potential for future implementation in practical applications.

Combining multiple algorithms in one system has enabled producing various outputs from a given input. In future, we plan on expanding this combined application from just harmonisation to the other processes of composition also. Manipulation of rhythm, non-harmonic tones, melodic structure etc. could all be carried out through a combination of multiple algorithms, producing an even richer diversity of compositions than we have listed in this research.

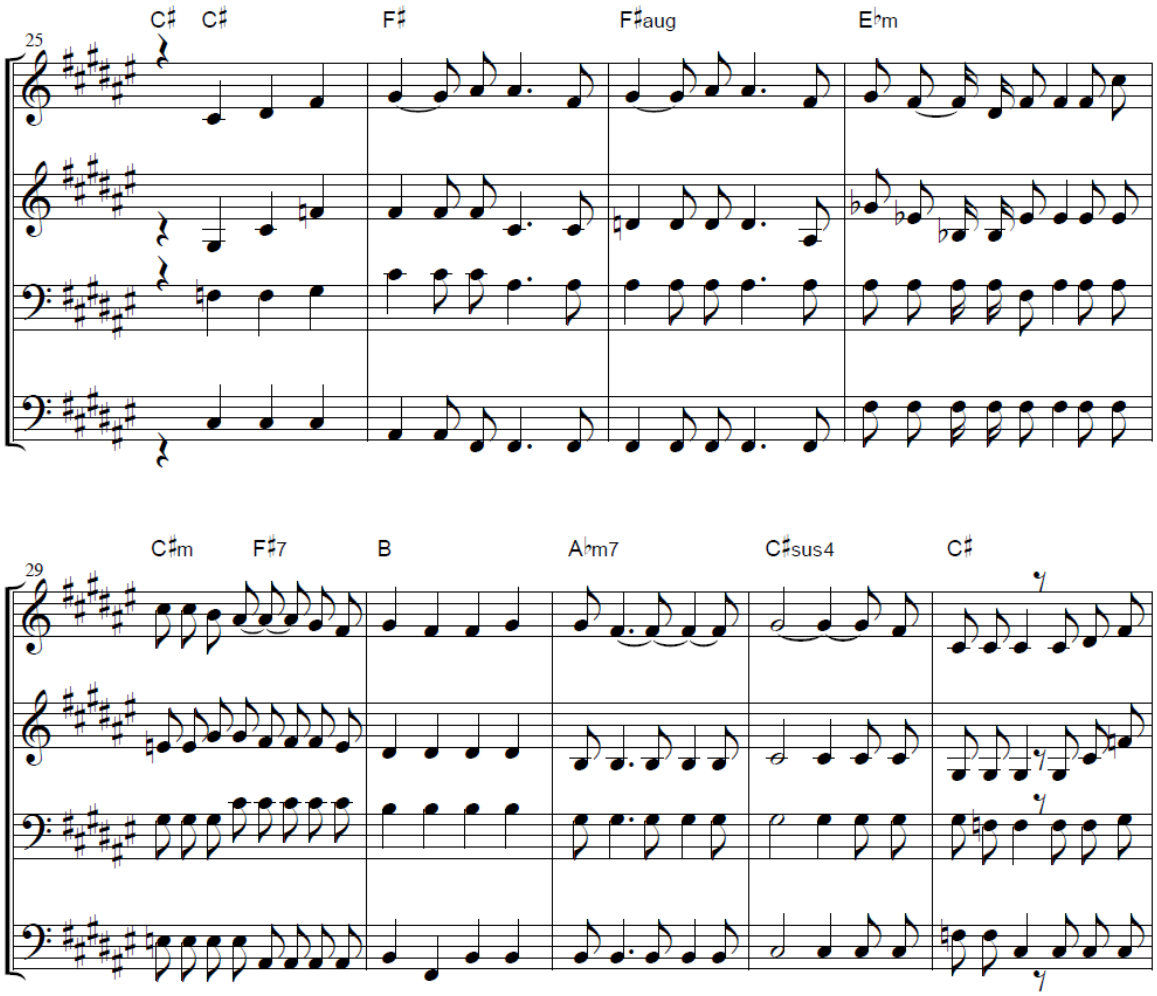

Figure 10. Example of system output using popular music as input, from "I Think of You" by Jeff Manning (RWC-MDB-P-2001 No. 87).

Though the dynamic search is fast enough for simple applications, advanced techniques of the search algorithm could be implemented to enlarge the search space and maintain the minimal execution time. Also, parameters used in the two modules can be learned from corpuses using machine learning techniques. This will enable ACC to arrange music in specific musical styles observed in the music of other composers and genres.

In this research, we included chord symbols in the input music so that we could focus on obtaining diversity within a single chord progression. Diversity of chord progressions themselves can be easily obtained, though, by implementing common chord replacement techniques already used in practical compositions. Furthermore, the system can be developed to incorporate automatic chord estimation techniques from other research such as [25, 26], so that only a melody line need be given as input. Automating a wider portion of the composition process will allow users to manipulate the outcome further to better meet their preferences.

Acknowledgments

This work was supported in part by CREST, JST.

5. REFERENCES

[1] L. Hiller and L. Isaacson, *Experimental Music; Composition with an electronic computer.* Greenwood Publishing Group, 1979.

[2] G. Nierhaus, *Algorithmic Composition.* Springer-Verlag/Wien, 2009.

[3] G. Papadopoulos and G. Wiggins, "AI Methods for Algorithmic Composition: A Survey, A Critical View and Future Prospects," in *Proc. AISB Symposium on Musical Creativity*, 1999, pp. 110-117.

[4] S. Fukayama, K. Nakatsuma and S. Sako et al., "Orpheus: Automatic Composition System Considering Prosody of Japanese Lyrics," in *Proc. Int. Conf. Entertainment Computing*, Paris, 2009, pp. 309-310.

[5] F. Pachet and P. Roy, "Musical Harmonization with Constraints: A Survey," *Constraints J.*, vol. 6, no. 1, pp. 7-19, 2001.

[6] F. Pachet, "The Continuator: Musical Interaction with Style," *J. New Music Research,* vol. 32, no. 3, pp. 333-341, 2003.

[7] D. Cope, *Machine Models of Music*. MIT Press, 1992.

[8] S. Phon-Amnuaisuk and G. A. Wiggins, "The Four-Part Harmonisation Problem: A Comparison Between Genetic Algorithms and a Rule-Based System," in *Proc. AISB'99 Symposium on Musical Creativity*, Edinburgh, 1999.

[9] J. A. Biles, "GenJam: A Genetic Algorithm for Generating Jazz Solos," in *Proc. Int. Computer Music Conference*, San Francisco, 1994, pp. 131-137.

[10] C. Ames, "The Markov Process as a Compositional Model: A Survey and Tutorial," in *Leonardo*, vol. 22, no. 2, pp. 175-187, 1989.

[11] F. Pachet, P. Roy, and G. Barbeiri, "Finite-length Markov Processes with Constraints," in *Artificial Intelligence. AAAI Press*, 2011, pp. 635-642.

[12] K. Ebcioğlu, "An Expert System for Chorale Harmonization," in *Proc. AAAI Conf. on Artificial Intelligence*, Philadelphia, 1986, pp.784-788.

[13] M. Davies, P. Hamel, K. Yoshii and M. Goto, "Automashupper: An Automatic Multi-Song Mashup System," in *Proc. Int. Conf. on Music Information Retrieva*l, Curitiba, 2013, pp. 575-580.

[14] E. Humphery, D. Trunbull and T. Collins, "Creativity in Music Informatics: Discussion Review," in *Int. Conf. on Music Information Retrieva*l *late-breaking news and demos*, Curitiba, 2013.

[15] H. Emura and M. Miura, "Recent Studies on Computer-Based Composition and Arrangement," in *J. Acoustic Society of Japan*, vol. 67, no. 6, pp. 233-238, 2011.

[16] M. Allan and C. K. I. Williams, "Harmonising Chorales by Probabilistic Inferences," in *Proc. Neural Information Processing Systems Conf.,* Vancouver and Whistler, 2004, pp. 25-32.

[17] S. Suzuki and T. Kitahara, "Four-Part Harmonization Using Probabilistic Models: Comparison of Models With and Without Chord Nodes," in *Proc. Sound and Music Computing Conf.*, Stockholm, 2013, pp. 628-633.

[18] R. M. MacCallum, M. Mauch, A. Burt and A. M. Leroi, "Evolution of Music by Public Choice," *Proc. National Academy of Sciences USA*, vol. 109, no. 30, 2012.

[19] R. Bellman, "On the Theory of Dynamic Programming," *Proc. National Academy of Sciences USA*, vol. 38, no. 8, pp.716-719, 1952.

[20] S. Fukayama, D. Saito and S. Sagayama, "Assistance for Novice Users on Creating Songs from Japanese Lyrics," in *Proc. Int. Computer Music Conference*, Ljubljana, 2012, pp. 441-446.

[21] Y. Ikeuchi and Y. Shimaoka et al., *Harmonics - Theory and Practice- (in Japanese)*. Ongakunotomo-sha, 1964.

[22] M. Miura and H. Emura, "Composition / Arrangement Systems Based on the Theory of Harmony (in Japanese)," *J. Institute of Systems, Control and Information Engineers*, vol. 56, no. 5, pp. 213-218, 2012.

[23] M. Goto, H. Hashiguchi, T. Nishimura and R. Oka, "RWC Music Database: Popular, Classical, and Jazz Music Databases," in *Proc. Int. Conf. on Music Information Retrieval*, Paris, 2002, pp.287-288.

[24] M. Goto, "Development of the RWC Music Database," in *Proc. Int. Congress on Acoustics*, Kyoto, 2004, pp. I-553-556.

[25] D. Ponsford, G. Wiggins and C. Mellish, "Statistical Learning of Harmonic Movement," *J. New Music Research,* vol. 28, no. 2, pp. 150-177, 1999.

[26] S. A. Raczyński, S. Fukayama and E. Vincent, "Melody Harmonization with Interpolated Probabilistic Models," *J. New Music Research,* vol. 42, no. 3, pp. 223-235, 2013.