http://dx.doi.org/10.5392/JKCA.2015.15.03.001

유전 알고리즘을 기반으로 한 자동 코드 악보 생성 프로그램 구현

Implementation of Automatic Chord Score Generating Program Based on Genetic Algorithm

김세훈, 김바울
한국과학영재학교

Sehoon Kim(sehoon95@naver.com), Paul Kim(paulkim3151@gmail.com)

요약

멜로디를 바탕으로 코드 악보를 생성해주는 작업은 음악의 채보 및 편곡과 직결되는 중요한 악보 작업이다. 하지만 자연스러운 코드악보 생성을 위해서는 풍부한 화성학적 배경 지식이 요구되기 때문에 음악 입문자들이 수행하기에는 큰 어려움이 따른다. 본 연구에서는 이러한 문제점을 해결하기 위해 멜로디 악보를 입력받아 자동으로 코드 악보를 생성하는 프로그램 'ACGP(Automatic Chord Generating Program)'를 개발하였다. ACGP는 유전알고리즘에 기반을 두어 다양한 화성학적 요인들과 사용자가 원하는 곡의 분위기를 효율적으로 고려할 수 있으며 이를 통해 더욱 화성학적으로 안정된 완성도 높은 코드 악보를 생성할 수 있다. 또한 편리한 사용자 인터페이스를 통하여 음악에 처음 접하는 비전문가들도 손쉽게 작업할 수 있도록 구현되었다. 또한 ACGP로 생성된 코드악보와 일반적으로 통용되는 코드악보를 비교·분석함으로써 프로그램의 적절성을 입증하였다.

■ 중심어 : | 음악 | 화성학 | 코드 악보 | 유전 알고리즘 | 소프트웨어 |

Abstract

Generating chord score based on melody is essential for composition and arrangement, while it is picky for amateurs who do not have harmonics knowledges. To solve this problem, we developed automatic chord score generating program, ACGP. Based on genetic algorithm, it successfully reflects diverse hormonic factors and the mood of the music. User interface was also implemented so that anyone can use the program conveniently. Additional analysis was conducted to prove the utility of ACGP.

■ keyword : | Music | Harmonics | Chord Score | Genetic Algorithm | Softwares |

I. 서론

음악은 현대인들에게 있어서 가장 보편적이며 대중적인 문화 및 여가 활동으로 자리 잡고 있다. 음악에 대한 대중적 관심의 증가와 더불어 일반인들에게도 단순한 악기 연주를 넘어선 작곡, 채보, 편곡과 관련된 다양한 악보 작업이 요구되고 있다. 하지만 이러한 악보 작업은 넓은 화성학적 지식과 풍부한 음악적 경험을 필요로 하기 때문에 비전문가들이 쉽게 접근하기 어렵다. 멜로디가 주어졌을 때 이에 어울리는 코드 악보를 제작하는 작업 역시 예외는 아니다.

* 본 연구는 미래창조과학부의 지원을 받아 수행되었습니다. 연구를 지도해주신 김호숙, 장진 선생님께 감사를 드립니다.
접수일자 : 2014년 11월 03일
수정일자 : 2014년 12월 04일
심사완료일 : 2014년 12월 11일
교신저자 : 김세훈, e-mail : sehoon95@naver.com

'멜로디'와 '코드'는 우리가 흔히 듣는 클래식, 대중가요, 동요 등의 음악을 구성하는 중요한 화성학적 요소 중 하나이다. 멜로디란 음악에서 주를 이루는 음의 진행이며, 코드란 멜로디를 조화롭게 꾸며주어 음악을 풍성하게 만들어주는 화음의 진행이다. 음악의 멜로디만이 주어졌을 때 이에 어울리는 코드 악보를 찾아 주는 작업은 작곡과 편곡에 있어서 매우 유의미하다. 이러한 작업을 통해 특정 멜로디에 꾸며주는 코드 악보를 부여하여 하나의 곡으로 완성시킬 수 있으며, 원곡의 멜로디에 새로운 코드악보를 입혀 전혀 다른 느낌의 음악으로 재해석 및 편곡할 수 있기 때문이다. 하지만 이는 화성음과 비화성음, 그리고 코드와 악보의 구성에 대한 이해를 필요로 하기 때문에 화성학적 지식이 부족한 비전문가들에게는 많은 시간과 노력을 요구하는 까다로운 작업이 될 수 있다.

이러한 문제점을 해결하기 위해 현재까지 많은 선행연구에서는 '마르코프 연쇄 모델'에 기반을 둔 자동 코드악보를 생성 프로그램 개발을 진행하여 왔다[1-4]. 하지만 이러한 접근은 '확률 테이블'에 의존하여 코드 악보를 생성하기 때문에 전체적인 코드 진행과 곡의 분위기, 다양한 화성학적 규칙들이 효율적으로 고려되지 못하는 한계점이 있다. 위와 같은 한계점을 극복하기 위해 본 연구에서는 유전 알고리즘에 기반을 둔 코드 악보 생성 프로그램 개발을 제안하였다. 지금까지 유전 알고리즘은 작곡 프로그램[5-7], 원곡 기반 편곡 프로그램[8], 코드 악보에 어울리는 멜로디 생성 프로그램[9], 4성부 악보 제작 프로그램[10][11] 등과 같은 다양한 악보 작업 프로그램 개발에 적용되어 그 효용성을 입증하여 왔다. 그러나 유전알고리즘을 적용한 자동 코드악보 생성 프로그램 개발 연구는 미흡한 실정이다.

따라서 본 연구에서는 유전 알고리즘을 적용하여 입력된 멜로디로부터 코드 악보를 생성해주는 알고리즘의 개발과 더불어 인터페이스를 통해 사용자로부터 편리하게 입력 값을 받아 악보를 시각화하는 '자동 코드 악보 생성 프로그램(Automatic Chord Generating Program, ACGP)'을 개발하고자 한다. ACGP는 유전 알고리즘을 활용한 자동 코드 악보 생성 프로그램의 초기 모델로서 복잡한 화성학적 요소는 배재하고 코드 생성 범위를 '다이아토닉 코드'와 '장조 조성'을 중심으로 코드 악보를 생성한다. 논문에서 제안하는 ACGP는 다음과 같은 장점을 갖는다. 첫째로 유전 알고리즘을 적용하여 다양한 화성학 법칙을 효율적으로 고려해 줄 수 있다. 따라서 생성되는 코드 악보의 완성도와 화성학적 안정도의 상향을 기대할 수 있다. 둘째로 사용자로부터 원하는 곡의 분위기를 입력받아 생성되는 코드 악보의 밝고 어두운 정도를 조절할 수 있다. 마지막으로 화성학적 배경 지식이 전혀 없는 사용자들도 쉬운 인터페이스와 프로그램 조작법으로 자신이 원하는 멜로디에 대한 코드 악보를 손쉽게 제작할 수 있다. 따라서 ACGP는 음악에 대한 배경 지식과 경험이 전혀 없는 사용자들의 작곡과 편곡과 관련된 악보 작업을 돕는 소프트웨어로 활용가능하며 비전문가 층에 대한 화성학의 진입장벽을 축소시키는 효과를 기대할 수 있다.

본 논문의 구성은 다음과 같다. 2장 관련 연구에서는 유전 알고리즘과 화성학적 이론에 대해 살펴본 후 3장 프로그램 설계에서는 프로그램에 사용된 자료구조와 알고리즘, 시각화 엔진에 대해 설명한다. 4장에서는 프로그램의 실행 화면을 중심으로 프로그램 결과에 대해 설명하며 마지막으로 5장에서는 일반인 설문조사를 통해 프로그램의 효용성을 분석한다.

II. 관련 연구

1. 자동 악보 작업 프로그램

현재까지 비전문가의 작곡, 편곡, 채보와 관련된 악보 작업을 돕는 다양한 프로그램들이 개발 및 배포되어왔으며 대표적으로는 MusicShake[12], Band-in-a-Box[13]와 Chordify[14]를 들 수 있다. MusicShake는 사용자가 원하는 멜로디 템플릿을 연결하여 자신만의 음악을 손쉽게 작곡하는 프로그램이며 Band-in-a-Box는 코드 악보를 입력하면 다양한 음색의 멜로디를 입혀 사용자가 원하는 스타일의 음악으로 재구성해주는 소프트웨어이다. Chordify는 Youtube 동영상으로부터 음들을 추출 및 재구성하여 코드 악보를 생성해주는 프로그램이다. 이의 경우 코드 악보를 생성해주는 목적 자

체는 본 연구와 비슷하나 사용자로부터 멜로디와 반주가 모두 포함된 완전한 음악을 입력받는다는 점에서 차이점을 보인다. 본 연구에서는 사용자로부터 멜로디만을 입력 값으로 받아 적합한 코드 악보를 생성하여 주는 더욱 '일반적인' 프로그램의 개발을 목표로 한다.

2. 마르코브 연쇄 기반 자동 코드악보 생성

멜로디를 바탕으로 코드 악보를 생성하는 알고리즘 개발에 있어서 다양한 연구들이 '마르코프 연쇄 모델'에 초점을 맞춰 왔다[1-4]. 이는 특정 코드가 다른 코드로 바뀌게 되는 확률을 정의한 '확률 테이블'에 근거하여 특정 코드 다음에 뒤따라올 코드를 확률론적으로 결정해주는 방법이다. 하지만 실제 코드악보는 이웃한 코드와의 관계뿐만 아니라 전체적인 코드의 진행과 곡의 분위기, 그리고 다양한 화성학적 법칙들에 의해서 결정된다. 즉, 확률 테이블에만 의존하는 마르코브 연쇄는 이러한 요인들을 모두 고려해주기 힘들다는 한계점을 보인다.

3. 유전 알고리즘

유전 알고리즘이란 생물의 진화 과정에서 착안한 알고리즘으로 특정 문제에 대한 가능한 여러 해답들을 컴퓨터상의 '염색체'로 표현한 후 적자생존법칙과 유전법칙을 적용하여 해당 문제에 최적화된 해에 도달하는 알고리즘이다[15].

유전 알고리즘을 적용하기 위해 우선 주어진 문제에 대한 유한개의 임의의 후보 해를 컴퓨터상의 '염색체'로 표현하여 '초기 개체군'을 구성한다[16]. 개체군 내에는 주어진 문제 조건에 적합한 염색체와 그렇지 못한 염색체가 동시에 존재하기 때문에 개체군을 구성하는 각각의 염색체가 주어진 문제에 얼마나 적합한지는 '적응도'로서 정량화된다. 또한 각 염색체의 적응도를 판별해주는 함수로써 적응도함수가 정의된다[17].

이후 개체군은 선택, 교차, 변이로 구성된 유전 연산이 적용된다. 선택 연산은 해당 세대의 개체군 내에서 다음 세대로 자신의 형질을 넘겨줄 염색체를 선택하는 과정으로 적응도가 높은 염색체일수록 선택될 확률이 증가한다[16]. 이렇게 선택된 염색체들은 교차 연산과 변이 연산을 통해 서로의 염색체의 일부를 교환한 새로운 '자식 염색체'를 생성하며, 이는 다음 세대의 개체군을 구성하게 된다[16-19]. 이러한 과정은 적응도가 높은 염색체가 선택적으로 자신의 형질을 다음 세대로 전달해주기 때문에 최종적으로 개체군은 주어진 문제에 최적화된 해에 수렴하게 된다.

4. 화성학 이론

'다이아토닉 코드'란 음악의 구성에 있어서 멜로디를 꾸며주는 가장 기본적인 코드이다. 하나의 장조 조표에는 총 7개의 다이아토닉 코드가 존재하며 각각은 I, ii, iii, IV, V, vi, vii° 화음으로 명명된다. [그림 1]은 다장조에서의 다이아토닉 코드를 보여준다. 이때 I, IV, V 화음은 장3화음으로 일반적으로 밝고 경쾌한 느낌을, ii, iii, iv는 단 3화음으로 일반적으로 어둡고 무거운 느낌을 준다[20]. 실제 음악의 작곡에 있어서는 다이아토닉 코드 이외에 5화음, 7화음과 감 화음, 증 화음 등을 추가하여 다양한 화성학적 효과를 부여한다. 하지만 본 연구에서 구현한 ACGP에서는 초기 모델 설계 및 알고리즘의 실효성 입증을 위하여 코드 생성 범위를 다이아토닉 코드만으로 한정하였다. 또한 생성되는 코드 악보에서 밝은 느낌의 장3화음과 어두운 느낌의 단3화음의 비율을 조절함으로써 사용자로부터 입력받은 곡의 분위기에 어울리는 코드 악보를 생성할 수 있다.

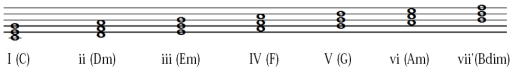

그림 1. 다장조의 다이아토닉 코드

또한 코드 악보와 멜로디가 화성학적으로 조화로운지를 판별해주기 위해서는 멜로디를 화성음과 비화성음으로 분류해주는 작업이 필요하다. 특정 멜로디를 특정한 코드가 꾸며줄 때, 멜로디의 구성 음들 중 해당 코드에 속해있는 음을 화성음, 그렇지 않은 음을 비화성음이라고 정의한다[21]. 특정한 화성학적 규칙에서 어긋나는 비화성음의 사용은 불협화음을 형성하기 때문에 ACGP에서는 입력된 멜로디를 화성음과 비화성음으로 분류하여 화성음의 비율이 높은 코드 악보일수록

보다 적합한 코드악보로 판단하여 높은 적응도를 부여하였다.

III. 프로그램 설계

본 장에서는 ACGP을 구성하는 자료구조, 알고리즘, 시각화 및 소리 생성 엔진에 대해 설명한다. 프로그램의 전체적인 구성은 [그림 2]에서와 같다.

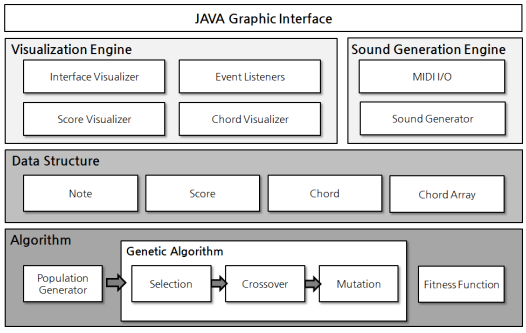

그림 2. 프로그램 구성도

1. 자료구조

악보와 코드악보를 효율적으로 저장하기 위해 4개의 객체를 정의하였다. Note, Score, Chord, ChordArray는 각각 음표, 악보, 코드, 코드 악보를 저장하는 객체이다.

1.1 Note 객체

Note는 각 음표의 정보를 저장하는 객체이다. 음표/쉼표 구분, 음의 높낮이와 길이, 붙임줄 및 임시표의 유무에 대한 정보를 저장하는 변수들로 구성되어있으며 각각에 대한 정보는 [표 1]과 같다.

표 1. Note 객체를 구성하는 변수

변수	설명
isRest	해당 음표가 쉼표인지 음표인지를 boolean으로 저장
pitch	음의 높낮이를 저장하는 0에서 6까지의 값을 갖는 정수형 변수로서, 각각은 '도' 부터 '시' 까지로 대응
state	해당 음표에 붙은 임시표의 종류를 정수형으로 저장
octave	해당 음표의 옥타브를 저장
duration	음의 길이를 저장
isTie	해당 음표가 이전 음표와 붙임줄로 연결되어 있는지를 boolean으로 저장

1.2 Score 객체

Score 객체는 사용자로부터 입력받은 악보의 정보를 저장하는 객체이다. Score 객체에서 악보를 구성하는 각각의 음표들은 Note 객체로서 표현되며 이들은 ArrayList에 의해 구조적으로 연결된다. 이는 연속된 자료를 저장하는 데에 효율적이며, 배열과는 달리 선언과 동시에 크기가 정해지지 않기 때문에 사용자로부터 입력받은 임의의 악보를 저장하기에 가장 적합한 자료구조로 판단된다. [그림 3]은 악보가 Score 객체에 의해 저장되는 구조도를 보여준다.

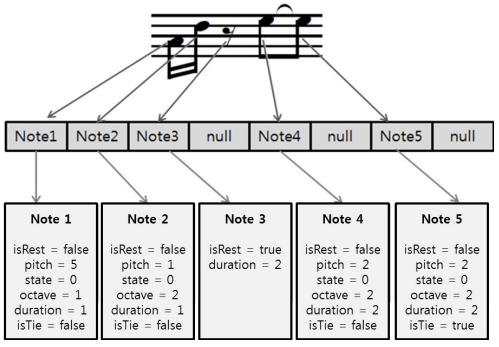

그림 3. 악보가 Score 객체에 저장되는 구조도

1.3 Chord 객체

Chord 객체는 코드의 정보를 저장하는 객체이다. 코드는 하나 이상의 음으로 구성되어있기 때문에 Chord 객체에서는 해당 코드의 구성음의 정보를 Note 객체로 표현하고 이를 배열로 묶어서 저장한다. [그림 4]는 C 코드가 Chord 객체에 저장되는 구조도이다.

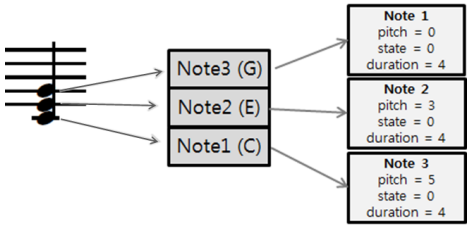

그림 4. C코드가 Chord 객체에 저장되는 구조도

1.4 ChordArray 객체

ChordArray 객체는 사용자로부터 악보를 입력받아

이에 적합한 코드 악보를 출력해주는 과정에 있어서 코드 악보를 저장해주는 기능을 수행한다. 코드 악보를 구성하는 각각의 코드들은 Chord 객체로서 표현되며 이들은 Chord형 ArrayList에 의해 구조적으로 연결된다.

2. 알고리즘

본 연구에서는 사용자가 입력한 멜로디에 최적화된 코드 악보를 생성해주기 위해 유전 알고리즘을 활용하였다. 이를 위해 우선적으로 코드 악보의 집합으로 개체군을 구성해 주었으며 개체군을 구성하는 각 코드악보의 적응도를 계산하는 적응도 함수를 정의하였다. 마지막으로 유전알고리즘을 진행하여 개체군의 적응도를 증가시켜주었다. 각 단계에 대한 설명은 다음과 같다.

2.1 개체군 생성 알고리즘

멜로디 악보의 한 마디는 [그림 5]의 첫 번째 마디와 같이 단 하나의 코드에 대응될 수도 있지만 두 번째 마디와 같이 2개 이상의 코드에 대응되는 경우도 있다. 전자의 경우는 한 마디를 구성하는 모든 구성 음들이 특정 코드에 화성학적으로 조화를 이루는 경우이며, 두 번째 마디의 모든 구성 음들을 화성학적으로 연결해줄 수 있는 유일한 코드가 존재하지 않는 경우이다.

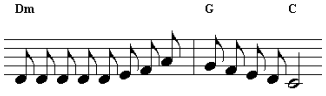

그림 5. 동요 '꼬까신'의 악보와 코드악보

ACGP의 개체군 생성 알고리즘에서는 전자와 후자의 경우를 모두 고려해주기 위해 임의의 코드 악보 생성 단계에서 주어진 멜로디 악보의 임의의 한 마디를 하나의 코드로 대응시킬지 둘 이상의 코드로 대응시킬지를 결정하여준다. 이는 [그림 6]과 같이 구현된다. 우선 입력된 악보의 각 마디를 '선두 반 마디'와 '후미 반 마디'로 나누어준다. 이후 선두 반 마디에는 임의의 코드를 대응시켜주며 이때의 코드가 뒤따라오는 후미 반 마디와 화성학적으로 어울리지 않는 경우에만 후미 반 마디에도 임의의 코드를 대응시켜준다. 이러한 과정을 유한번 반복하여 1세대 개체군을 생성한다.

```
Population(Score s)
  pop := ChordArray[n] // pop:개체군, n:개체군의 크기
  for (i:=0 to n-1)
    initialize pop[i]
    for (j:=0 to s.length-1)
      /* 선두 반 마디에 임의의 코드 대응*/
      let Chord c : 임의의 다이아토닉 코드
      pop[i][2*j] := c
      let k : s의 j번째 후미 반 마디에서
              c에 대한 비화성음들의 길이의 합
      /* 후미 반 마디에 null 대응*/
      if (k<tension) // tension:사전에 정의된 값
        pop[i][2*j+1] := null
      /* 후미 반 마디에 임의의 코드 대응*/
      else
        let Chord c2 : 임의의 다이아토닉 코드
        pop[i][2*j+1] := c2
return pop
```

그림 6. 개체군 생성 알고리즘의 의사코드

2.2 적응도함수 정의

ACGP에서의 적응도함수는 코드악보가 사용자가 입력해준 멜로디 악보와 얼마나 화성학적으로 조화로운지를 '적응도'로서 수치화하는 함수이다. 코드악보의 화성학적 적절성은 매우 추상적인 개념이므로 이를 수치화하기 위해서는 적절한 코드 악보가 갖춰야 하는 기준들을 [표 2]와 같이 세분화하여서 분석하였으며 [그림 7]은 적응도함수의 알고리즘에 대한 의사코드이다.

표 2. Note 객체를 구성하는 변수

기준	설명
화성음의 비율	입력된 멜로디 악보에 화성음이 많이 포함된 코드악보에 높은 적응도를 부여한다.
종지 코드	코드 악보의 마지막 코드 진행이 종지(I, V→I, IV→V→I)로 끝나는 코드악보에 높은 적응도를 부여한다.
반종지 코드	곡의 중간부에 반종지를 알리는 V코드 대신 종지를 알리는 I코드가 오는 코드악보에 낮은 적응도를 부여한다.
코드 사이의 간격	동일한 코드가 연속적으로 사용되거나 코드가 연속적으로 한 단계씩 증가 혹은 감소하는 경우 코드악보의 화성학적 안정성이 감소한다[18]. 이러한 경우 코드악보에 낮은 적응도를 부여한다.
불안정한 코드진행	V혹은 vii° 코드에서 ii혹은 iv코드로의 불안정한 코드진행이 있는 코드 악보에 낮은 적응도를 부여한다.
안정적인 코드진행	ii→V, iii→vi, vi→ii, vii° →iii의 안정적인 코드 진행이 있는 코드 악보에 높은 적응도를 부여한다.
반복 마디	악보에 반복되는 멜로디가 존재한다면 반복되는 멜로디에 동일한 코드가 사용되는 코드악보에 높은 적응도를 부여한다.
곡의 분위기	사용자가 입력한 곡의 분위기가 밝은 경우 장3화음의 사용 빈도가 높을수록, 어두운 경우 단3화음의 사용빈도가 높을수록 코드악보에 높은 적응도를 부여한다.

```
fitnessFunction(Score s, Chord c)
  initialize int fitness == 0 // 적응도
  initialize int f1, f2, f3, f4, f5, f6, f7 // 요인별 가중치
  for(i:=0 to c.length)
  /* 1. 화성음의 비율 */
    for each Note j in ith node of Score s
      if(j is in i) // j가 화성음이면
        fitness += f1*j.length
  /* 2. 종지코드 */
    if(i==c.length and c[i]==1화음)
      fitness += f2
  /* 3. 반종지코드 */
    double l1, l2, l3 : Score s에서 1음, 3음, 5음이 차지하는 비율
    if(l1>=0.75 or l2>=0.75 or l3>=0.75)
      if(i!=c.length and c[i]==1화음)
        fitness −= f3
  /* 4. 불안정한/안정적인 코드진행 */
    c1 := c[i], c2 := c[i−1], c3 := c[i+1]
    if((c1==5화음 and c2==2화음) or (c1==7화음 and
c2==2화음)      or (c1==5화음 and c2==4화음) or (c1==7화
음 and c2==4화음))
        fitness −= f4
    if((c1==2화음 and c2==5화음) or (c1==3화음 and
c2==6화음)      or (c1==6화음 and c2==2화음) or (c1==7화
음 and c2==3화음))
        fitness += f5
  /* 5. 코드 사이의 간격 */
    d1 := c1과 c2 사이의 간격
    d2 := c1과 c3 사이의 간격
    if((d1:=0 and d2:=0) or (d1:=1 and d2:=1))
      fitness −= f6
  /* 6. 반복 마디 */
  for(int i := 0 to c.length/4)
    for(int j := i to c.length/4)
      if(Score s의 i번째 마디와 j번째 마디가 일치)
        if(c[i]==c[j])
          fitness += f7
  return fitness
```

그림 7. 적응도함수 알고리즘

```
cross(ChordArray parent1, ChordArray parent2)
//parent1, parent2 : 두 부모 염색체
  initialize ChordArray offspring // 자식염색체
  initialize double p : 교차 확률 (0과 1사이 상수)
  initialize int isParent1 := 1
  for(i:=0 to parent1.length−1)
    if(Math.random( )<p) // 교차 발생
      isParent1 := (isParent1+1)%2
    if(isParent1 == 1)
      offspring[i] := parent1[i]
    else
      offspring[i] := parent2[i]
return offspring

mutation(ChordArray offspring)
  let double p : 0에서 1사이의 상수 // 변이 확률
  for(i:=0 to offspring.length−1)
    if(Math.random( )<p) // 변이 발생
      offspring[i]:= 임의의 다이아토닉 코드
```

그림 8. 교차 및 변이함수의 알고리즘

2.3 유전 알고리즘

개체군을 구성하는 코드 악보들의 적응도가 모두 구해지면 이를 바탕으로 선택, 교차, 변이의 유전 연산을 적용하여준다. 먼저 선택 연산을 통해 개체군에서 두 개의 코드 악보를 적응도에 비례하는 확률로 택하여 '부모 염색체'로 삼는다. 이들에 교차 연산을 적용하여 '자식 염색체'에 해당하는 새로운 코드악보를 생성하여 준다. 이때 두 염색체 위에서 임의로 잡은 교차 발생 지점을 기준으로 두 염색체의 형질을 섞어준다. 마지막으로 변이 연산을 통해 교차 연산으로 생성된 자식염색체의 코드 악보에서 임의의 위치의 코드를 새로운 코드로 바꾸어준다. [그림 8]은 교차 및 변이 함수의 알고리즘에 대한 의사코드이다. 위와 같은 과정을 거쳐 생성된 자식 염색체의 코드악보는 다음 세대의 개체군을 구성하게 된다.

3. 시각화 엔진 및 음악 재생 엔진

ASGP의 사용자 인터페이스는 JAVA Swing 툴의 javax.swing 패키지가 제공하는 그래픽 객체를 통해 구현되었으며 크게 '메인 창'과 '메뉴 바'로 구성되어있다. 메인 창은 악보 및 코드 악보를 시각화하는 기능을 수행하며 메뉴 바는 버튼과 슬라이더를 통해 사용자로부터 입력 값을 받아 자료구조 및 알고리즘 담당 파트로 전달해주는 기능을 수행한다.

또한 ACGP에는 MIDI 입출력을 통한 음악 재생 기능을 갖추고 있다. MIDI란 전자 악기들과 컴퓨터들 사이에 음악적 신호를 주고받기 위해 규격화된 표준 신호 체계로[22] 본 프로그램에서는 javax.sound.midi 패키지의 MidiChannel 인터페이스를 사용하여 악보 및 코드 악보를 MIDI 출력 값으로 변환 및 소리를 재생하였다.

IV. 프로그램 구현 결과

이번 장에서는 완성된 프로그램의 실행 화면과 인터페이스 구성 요소에 대해 소개한다. 해당 프로그램의 서버 url은 다음과 같다.

http://gaonnuri.ksain.net/ChordGenerator/

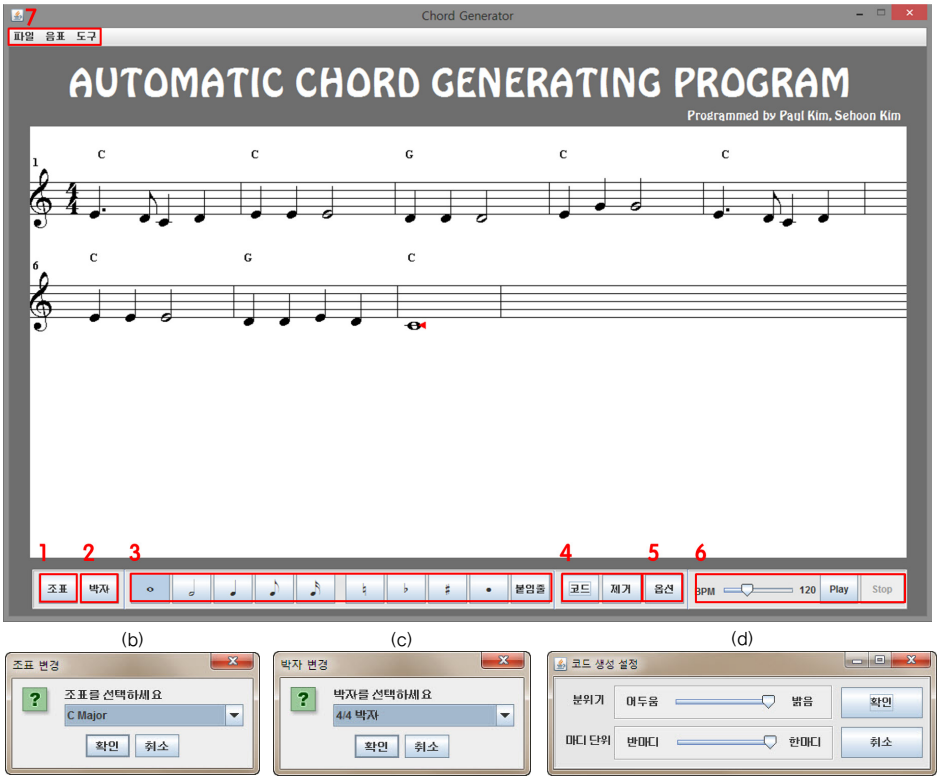

그림 9. (a) 프로그램 실행 화면 (b) 조표 선택 팝업 (c) 박자 선택 팝업 (d) 코드 악보 생성 옵션 팝업

[그림 9]는 프로그램 실행 화면 및 팝업'창을 보여준다. 사용자 인터페이스는 [그림 9(a)]와 같으며 1에서 7까지의 번호가 표기된 버튼 및 슬라이더 각각의 기능에 대한 설명은 다음과 같다.

(1) 조표 입력 기능 : [그림 9(b)]와 같은 팝업 창을 띄워 사용자로부터 악보의 조표를 입력받는다. C, G, D, A, E, B, F#, C#, F, B♭, E♭, A♭, D♭, G♭, C♭의 총 15개의 장조 조표를 입력할 수 있다.

(2) 박자 입력 가능 : [그림 9(c)]와 같은 팝업 창을 띄워 사용자로부터 악보의 박자를 입력받는다. 4/4, 3/4. 2/4의 총 3개의 박자를 입력할 수 있다.

(3) 음표 입력 기능 : 온음표, 2분음표, 4분음표, 8분음표, 16분음표 및 쉼표를 입력할 수 있으며 임시표(♮, ♭, #)와 점음표, 그리고 붙임줄을 추가할 수 있다. 입력된 음표는 실시간으로 메인 창에 시각화 된다.

(4) 코드 악보 출력 및 제거 기능 : '코드' 버튼은 입력된 멜로디에 알맞은 코드 악보를 생성 및 출력해주는 기능으로, [그림 9(a)]와 같이 메인 창에 시각화된다. '제거' 버튼은 출력된 코드 악보를 제거해주는 기능이다.

(5) 코드 악보 생성 옵션 조절 기능 : [그림 9(d)]와 같은 팝업을 띄워 사용자로부터 코드 악보의 생성 옵션을 입력받는다. 곡의 분위기를 어두움부터 밝음까지 총 5단계로 입력이 가능하며, 코드 악보 길이의 최소 단위(한마디, 반마디)의 비율을 총 5단계로 입력 가능하다.

(6) 음악 재생 기능 : 사용자는 곡의 재생 속도(bpm)를 60에서 240으로 입력 가능하며 입력한 악보와 출력된 코드악보를 동시에 재생할 수 있다.

(7) 저장 및 불러오기 기능 : 악보를 저장 또는 불러오기 할 수 있으며 이미지 파일로도 내보낼 수 있다.

V. 프로그램 효용성 분석

논문에서 제안한 AGCP의 효용성을 검토하기 위해 본 연구에서는 다음과 같은 '설문조사' 분석 실험을 설계 및 진행하였다. 현존하는 음악은 그 수가 무수히 많아 모든 곡에 대하여 프로그램의 효용성을 분석할 수 없으며, 따라서 본 연구에서는 일반인들에게 친숙한 대중음악과 동요를 토대로 샘플 음악 10곡을 선정하였다. 이때 대중음악의 경우 대표적인 장르인 락, 발라드, 어쿠스틱, 대중가요(pop), 댄스에서 대표성을 띄며 대부분의 일반인들에게 익숙한 멜로디를 갖는 곡을 기준으로 선정하였다. [표 3]은 이렇게 선정된 샘플 곡의 명단과 장르를 보여준다.

이렇게 선정된 곡들에 대해 아래와 같은 방법으로 설문조사를 진행하였다. 우선 ACGP를 활용하여 코드악보를 생성하였으며 ACGP 생성 코드악보와 일반적으로 통용되는 코드악보를 각각 멜로디와 함께 임의의 순서로 피실험자에게 들려주었다. 이후 피실험자로부터 각각의 코드악보에 대한 자연스러운 정도를 1점에서 5점까지의 점수로서 부여받았다. 10곡의 샘플 음악에 대하여 총 30명의 피실험자에게 위와 같은 설문조사를 진행하였으며 이를 토대로 ACGP 생성 코드 악보에 대한 임의의 일반인들의 주관적인 평가를 분석할 수 있었다.

표 3. 샘플 곡들에 대한 설문조사 응답 결과 및 점수 차이와 p 양측 검정 값

곡 이름	장르	ACGP 생성 코드 점수	통용되는 코드 점수	점수 차이 (%)	p 양측 검정
비행기	동요	3.80	3.60	5.26	0.34
학교종	동요	3.57	2.33	34.7	1.3E-5
나비야	동요	4.10	3.40	17.1	0.0056
나에게로...	락	3.83	3.93	-2.61	0.67
비와 당신	발라드	4.13	2.88	30.0	2.5E-7
너에게 난...	어쿠스틱	3.33	3.47	-4.20	0.66
봄봄봄	어쿠스틱	4.10	3.26	20.5	0.00037
네모의 꿈	대중가요	3.47	2.73	21.3	0.012
붉은 노을	대중가요	3.93	3.00	23.7	0.00015
Mr.Chu	댄스	3.83	3.40	11.2	0.052
평균	-	3.80	3.20	16.0	-

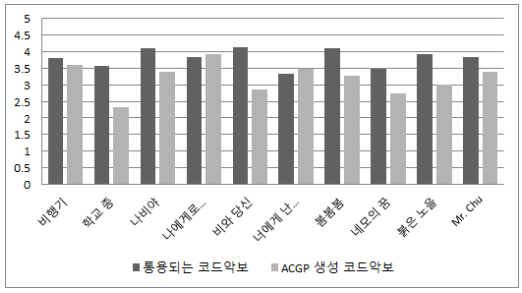

그림 10. 샘플 곡들에 대한 설문조사 응답 결과

[표 3]와 [그림 10]은 각각의 노래에 대한 설문조사 분석 결과를 보여준다. [그림 10]에서 볼 수 있듯이 '나에게로 떠나는 여행'과 '너에게 난 나에게 넌'의 두 곡에서는 ACGP 생성 프로그램이 오히려 더욱 높은 점수를 획득하였으며 이 두 곡에 대해서는 ACGP가 통용되는 코드 악보만큼의 자연스러운 코드 악보를 생성하였음을 명시한다. 나머지 8곡의 경우 ACGP 생성 코드 악보가 통용되는 코드 악보에 비해 비교적으로 낮은 점수를 획득하였는데, 이를 점수 차이의 정량화와 t테스트를 통해 다음과 같이 분석해 보았다.

우선 점수 차이는 오차 공식과 동일하게 {(통용되는 코드악보의 평균 점수)-(ACGP 새성 코드악보의 평균 점수)}/(통용되는 코드 악보의 평균 점수)로서 정량화하였으며 각각의 샘플 음악에 대한 그 값들은 [표 3]에서와 같다. '학교 종'을 제외한 대부분의 곡들의 ACGP 생성 코드악보의 점수는 통용되는 코드악보가 받은 점수에 대해 30% 이내의 차이를 보였으며 평균적으로는 16%의 작은 점수 차이를 보였다.

추가적으로 각각의 음악에 대한 설문조사 결과를 t-테스트로 분석해본 결과 10개의 곡 중 '비행기', '나에게로 떠나는 여행', '너에게 난 나에게 넌', '네모의 꿈', 'Mr.Chu'의 5개의 곡이 0.05 이상의 p 양측 검증 수치를 보였다[표 3]. 이는 이들 5곡에 대해서는 ACGP 생성 코드악보와 통용되는 코드악보가 통계학적으로 유의미한 차이를 보이지 않음을 의미한다. 따라서 점수 차이와 p 양측 검증 수치를 통해 많은 사람들이 ACGP에 의해 생성된 대부분의 코드 악보에 대해서 통용되는 코드악보에 근접한 자연스러움을 느낀다고 결론내릴 수 있으며 이는 ACGP의 실효성을 잘 뒷받침해 준다.

V. 결론 및 향후 연구

본 연구에서는 사용자로부터 악보를 입력받아 화성학적으로 적절한 코드 악보를 자동으로 생성해주는 프로그램 ACGP를 개발하였다. 프로그램의 알고리즘은 코드 악보 생성에 필요한 다양한 화성학적 변수들과 곡의 분위기, 그리고 악보의 전체적인 짜임새를 보다 잘 고려해줄 수 있는 '유전 알고리즘'에 기반을 두었으며 사용자의 편의를 위해서 JAVA Swings 툴을 기반으로 사용자 인터페이스를 제작하였다.

프로그램의 적절성을 검토하기 위해 10개의 샘플 음악에 대해 ACGP에 의해 생성된 코드 악보와 실제 통용되는 코드 악보의 자연스러운 정도를 일반인 대상 설문조사를 통해 분석하였다. 그 결과 ACGP 생성 코드악보가 받은 점수는 통용되는 코드악보가 받은 점수에 비해 평균적으로 16% 이내의 작은 차이를 보였으며 이들 중 5개의 곡에 대해서는 t-테스트의 결과 유의미한 점수 차이를 보이지 않음이 입증되었다. 즉, 해당 분석 결과는 ACGP의 효용성을 뒷받침해준다.

본 프로그램은 유전알고리즘을 기반으로 하여 다양한 화성학적 요인들을 고려함으로써 자연스러운 코드 악보 생성이 가능하다. 이는 마르코브 확률 테이블을 바탕으로 직전에 오는 코드에 의해 연쇄적으로 코드 악보를 생성하는 사전 연구의 한계점을 개선하여 안정적인 코드 진행뿐만 아니라 곡의 분위기와 종지 및 반종지 코드, 화성음과 비화성음 등의 화성학적 요인들을 폭넓게 고려해 줄 수 있었다. 또한 해당 프로그램은 사용자가 원하는 분위기의 코드 악보를 생성할 수 있으며 편리한 인터페이스를 통해 누구나 손쉽게 사용 가능하다. 따라서 ACGP은 화성학적 지식과 경험이 부족한 비전문가의 채보와 편곡 등의 악보 작업을 돕는 프로그램으로서 활용가능성을 보인다.

본 연구에서 제안한 ACGP는 유전 알고리즘을 바탕으로 하는 코드 제작 알고리즘의 실효성을 입증하기 위해 복잡한 화성학적 요소를 일부 생략하고 설계된 초기 모델이다. 따라서 다양한 화성학적 요소를 추가적으로 고려하여 프로그램의 성능을 향상할 수 있는 잠재력을 갖고 있으며 이를 위해서는 다음과 같은 향후 연구가 필요할 것으로 전망된다.

우선적으로 알고리즘에 다양한 고급스러운 화성학적 요소들을 고려해 준다. 실제로 악보는 다이아토닉 코드뿐만 아니라 7화음, 증화음, 감화음 등의 다양한 코드로 구성되어 있으며 단조조성을 사용하여 우울한 분위기를 자아내기도 한다. 이러한 화성학적 장치들을 고려해 줌으로써 프로그램에 의해 생성되는 코드악보의 화성학적 안정도와 다양성을 높일 수 있을 것이다. 또한 코드 악보를 반 마디보다 작은 단위로 생성해 줌으로써 더욱 다양하고 안정적인 코드 악보를 생성해 낼 수 있을 것으로 고려된다. 마지막으로 음악에는 다양한 장르가 존재하며 각 장르별로 사용되는 화성학적 규칙이 다소 다르다. 따라서 사용자로부터 곡의 장르를 입력받아 서로 다른 화성학적 규칙에 의한 코드 악보를 생성해주는 방향의 연구가 필요할 것으로 보인다.

참 고 문 헌

[1] Ian Simon, Dan Morris, and Sumit Basu, "MySong: automatic accompaniment generation for vocal melodies," SIGCHI Conference on Human Factors in Computing Systems, ACM, pp.725-734, 2008.

[2] Arne Eigenfeldt and Philippe Pasquier, "Realtime generation of harmonic progressions using controlled Markov selection," Int. Conf. on Computational Creativity, pp.16-25 2010.

[3] 김나리, 권지용, 유민준, 이인권, 황정규, "강화 학습을 통한 자동 반주 생성", 한국 HCI 학회 학술대회, pp.739-743, 2008.

[4] Raymond P. Whorley, Geraint A. Wiggins, and Marcus T. Pearce, "Systematic evaluation and improvement of statistical models of harmony," international joint workshop on computational creativity, pp.81-88, 2007.

[5] Andrew Horner, "Genetic algorithms and computer-assisted music composition,"

Urbana, Vol.51, No.61801, 1991.
[6] Bruce Jacob, "Composing with genetic algorithms," International Computer Music Conference, pp.452-455, 1995.
[7] John Biles, "GenJam: A genetic algorithm for generating jazz solos," International Computer Music Conference, pp.131-131, 1994.
[8] 정재훈, 이종현, 안창욱, "유전 프로그래밍을 이용한 원곡 기반의 새로운 멜로디 생성", 한국컴퓨터종합학술대회 논문집, pp.1565-1567, 2013.
[9] George Papadopoulos and Geraint Wiggins, "A genetic algorithm for the generation of jazz melodies," STEP 98, 1998.
[10] Ryan A. McIntyre, "Bach in a box: The evolution of four part baroque harmony using the genetic algorithm," IEEE Conference, Vol.2, pp.852-857, 1994.
[11] Phon-Amnuaisuk, Somnuk, and Geraint Wiggins, "The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system," AISB'99 Symposium on Musical Creativity, pp.28-34, 1999.
[12] http://www.musicshake.com
[13] http://www.band-in-a-box.com
[14] http://www.chordify.net
[15] John H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, U Michigan Press, 1975.
[16] Michael Affenzeller, S Wagner, S Winkler, and A Beham, *Genetic algorithms and genetic programming: modern concepts and practical applications*, Crc Press, 2009.
[17] David Beasley, R. R. Martin, and D. R. Bull, "An overview of genetic algorithms: Part 1. Fundamentals," University computing, Vol.15, pp.58-58, 1993.
[18] Austin Scott, "An introduction to genetic algorithms," AI expert, Vol.5, No.3, pp.48-53, 1990.
[19] Kim-Fung Man, Kit-Sang Tang, and Sam Kwong, "Genetic algorithms: concepts and applications," IEEE Transactions on Industrial Electronics, Vol.43, No.5, pp.519-534, 1996.
[20] 이동민, *대중음악을 위한 화성학*, 음악세계, 2013.
[21] 이교숙, *알기 쉬운 편곡법*, 세광음악출판사, 2002.
[22] http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol1/aps2/

저 자 소 개

김 세 훈(Sehoon Kim) 준회원

▪ 2011년 2월 ~ 현재 : 카이스트 부설 한국과학영재학교(재학 중)

<관심분야> : 멀티미디어 콘텐츠

김 바 울 (Paul Kim) 준회원

▪ 2012년 2월 ~ 현재 : 카이스트 부설 한국과학영재학교(재학 중)

<관심분야> : 멀티미디어 콘텐츠