Regular paper | Journal of the KIECS. pp. 687-692, vol. 18, no. 4, Aug. 31. 2023, t. 120, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2023.18.4.687

양자의 중첩 특성을 이용한 소리의 생성 및 제어에 대한 연구

조민호*

A Study on The Create and Control of Sound using The Quantum Superposition Characteristics

Min-Ho Cho*

요 약

이번 연구는 양자 컴퓨터가 가지는 중첩의 특성을 이용하여 음악을 만들어 보고자 하는 의도에서 시작하였다. 기존의 음악은 작곡가가 작곡한 것에 제한되는 특징을 가진다. 하지만, 양자 컴퓨터의 중첩을 이용한 음악은 제한된 범위 내에서 실행 시에 변화되는 음악적 특징을 가진다. 이를 이용하여, 실행 시에 특정 화음을 기준으로 변화하는 음악을 만들 수 있을 것이다. 이번 논문에서는 양자 컴퓨터와 기존 컴퓨터를 연결하여 소리를 발생시킨다, 그리고 중첩의 성질을 적용하여 변화하는 음을 만들어내는 것에 중점을 둔다.

ABSTRACT

This research began with the intention to create music using the superposition characteristics of quantum computers. Existing music has characteristics that are limited to those composed by composers. However, music using the overlap of quantum computers has musical characteristics that change when executed within a limited range. Using this, you will be able to create music that changes based on specific chords at run time. In this paper, quantum computers and existing computers are connected to generate sound, And it focuses on creating changing sounds by applying the nature of superposition.

키워드

Quantum Computing, Superposition, Midi, Music, Qiskit, Python
양자컴퓨팅, 중첩, 미디, 음악, Qiskit, 파이썬

I. 양자 음악 제작의 배경

인공지능의 발전은 점점 다양한 분야에 적용되고 있고, 특히 딥러닝은 번역, 채팅 등 많은 분야에서 사용되고 있다[1~9]. 하지만, 사용이 늘어나면서 지금보다 빠른 컴퓨팅 성능을 요구하고 있고, 어떤 경우에는 컴퓨팅 성능의 문제로 구현하지 못하는 경우가 발생하고 있다. 양자 컴퓨터는 특정 분야에서 기존 컴퓨터의 한계를 뛰어넘는 성능을 보여줌으로써 인공지능을 포함한 기존 환경이 구현하지 못하는 분야의 해결 방

* 교신저자 : 중원대학교 컴퓨터공학과
• 접 수 일 : 2023. 06. 26
• 수정완료일 : 2023. 07. 19
• 게재확정일 : 2023. 08. 17
• Received : Jun. 26, 2023, Revised : Jul. 19, 2023, Accepted : Aug. 17, 2023
• Corresponding Author : Min-Ho Cho
Dept. Computer System Engineering, JungWon University,
Email : chominhokr@jwu.ac.kr

안을 제시할 수 있을 것으로 주목받고 있다[10]. 하지만, 양자 컴퓨터가 가지는 특성을 잘 활용하는 애플리케이션의 부족과 하드웨어 환경의 관리 어려움으로 인하여 확산에 한계가 있다[11]. 이번 연구에서는 양자 컴퓨터를 이용하는 애플리케이션의 부족을 해결하고, 새로운 응용 분야를 개척하고자 한다.

연구의 주제는 양자 컴퓨터를 이용한 음악의 생성이다. 양자 컴퓨터는 실행 시에 특정 상태가 결정되는 중첩의 특징을 가진다. 이를 이용하여, 음악을 제작하게 되면, 새로운 음악의 창조가 가능할 것으로 생각한다. 예를 들어, 제한된 범위에서 실행 시에 약간씩 변화되는 음악의 제작이 가능하다, 추가로 저작권이나 다른 분야에 대한 제한 없이 음악을 즐기고, 배포할 수 있는 길이 열리게 될 수도 있다.

현재 시점에서 양자 컴퓨터는 음악에 관련된 기능을 지원하지 않는다. 그러므로 음악을 만들기 위해서는, 기존 컴퓨터에서 제공하는 음악의 정의 및 연주 기능을 이용하는 것이 필요하다. 그리고, 음에 관여한 코드를 생성하는 부분은 양자 프로그램을 이용하여 제작한다. 이 두 가지 프로그램을 합쳐서 최종 프로그램이 완성된다. 구체적으로 기존 컴퓨터에서 제공하는 음악 관련 표준인 MIDI를 사용하고, 프로그램 언어로는 파이선을 사용한다[12]. 그리고, 양자컴퓨팅을 적용하기 위하여 Qiskit을 사용하여 양자컴퓨팅을 시뮬레이션하고, 양자 컴퓨터에서 실행하여 결과는 그림 1에서 확인한다.

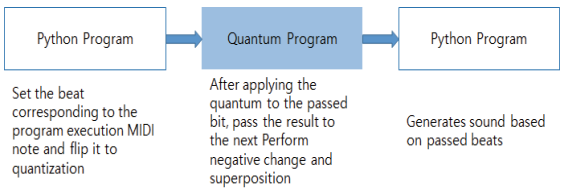

그림 1. 시스템 구현의 개념도

Fig. 1 Conceptual diagram of system implementation

이번 연구에서는 MIDI를 사용한다. MIDI는 디지털 음악 장치와 정보를 주고받기 위한 표준 명세이다. MIDI 메시지는 상태바이트와 데이터바이트로 구성되며 상태바이트는 0으로 시작하고, 데이터바이트는 1로 시작하며 7바이트로 구성된다 그러므로 128개의 다른 값을 정의할 수 있다. MIDI는 3개의 바이트로 구성되며, 각각 장치의 식별, 음의 높이, 음의 속도를 정의한다.

이번 연구의 수행을 위하여 장치의 식별과 음의 속도는 고정하고, 음의 높이만을 변경하는 것으로 제한하였다. 향후, 음의 속도 부분은 추가로 고려할 예정이며, 장치의 식별에 대한 부분도 향후 활용의 범위에 따라 적용할 수 있다.

MIDI는 음을 A, B, C, D, F, G로 표현하며 각 음 사이는 #을 이용하여 표현한다. 즉, C#은 C와 D 사이의 음을 나타낸다.

이제 이러한 배경을 이해하였다면, 본격적인 설명을 진행하자. MIDI에서 '01111100'은 C4 음을 나타낸다. 즉, 모든 음은 7비트로 표현할 수 있으며, 비트로 표현된 정보가 전달되면, 표준 규약에 의거 특정 소리가 발생하는 것이다.

이번 논문의 아이디어는 MIDI로 표현된 음의 비트 모양이 단지 한 비트의 변화만으로 다른 음이 나오게 된다는 것이다. 즉, A4음은 '1000101'이고, F4음은 '1000001'이다. 그러므로 3번째 비트만 변화하면 A4음과 F4음을 표현할 수 있다. 그래서, 기존의 MIDI 비트 중, 특정 음을 나타내는 비트를 중첩으로 설정하고, 실행 시에 값이 결정되도록 함으로써 실행할 때마다 다른 음을 발생시키는 프로그램의 제작이 가능하다는 것이다.

이런 개념을 확장하면, MIDI가 표현하는 비트를 조화있게 중첩 기능을 적용함으로써, 특정 범위를 벗어나지 않는 음의 생성이 가능해진다. 여기에 음의 속도 부분을 추가하여 고려한다면, 일반적인 수준의 음악을 제작하는 것도 가능할 것으로 판단된다. 당연히 장비에 대한 기능을 추가하면 컴퓨터 외에 다양한 장비에서도 작동하는 음악을 만들 수 있다.

이번 논문에서는 하고자 하는 것은 음의 높이만을 고려하여 MIDI의 음에 대한 비트 모양을 분석하고, 화음을 이룰 수 있는 비트를 찾아내어 이들에게 중첩의 성질을 적용하는 것이다.

처음에는 한 개의 비트만 다른 것을 이용하여 소리를 발생시켜보고, 이어서 두 개, 세 개가 다른 것을 연결하여 소리와 화음을 발생하는 것으로 확장할 수 있다. 이 과정을 반복하여 음의 높이에 대한 비트 변화의 고려가 완료되었다면, 음의 속도를 나타내는 비트도 같은 방식으로 분류하고 조정한 후에 중첩을 적

용하여 음을 발생하게 할 수 있다. 이런 과정이 적절하게 조화된다면 기존과는 다른 형태의 음악 제조가 가능할 것이다.

이번 연구는 이런 과정의 시작에 해당한다. MIDI에서 음의 높이만을 고려한다. 그리고 한, 두 개의 비트를 조작하여 음을 변화시키는 프로그램을 제작한다. 물론 제작된 프로그램은 여러 비트를 변화시켜서 음을 발생시키도록 확장할 수 있다.

II. 음악 프로그램의 제작

음악 프로그램의 제작은 두 개의 단계로 나뉘게 된다. 첫 번째 부분은 파이선에 MIDI를 적용하여 소리를 발생시키는 부분이고, 두 번째 부분은 MIDI를 적용할 비트 조합을 생성하기 위하여 양자 컴퓨터를 적용하는 것이다.

이번 연구에서 양자 컴퓨터의 적용은 두 가지 단계로 나누었다. 첫 번째 단계는 파이선에서 사용할 수 있는 양자 컴퓨터 시뮬레이션 라이브러리인 Qiskit을 사용하는 것이고, 두 번째 단계는 이것을 사용하여 성공적으로 개발된 양자 프로그램을 IBM의 실제 양자 컴퓨터에서 수행하여 결과를 확인하는 것이다. 양자 컴퓨터를 이용하는 프로그램의 제작은 처음부터 IBM이나 구글의 양자 컴퓨터를 이용하는 것보다는 파이선에 Qiskit을 이용하는 것이 쉽고 편리하다.

첫 번째로 파이선으로 음을 합성하는 함수를 만든다. 기본적으로 음은 주파수와 크기를 가진 파동이다. 그러므로 그림 2의 소스를 이용하여 소리를 발생시킬 수 있다.

그림 2의 소스는 MIDI를 초기화하고, 주어진 빈도와 높이 데이터를 기반으로 소리를 생성하고, 이것을 스피커를 통해 사용자에게 제공한다. 상세한 내용은 파이선에서 제공하는 pygame을 참고한다.

```
import numpy
import pygame, pygame.sndarray
import pickle

def sound_gen(freqs,volumes):
  pygame.mixer.init()

  sound_wave=sum([numpy.resize(volume*16384*numpy.
                sin(numpy.arange(int(44100/float(hz)))
                *numpy.pi*2/(44100/float(hz))),(44100,)).
                astype(numpy.int16) for hz,volume in zip(freqs,volumes)])
  stereo = numpy.vstack((sound_wave, sound_wave)).T.copy(order='C')
  sound = pygame.sndarray.make_sound(stereo)
  sound.play(-1)
  pygame.time.delay(1200)
  sound.stop()
  pygame.time.delay(1200)
```

그림 2. 소리를 발생시키는 함수

Fig. 2 Function that produces sound

다음 단계는 MIDI에서 두 음의 중첩을 만들기 위하여, MIDI 코드가 1비트 차이가 나는 두 개의 노드를 찾고, 1비트의 차이에 중첩을 적용하기 위하여 해당되는 비트에 아다마드(Hadamard) 게이트를 적용하는 것이다. 이때, 나머지 비트는 고정된 값을 가진다. 물론, 이번 시도가 성공한 이후에는 여러 비트로 프로그램을 확장할 수 있다.

프로그램의 제작을 위하여 선택한 음은 F4와 A4이다. F4는 MIDI 코드에서 65, '1000001'이고, A4는 MIDI 코드에서 69, '1000101'이다. 두 개의 음은 3번째 비트만 다르며, 이번 연구에 적용하기에 적합하다. 실제 MIDI의 음에 대한 비트의 모양을 분석해 보면 앞에서 언급한 F4, A4 외에도 유사한 특징을 가지는 음이 다수 있다는 것을 확인할 수 있다. 다만, 이 음들이 서로 조화를 이루는 음의 여부는 이번에 고려하지 않는다.

그림 3에 F4와 A4의 중첩을 위한 파이썬 프로그램이 있다. 이 프로그램은 파이선에 양자 컴퓨터 시뮬레이션 기능을 제공하는 Qiskit을 사용하여 제작하였다.

그림 3 프로그램은 양자 회로를 만들고, 양자 회로에 F4와 A4의 비트 중, 다른 값을 가지는 3번째 항목에 아다마르 게이트를 적용한 것이다.

적용한 결과 quantum_play_sound 함수를 통해서 그림 1의 sound_gen 함수를 불러서 소리를 발생시킨다. 두 번째는 발생한 소리를 모아서 화음이 된 소리를 발생시키는 것을 quantum_play_chorus 함수에서 수행하게 된다. 그림 3을 수행하면, 컴퓨터에서 특정 소리가 수행할 때마다 변화되어 출력되며, 실제 수행하는 과정에 print 명령어를 넣어서 출력을 확인한 것이 그림 4에 있다.

```
import qiskit
from qiskit import ClassicalRegister, QuantumRegister
from qiskit import QuantumCircuit, execute

# 레지스터와 프로그램의 구성
qr = QuantumRegister(7)
cr = ClassicalRegister(7)
qc = QuantumCircuit(qr, cr)

# F4 and A4 together: F4는 1000001 이다  A4는 1000101이다
qc.x(qr[0]) # 1 at qubit 0
qc.x(qr[6]) # 1 at qubit 6
qc.h(qr[2]) # F4와 A4는 2의 위치만 다르다. 이것이 변한다

for j in range(7):
    qc.measure(qr[j], cr[j])

-----------------------------------------------
# qc가 1000001이다. 매측정마다 둘 중 한개의 소리가 나온다.
quantum_play_sound(qc,4)

# 4번 연속된 측정의 결과를 합쳐서 보여준다. 각각이 50% 확률을 가진다
quantum_play_chorus(qc,4)
```

그림 3. 소리를 발생시키는 양자함수

Fig. 3 Quantum function that produces sound

```
{'1000101': 1}
A4
{'1000001': 1}
F4
{'1000001': 1}
F4
{'1000001': 1}
F4
75.000000 percent F4
25.000000 percent A4
```

그림 4. 수행 결과

Fig. 4 Run result

그림 4를 보면, 프로그램이 제작 의도대로 작동하고 있는 것을 확인할 수 있다. 그림 3의 프로그램이 수행되면서 실시간으로 A4와 F4 음이 무작위로 생성하게 되고, 생성된 결과는 파이선의 MIDI를 통해서 스피커로 전달되게 된다. 그림 4의 결과는 여러 번 수행하면 F4가 75%, A4가 25% 발생하였음을 확인할 수 있고, 이 결과는 매번 수행마다 달라진다.

두 개 음의 변화를 모아서 조화음을 발생시키면 새로운 음이 발생하게 된다. 이때, 만약 우리가 선택한 A4와 F4가 상호 보완적인 음이라면 듣기 좋은 소리가 나오게 된다. 물론, 그림 3에서 제시한 프로그램은 두 개의 음, 그것도 높이만을 다루기 때문에 듣기 좋은 화음은 기대하기 어렵다.

이제 프로그램의 제작이 완료되었고, 실제로 수행하여 소리도 발생시켰다. 다음 단계는, 실제 양자 컴퓨터에서 수행시켜보는 과정이다. 그림 3의 프로그램을 그대로 IBM 양자 컴퓨터에서 수행한 화면이 그림 5에 있다.

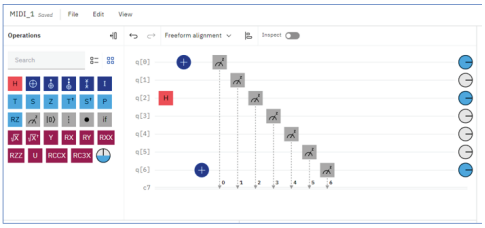

그림 5. IBM 양자 컴퓨터 수행화면

Fig. 5 IBM Quantum function execution screen

그림 5에서 확인할 수 있는 것과 같이, 그림 3에서 개발한 프로그램은 실제 양자 컴퓨터에서도 잘 작동한다. 그러므로, 이번 연구의 결과를 확장하여 더 의미 있는 결과를 얻고자 하는 경우, 이번 연구에서 개발한 코드와 프로그램의 구성을 활용할 수 있다.

이번 연구의 결과를 확장하는 첫 번째 사례는 MIDI의 코드값이 하나만 다른 경우 외에 두 개 이상이 다른 경우로 확장하는 것이다. 이를 위하여 MIDI 코드를 분석해서 2개의 비트가 다른 코드를 식별해서 그림 3의 프로그램을 확장할 수 있다.

그림 6에 MIDI 코드를 분석하여 2개가 다른 코드를 조사한 예를 제시한다. 이해를 위하여 이미 설명한 F4, A4도 추가하였다. 그림 6을 보면 2개의 비트가 다른 것이므로 MIDI의 음 중에서 4개의 음이 선택되었다. 지금의 시점에서는 4개 음의 조화에 대한 것보다는 음의 선택과 생성에 관심이 있으므로 이것을 이용하여 그림 3의 프로그램이 2개의 비트에서 중첩을 적용할 수 있도록 개선하였다. 결과는 그림 7에서 확인할 수 있다.

유사하게 MIDI를 구성하는 전체를 대상으로 해당 시점에 특정한 비트를 조작하도록 프로그램을 구성하는 것이 가능하다. 이를 기반으로 음악의 제작이 가능하다는 것을 확인할 수 있다. 이러 개념을 확장하면, 기존 음악의 음계를 구현하는 프로그램의 제작이 가능하다.

```
C3    0 1 1 0 0 0 0
E3    0 1 1 0 1 0 0
G#3   0 1 1 1 0 0 0
C4    0 1 1 1 1 0 0
----------------------
F4    1 0 0 0 0 0 1
A4    1 0 0 0 1 0 1
```

그림 6. 2비트가 다른 MIDI 코드 예
Fig. 6 MIDI code example with 2 different bit

```
import qiskit
from qiskit import ClassicalRegister, QuantumRegister
from qiskit import QuantumCircuit, execute

# set up registers and program
qr = QuantumRegister(7)
cr = ClassicalRegister(7)
qc = QuantumCircuit(qr, cr)

qc.x(qr[4]) # 1 at qubit 4  ==> 0010000
qc.x(qr[5]) # 1 at qubit 5  ==> 0110000 : C3
qc.h(qr[2]) # create a superposition on qubit 2 ==> 0110100 : E3
qc.h(qr[3]) # create a superposition on qubit 3 ==> 0111000 : G#3 :

for j in range(7):
    qc.measure(qr[j], cr[j])

quantum_play_notes(qc,4)
quantum_play_chords(qc,45) # 4 가지 음이 섞여서 출력된다.
```

그림 7. 2비트가 다른 MIDI 코드 프로그램
Fig. 7 MIDI code program with 2 different bit

그림 7의 코드를 살펴보면, 그림 3에서 제시한 소스와 크게 다르지 않다. 즉, 기존에 개발했던 코드를 더 많고 다양한 기능을 제공하는 코드로 변경하는 것이 어렵지 않다는 점을 확인할 수 있다.

그림 7에서 제시한 프로그램의 결과를 그림 8]에 제시하였다.

```
E3
G#3
C3
C4
22.222222 percent E3
26.666667 percent C4
8.888889 percent G#3
42.222222 percent C3
```

그림 8. 수행 결과
Fig. 8 Run result

그림 8을 보면 우리가 예상한 것과 같이 4개의 음이 임의로 선정되어 출력되는 것을 확인할 수 있다. 이 음을 묶어서 처리하게 되면 좋은 화음을 기대할 수 있을 것이다.

이제 두 개의 비트가 다른 경우를 실제 양자 컴퓨터에서 수행하고자 하면, 그림 5가 1개가 다른 경우에 1개의 아다마스 게이트를 사용했으므로, 두 개의 비트가 다른 경우에는 2개의 아다마스 게이트를 사용하면 될 것이다. 실제 수행하여 보면, 잘 수행되는 것을 확인할 수 있다. 그리고, 좀 더 많은 아다마스 게이트를 사용해서 다양한 화음을 생성할 수 있을 것이다.

III. 결론 및 향후 연구 방향

이번 연구는 양자 컴퓨터의 응용 사례로 음악을 선택하여 진행하였다. 특히, 양자 컴퓨터의 특징인 중첩 기능을 이용하여 실행 시에 변화하는 음악을 구현하고자 했다.

이를 위하여 고려할 사항은 음을 구성하는 높이 와 속도인데, 구현의 편리성을 위하여 속도를 고정하고 음의 높이만을 조절하는 방향으로 개발을 진행하였다. 개발의 방법은 기존 컴퓨터에서 사용하는 MIDI 표준을 파이선으로 구현하였고, MIDI를 구성하는 비트의 조절을 위하여 양자 컴퓨터를 사용하였다.

비트의 조절은 처음에는 한 개의 비트만 다른 경우를 고려하여 파이선과 Qiskit을 이용하여 프로그램을 제작하였고, 제작된 프로그램은 IBM 양자컴퓨터에서 수행되는 것을 확인하였다. 이제 확인된 결과를 기반으로 MIDI의 코드 중에서 2개 이상의 비트가 다른 경우를 대상으로 작업을 확대하였다.

이번 연구를 통하여 멋진 음악은 제작하지 못했으나, MIDI를 기반으로 양자 컴퓨터와 연계하여 중첩의 특징을 활용한 다양한 소리를 얻는 방법을 개발하였다. 이제 이 방법을 기반으로 MIDI의 다양한 코드 간 연계를 통해 유용한 음악을 생성하는 방법을 연구할 예정이다. 그리고 비트의 조작에 의한 음의 조화로운 발생을 만들 수 있게 되면 음을 구성하는 속도를 추가로 고려하여, 현재 우리가 익숙한 음악을 양자 컴퓨터를 이용하여 구현할 계획이다. 이번 연구가 양자 컴퓨터의 상용화 및 확산에 이바지할 수 있는 초석이 되기를 바란다.

References

[1] C. Kim, S, Choi and K. Kwahk, "Investigation of Research Trends in Information Systems Domain Using Topic Modeling and Time Series Regression Analysis," *Journal of Digital Contests Society,* vol. 18, no. 6, Oct. 2017, pp. 1143-1150.

[2] Y. Guo, Y. Liu, A. oerlemans, S. Lao, S. Wu and M. S. Lew, "Deep Learning for visual understanding : A review," *Neurocomputing*, vol. 187, Apr. 2016, pp. 27-48.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senoir, V. Vanhoucke, P. Nguyen, T. N. Sainath and B. Kingbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, issue. 6, Nov. 2012, pp. 82-97.

[4] M. Cho, "A study on the history, classification and development direction of artificial intelligence," *J. of the Korea Institute of Electronic Communication Sciences,* vol. 16, no. 2, April. 2021, pp. 307-312.

[5] J. Moon and Y. Lee, "Artificial Intelligence Computing Platform Design for Underwater Localization" *J. of the Korea Institute of Electronic Communication Sciences*, vol. 17, no. 1, Feb, 2022, pp. 119-124.

[6] C. Lee and H. Park, "A Comparative Study on the Accuracy of Important Statistical Prediction Technique of Marketing Data," *J. of the Korea Institute of Electronic Communication Sciences*, vol. 16, no. 3, June. 2021, pp. 519-524.

[7] M. Cho, "Comparison of prediction accuracy between regression analysis and deep learning and empirical analysis of the importance of techniques for optimizing deep learning models." *J. of the Korea Institute of Electronic Communication Sciences,* vol. 18, no. 2, Febl. 2022, pp. 299-304.

[8] M. Dahiya, "A Tool of Conversation : Chatbot," *International Journal of Computer Sciences and Engineering,* vol. 5, issue. 5, May 2017, pp. 158-161.

[9] J. Jhang and C. Zong, "Deep Neural Networks in Machine Translation: An Overview," *IEEE Intelligent Systems*, vol. 30, issue. 5, Sept.-Oct. 2015, pp. 16-25.

[10] T. Forcer, T. Hey, D, Ross and P. Smith, "Superposition, Entanglement and Quantum Computing," *Quantum Information & Computation*, vol. 2, no. 2, Feb. 2002, pp. 97-116.

[11] A. Khrennikov, "Roots of quantum computing supermacy:superposition, entanglement, or complementarity?," *The European Physical Journal Special Topicss*, vol. 230, 2021, pp. 1053-1057.

[12] E. Feldman, "MIDI music model," *SoutheastCon 2015, Fort Lauderdale, FL, USA*, June, 2015, pp. 1-4, doi: 10.1109/SECON.2015.7133038.

저자 소개

조민호(Min-Ho Cho)

1989년 인하대학교 졸업(공학사)
1989년 ~ 2012년
HP Korea, Openwave, SK C&C에서 개발자, 컨설팅, PM의 역할을 수행

2003년 숭실대학교 대학원 컴퓨터공학과 졸업 (공학박사)
2013년~ 중원대학교 컴퓨터공학과 교수
※ 관심분야 : 인공지능 및 기계학습, 데이터분석, 소프트웨어공학, 데이터 마이닝 및 빅데이터