코드기반 작곡을 위한 트리구조 프로그램 구현

조재영* · 김윤호* · 강희조* · 이명길**
* 목원대학교 IT 공학부 · ** 대덕대학

The embodiment of tree construction program for the composition by cord basis

Jae-young Jo* · Youn-ho Kim* · Hee-jo Kang* · Meung-gil Lee**
race4808@hanmail.net

요 약

본 논문은 미디라는 컴퓨터 작곡 시스템으로 일반인들도 작곡을 보다 쉽게 할 수 있는 방향이 주어졌음에도 음악을 전문적으로 배워오지 않은 사람들은 음악 작곡을 하기 힘들다는 점에 착안하여 본 논문을 통해 코드작곡에 있어서 사람이 아닌 컴퓨터가 모든 것을 할 수 있는 프로그램을 지금까지 나온 기존의 곡들의 코드진행 방식을 데이터베이스로 하여 구현함으로 해서 음악을 전문적으로 배워오지 않은 일반인들이 작곡을 함에 좀 더 편할 수 있는 방향을 제시하고자 한다.

ABSTRACT

The appearance of the computer composition system(MIDI) expects that non-musicians can make musics easily, but, in fact, non-musicians still compose hard. This thesis shows that a computer makes cord progerss through a database, which in-putted all practicable cord progress. In other words, non-musicians just makes some melodies over the settled cord progress, so they can make a music more easely.

키워드

대중음악, 코드작곡, 트리 구조 진행, 쿼드트리, 코드진행패턴

Ⅰ. 서 론

모든 장르의 대중음악은 코드진행을 기본으로 한다. 그리고 모든 음악 코드들은 정의되지는 않았지만 적어도 일반적인 진행공식을 가지고 있다. 밝은 음색을 표현하는 메이저코드와 어둡고 무거운 음색을 표현하는 마이너코드등을 적절히 사용하여 기존의 대중음악은 만들어진다. 과거 바흐나 헨델 등의 음악가가 있던 바로크 시대를 예로 들어보면 그 당시에 음악의 작곡법은 A다음에 B가 나오는 것은 잘못되었다 등의 어떠한 무형의 법칙이 정해져 있었다. 하지만 최근 대중음악에 있어서 작곡의 자유도 란 말 그대로 굉장히 자유롭다. 대중들이 성악가의 완성된 목소리보다 가수들의 길들여지지 않은 허스키 보이스를 더욱 선호하듯 대중음악은 말 그대로 대중들에게 사랑 받을 수 있느냐 없느냐, 대중들에게 얼마나 어필 할 수 있느냐가 관건이다.

하지만 그렇다고 대중음악의 작곡에 있어서 과거로부터 지금까지 전해 내려오는 대위법이나 화성악 등의 작곡방법이 완전히 무시되는 것은 아니다. 컴퓨터 음악이 발달되면서 음악에 대한 전반적인 이해를 가지고 있지 않은, 즉 어려서부터 필수 교육과정 외의 음악수업 외에는 배워오지 않은 일반 사람들도 음악작곡을 할 수 있게 되었다. 하지만 이들이 컴퓨터 음악을 이용해 작곡한 음악을 들어봐도 그리 어색하지 않음을 느끼는 것은 이들 역시 비록 과외 음악 교육은 받지 않았을지 모르지만 어려서부터 들어오던 여러 음악들에 의해 자신도 모르게 어느 정도의 감각적인 교육은 받았다는 것이 본인의 생각이다.

그러므로 그들이 음악의 코드를 배워서 작곡을 한다해도 그렇게 어색하지 않은 코드진행이 나오는 경우가 대부분이다. 그 이유가 바로 대위법이나

화성악을 기반으로 작곡되어진 여러 음악들을 예전부터 접해왔다는 이유 때문일 것이다. 어려서부터 음악을 들어온 사람들이 대부분이듯이 그런 사람들은 너무 많이 일반적 에서 벗어나는 음악을 듣는다면 거부감을 느낄 것이다. 반대로 그들이 작곡을 한다면 그들 역시 그런 어색한 음악을 만들지는 않을 것이다.

작곡이라 함은 예술인들의 전유물이라고 일반적으로 생각되어지고 있다. 하지만 본 논문에서는 음악적인 교육을 전문적으로 받아오지 않은 사람이라도 누구나 작곡을 할 수 있는 프로그램을 구현함으로 해서 전문 작곡가가 아닌 사람들도 코드 작곡에 한해서 보다 쉽게 작곡을 할 수 있는 방법을 제시한다.

II-1 대중음악의 코드 패턴

음악작곡은 크게 멜로디 작곡법과 코드 작곡법으로 구분되어진다. 코드 작곡이란 정의되어져 있는 음악코드를 이용해 작곡을 하는 것이고 멜로디 작곡이란 음 하나하나를 입력하여 작곡을 하는 것이다. 여기서 코드란 화음을 말하는 것이다. 예를 들어 C코드는 도, 미, 솔, 이 세 가지 음을 동시에 눌러 내는 소리이며 G코드는 솔, 시, 레, 세 가지 음을 동시에 누름으로서 내는 소리이다. 코드진행에는 정확히 정의되지는 않았지만 적어도 어색한 코드진행과 어색하지 않은 코드진행이 있다. 예를 들어 C코드 다음에 F코드가 나오는 것은 일반적인 코드진행으로 전혀 어색하지 않은 코드진행이지만 C코드 다음에 Bb코드가 나온다면 그것은 거의 사용되지 않는 어색한 코드진행이다. 어떠한 규칙이 없는 대중음악작곡에 있어서 코드진행이 어색하지 않아야 할 필요성이 없다라고 말할 수도 있겠지만 적어도 어색한 코드진행으로 만들어진 곡은 들어주는 사람이 없다는 것만으로도 코드진행은 어색하지 않게 표현되어야 할 필요가 있다. 서론부 에서도 언급했지만 대중음악이라는 것은 대중들이 듣지 않는다면 그건 이미 대중음악으로서의 가치가 없는 것이다. 코드진행에 있어서 어색한 코드진행과 어색하지 않은 코드진행은 물론 개인차가 있겠지만 본 논문에서는 최근 1년간 가요순위 10위권 안에 들었던 곡들을 100곡 선곡하여 대중음악에서 가장 많이 쓰이는 코드진행을 베이스로 하여 만들어진 어색하지 않은 코드진행을 다음과 같은 표 1로 표현하였다.

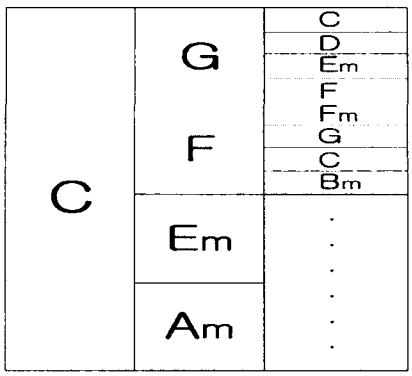
시작코드	진행 가능한 어색하지 않은 코드
C	G, F, Em, Am
D	G, Em, Gm, C
E	Am, D, A, F
F	Fm, G, C, Bm
G	C, Dm, Em, F
A	Dm, F, E, D
B	Gm, Em, Cm, G
Cm	Fm, G, Gm, D
Dm	Gm, Am, G, C
Em	Am, A, Cm, F
Fm	C, G, Em, Gm
Gm	Am, Em, C, D
Am	Dm, Em, D, F
Bm	Gm, E, Cm, G

<표 1. 코드진행도식>

표 1에서는 어색하지 않은 코드진행이라는 이름하에 한 코드에서 이동할 수 있는 코드진행 가능 코드를 4가지씩 제시하였다.

그리고 이 코드진행도식은 앞에서도 언급하였듯 최근 1년간 가요순위 10위권 안에 들었던 가장 대중적인 곡들을 기반으로 만들어진 것이다. 단순히 메이저 코드와 마이너 코드들로만 구성하였으며 그 외에 코드들은 사용하지 않았다.

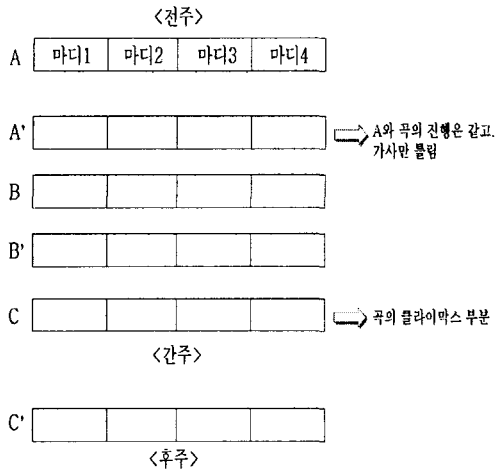
하나의 코드에서 다른 코드로의 어색하지 않은 진행패턴을 4가지씩 부여했을 경우 초기 값을 C코드로 줬다는 가정 하에 다음 그림1과 같이 트리구조로 진행 가능 코드를 표현할 수 있다.



<그림 1. 트리구조 코드진행>

II-2 작곡을위한 코드나열

대중음악에서의 코드나열 방식은 어떠한 패턴이 있다. 항상 똑같은 패턴이 존재하는 것은 아니지만 일반적으로 많이 쓰이는 패턴중 하나는 다음과 같은 패턴이다.



<그림 2. 대중음악의 코드나열 패턴>

그림2 에서 첫 패턴A 와 두 번째 패턴A′ 는 유사한 멜로디를 가지고 있지만 가사는 다른, 즉 코드진행은 같고 가사내용만 다르다는 것을 표현한다. 즉 가사를 제외하고 코드 진행으로만 본다면 패턴A의 8마디 혹은 4마디가 다시 한번 반복된다. (마디 수는 변동 가능하다.)

그리고 바로 클라이막스라고 표현되는 패턴C 부분이 나오는 곡이 있는가하면 패턴B와 패턴B′를 거쳐서 패턴C로 이동하는 경우도 있다. 위 코드나열 패턴은 많이 사용되고 있는 패턴을 표현한 것이지 항상 대중음악이 저런 패턴으로 작곡되어지는 것은 아니다.

여러 코드나열 패턴에 대해서는 얼마든지 수정이 가능하다. 본 논문에서 구현한 코드작곡 프로그램의 경우 그림2의 패턴으로 구현되었기에 이후 설명에서도 위와 같은 패턴을 기준으로 설명하도록 한다.

II-3 코드기반 작곡을 위한 쿼드트리 표현

먼저 그림3의 Intro(전주) 부분과 Outro(후주), 그리고 간주부분은 제외하고 패턴A, 패턴A′, 패턴B, 패턴B′, 패턴C, 패턴C′ 부분만을 본다면 앞에서 보여준 코드진행도식 이라는 그림을 이용하여 어떠한 진행공식을 만들 수 있다. 예를 들어 C 라는 코드가 첫 코드로 주어졌을 때 그림2 에서 보여준 트리 구조를 통해서 나올 수 있는 4가지의 코드가 G, F, Em, Am 라는 걸 알 수 있다. 컴퓨터는 그 4가지의 어색하지 않은 코드 진행중 하나를 랜덤 하게 택하게 된다. 그리고 랜덤 하게 골라진 두 번째 코드에서 선택 가능한 4가지의 코드들 중 하나가 다시 선택되어지고 그런 식으로 4마디가 만들어진다.(각 코드나열 패턴이 4마디씩이라는 전제하에) 물론 이때의 경우의 수는 64가지이다. 즉 64가지의 4마디 어색하지 않은 코드진행 패턴이 만들어지는 것이다. 아래에 그림3은 첫 코드가 C로 주어졌다는 가정 하에 만들어질 수 있는 수많은 4마디의 경로 중 하나의 경로를 표현한 그림이다.

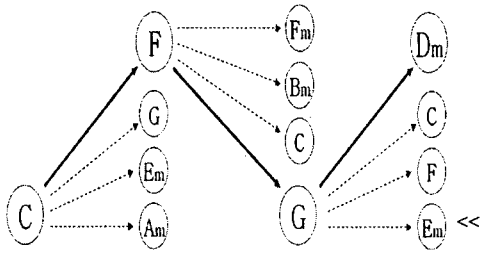
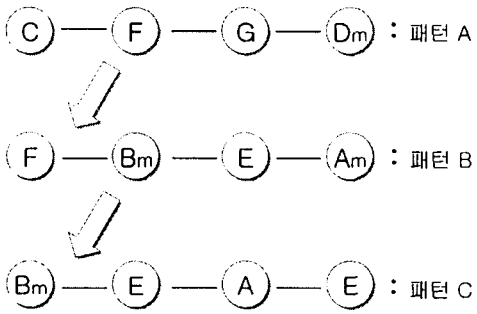


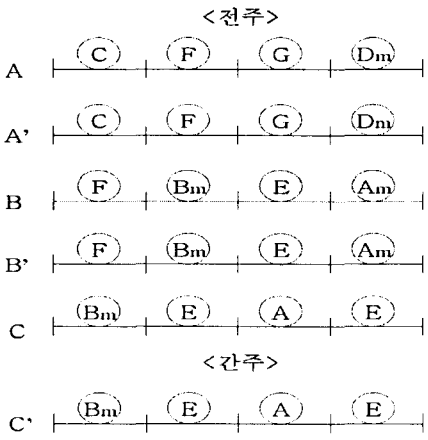
그림 3. C코드 시작 시 트리경로 >

그림2 에서보인 대중음악의 패턴생성과 같이 패턴 A′를 만들기 위해 우리는 그 만들어진 4마디의 패턴을 다시 한번 반복시킨다. 작곡을 하는 사람은 첫 번째 소스코드를 부여한다. 그 이후로는 컴퓨터가 여러 코드들의 진행패턴을 만드는 것이다. 그렇다면 첫 번째 소스코드를 부여하여 첫 패턴A 4마디가 나왔다고 하다면 두 번째 패턴B 나 혹은 패턴C 를 만들기 위해서는 컴퓨터가 어떤 소스코드를 패턴B 나 패턴C 의 시작으로 할 것인가의 문제가 생긴다. 이의 해결책으로는 많은 방법이 있지만 본인은 A패턴에서 사용되어진 4가지의 코드들 중에 두 번째 쓰인 코드를 B패턴이나 C패턴의 시작으로 하는 방법을 택하였다. 즉 예를 들어서 컴퓨터가 선택한 4마디의 코드진행이 앞의 그림4에서와 같이 C-F-G-Dm 라면 이 4개의 코드 중에서 두 번째로 나온 F 코드를 B패턴 혹은 C패턴의 시작으로 하는 것이다. B패턴과 C패턴이 둘 다 존재하는 곡을 작곡한다면 C패턴을 위해서는 B패턴의 4마디 코드 중 두 번째 코드를 C패턴의 시작점으로 부여한다. (그림.4)

이로서 코드작곡의 전주, 간주, 후주를 제외한 모든 작곡이 끝난다. 본인이 조사한 바에 의하면 전주의 첫 코드는 A패턴에서 처음으로 나오는 코드와 같은 확률이 매우 높다. 그러므로 전주를 위해서 A패턴의 시작과 같은 코드를 부여하고 A패턴과 같지 않은 4마디를 만든다. (ex. C-G-Em-Am) 그리고 간주를 위해서 간주가 나오기 전에 끝난 코드를 시작으로 하는 4마디(마디의 수는 본 논문의 단순한 예를 위한 것이므로 때에 따라서 수정을 요한다. 마디의 수가 늘어난다 해도 어색한 코드진행의 경로를 따라가는 방법에 전혀 문제는 없다.)를 만든다. (그림.6은 간주 전 E 코드를 소스로 부여한 4마디의 간주를 만드는 예이다.)

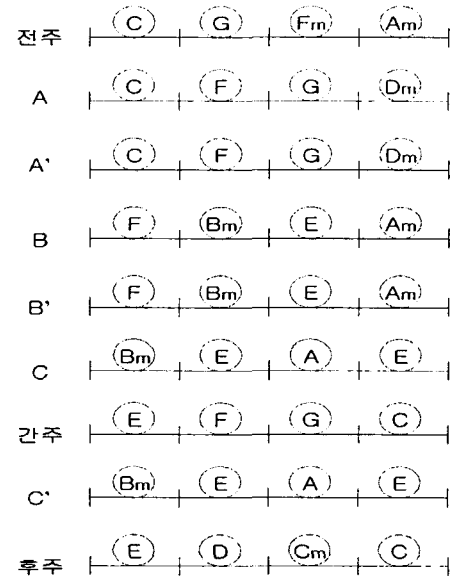


<그림 4. 새로운 패턴 첫 소스 부여방법>



<그림 5. 작곡 완성 본>

마지막 후주는 그 곡의 마지막을 장식하는 부분으로서 후주에서 가장 마지막에 나오는 코드는 그 곡의 처음 시작 코드와 같을 확률이 매우 높으므로 여기서는 곡의 처음 시작에 들어갔던 코드로(즉, 여기서는 C코드) 끝나는 4마디를 만든다. (ex. E-D-Gm-C)이것은 방금 전에도 언급했듯이 거의 대부분의 대중가요가 처음 시작하는 코드와 마지막에 끝나는 코드가 같다는 점을 고려한 것이다. 작곡 프로그램 상에서는 마지막 코드가 첫 코드와 일치 할때까지 4마디의 코드진행이 계속 반복되고 마지막 코드가 처음 코드와 일치하는 순간의 4마디를 마지막 후주로 선택하게 프로그램 되어있다.

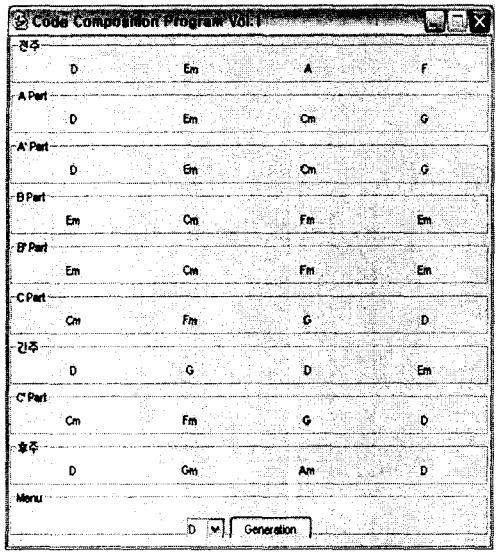


<그림 6. 최종 작곡 완성본>

```
class TimeListener implements ActionListener{
  int md;
  int count = 1;

  public void actionPerformed(ActionEvent ae){
    while((md = ((int)(Math.random()*400))%4) == 0){
      ;
    }

    for(int i=0; i<melodyList.length; i++){
      if(lastCode.equals(melodyList[i][0])){
        lastCode = melodyList[i][md];
        break;
      }
    }

    if(count == 12 || count ==20){
      lastCode = syllable[count-3].getText();
    }

    if((count >=8 && count <= 11) || (count >=16 && count <= 19) || (count ==4)){
      lastCode = syllable[count-4].getText();
    }else if(count == 24 || count == 32){
      lastCode = syllable[count-1].getText();
    }else if(count >=28 && count <= 31){
      lastCode = syllable[count-8].getText();
    }

    if(count == 35 && lastCode.equals(syllable[0].getText())){
      timer.stop();
    }else if(count == 35 && !lastCode.equals(syllable[0].getText())){
      count = 32;
      syllable[32].setText("~~~");
      syllable[33].setText("~~~");
      syllable[34].setText("~~~");
      syllable[35].setText("~~~");

      return;
    }

    System.out.println(count + " +++ "+ md);
    syllable[count].setText(lastCode);
    count++;
  }
```

<그림 7. 코드작곡 프로그램 예>

그림8은 자바로 코드생성 프로그램을 구현한 화면이다. Generation 버튼을 클릭하여 첫 소스코드를 부여할수 있으며(그림8은 처음 소스코드를



<그림 8. 자바로 구현한 코드생성 프로그램>

D코드로 부여한 그림) 소스코드 부여후에는 앞에서 설명한 이론대로 소스작곡이 되어진다. 서론부 에서도 언급했듯이 음악을 체계적으로 배워오지 않은 사람들도 어색하지 않은 작곡이 가능한 건 어려서부터 자신들도 모르게 들어온 여러 음악들의 영향 때문일 것이다. 그러므로 그런 무의식중에 받아온 교육을 기반으로 하는 대부분의 사람들에게 있어서 코드작곡이 되어있는, 즉 적어도 바탕이 그려져 있는 곳에 자신들이 표현하고 싶은 멜로디를 입히는 것은 누구나 가능할 것이라고 생각된다. 적어도 무에서 유를 창조하는 것보다는 코드작곡이 되어있는 곳에 멜로디를 입히는 것이 훨씬 더 수월한 일 일 것이다. 그리고 그들이 어색한 음악을 가려낼 줄 알 듯이 그들 역시 베이스가 완성되어 있는 상황에서의 어색하지 않은 멜로디의 창조는 얼마든지 가능하리라 본다.

III. 결과 및 고찰

본인은 상기 논문을 통하여 코드작곡에 한해서 사람이 아닌 컴퓨터가 작곡을 대신해주는 프로그램을 구현하였다. 결과로 수많은 코드진행패턴이 나올 수도 있겠지만 계이름조차 모르는 사람들의 경우에는 그 결과물로 나와있는 코드진행패턴을 봐도 그게 어떤 화음으로 어떻게 연주되는지, 어떤 음으로 귀에 들리는지 조차 모르는 경우가 많을 것이다. 그런 사람들은 그 코스진행패턴을 본다해도 작곡에 전혀 도움이 되지 않을 것임은 분명하다. 그러므로 결과물로 나온 코드진행패턴을 미디와 연동하여 비트와 장르만 부여해 준다면 컴퓨터가 스스로 연주해줄 수 있다면 훨씬 더 많은 사람들이 이 프로그램을 통하여 작곡을 할 수 있을 것이다.

본 논문에서 구현한 코드작곡 프로그램은 한 코드 당 어색하지 않은 진행을 4가지씩 으로만 예로 들었을 경우와 다른 여타코드를 배제한 메이저와 마이너 코드만을 사용한다는것을 전제로 구현하였다. 하지만 좀 더 많은 자료를 바탕으로 어색하지 않은 모든 코드진행패턴을 찾아내고 그것을 바탕으로 메이저나 마이너코드 뿐만 아니라 기존에 사용되는 모든 코드들을 부여해서 좀 더 완벽에 가까운 작곡 프로그램을 만들어내는 것이 앞으로의 연구과제가 될것이다.

참고 문헌

(1) 도용태 김일곤 김종환 박창현 공저, 인공지능 개념 및 응용, 사이텍미디어, 2001. 16page
(2) 유동선 이교원 공저, 기초 퍼지 이론, 교 우사 2001.9, 188page
(3) Neapolitan Naimipour, ' Fundation of Algorithms', 1998
(4) Triton, 'Music workstation /sampler-Voice Name List',KORG
(5) Rack ATTACK User's manual, waldorf, 2002,
(6) 1604-VLZ PRO 16-channel owner's Manual, 2000
(7) PC2 Kurzweil Musician's Guide, 2002