KIPS Trans. Softw. and Data Eng.
Vol.8, No.2 pp.51~60 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2019.8.2.51

Software Platform for Analyzing Gene and Disease Relevance

Myeong Ho Song† · Soo Dong Kim††

ABSTRACT

While the quality of life is enhanced as many types of diseases are remedied, there is a high demand for analysis and research on gene-related diseases. There exists various forms and requirements in analyzing the relevance between genes and diseases, and the runtime efficiency can be decreased due to the level of algorithm optimization. This paper proposes a platform for analyzing gene disease relevance, provides API for remedying the variability issue, and suggests two algorithms which optimize the runtime efficiency. And, we conduct experiments for measuring the relevancy using the analysis API, and compare the two algorithms. The first algorithm is to improve the runtime efficiency comparing to the conventional methods, and the second algorithm is to improve the runtime efficiency with lower accuracy. This platform can be well utilized for analyzing various forms of gene disease analytics.

Keywords : Analytics, Software Platform, Algorithms, Gene, Optimization

유전자 질병 관련도 분석을 위한 소프트웨어 플랫폼

송 명 호† · 김 수 동††

요 약

많은 질병들이 정복되면서 삶의 질을 향상시키지만, 유전병들은 많은 분석 및 연구가 필요하다. 이러한 질병과 유전자 관련도를 분석할 때 다양한 요구사항이 존재하고, 알고리즘 최적화 유무로 인해 런타임 효율성이 저하된다. 본 논문은 유전자 질병 관련도 분석 플랫폼을 소개하고 위의 이슈를 해결하기 위한 분석 API와 두가지 런타임 효율성 최적화 알고리즘을 제시한다. 그리고 제시한 분석 API를 이용하여 관련도 측정 실험을 진행, 두 최적화 알고리즘의 결과와 비교했다. 첫 번째 알고리즘은 이전 실험과 같은 결과를 적은 시간에 도출했고, 두 번째 알고리즘은 이전 실험들에 비해 낮은 정확도의 결과를 더 적은 시간에 도출했다. 따라서 본 플랫폼을 통해 여러 방식의 유전자와 질병 관련도를 효율적으로 얻을 수 있다.

키워드 : 분석, 소프트웨어 플랫폼, 알고리즘, 유전자, 최적화

1. 서 론

생명 공학의 발전 등을 통해 많은 질병들이 정복되면서 삶의 질이 향상되고 있지만 아직 해결책을 찾지 못한 질병 또한 많다. 그 중 유전자 간의 관계나 특정 유전자의 유무, 특정 유전자의 변이 등 여러 요인에 의해 발현되는 유전병이 큰 비율을 차지한다[1-5].

현재 유전자의 기능에 대한 연구[6-9]나 특정 유전자를 통한 질병을 예측하는 연구와 메소드(Method) 개발은 많이 진행되고 있다[10-16]. 그러나 다수의 유전자와 질병 간의 관계를 찾는 연구와 메소드 개발은 비교적 적은 편이다[17-19]. 또한 유전자와 질병에 대한 데이터는 방대하기 때문에 유전자 간의 질병 관련도 연산 시 런타임(Runtime)은 증가하게 된다. 이 때 발현 정도를 연산하는 알고리즘의 최적화 유무에 따라 런타임 효율성이 저하되는 이슈가 발생한다.

본 논문에서는 관련도의 관계를 그래프 모델로 표현 및 제안한 후, 분석 API에 대한 설계 및 메소드를 제안한다. 그리고 유전자 간의 질병 관련도 측정 시 발생하는 런타임(Runtime) 효율성 저하 이슈에 대한 해결법을 제시한다.

2. 관련 연구

Ha의 연구에서는 miRNA와 PPI, 질병정보를 이용하여 데이터 통합모델을 만드는 연구를 진행하였다[20, 21]. Kim의 연구에서는 특정 질병에 대한 유전자를 추론하고 유전자 네트워크를 만드는 연구를 진행하였다[22]. 이 연구들은 유전형질

† 준 회 원 : 숭실대학교 컴퓨터학과 석사과정
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
Manuscript Received : July 13, 2018
First Revision : September 27, 2018
Accepted : October 1, 2018
* Corresponding Author : Soo Dong Kim(sdkim777@gmail.com)

또는 유전자와 질병 간 관계를 통해 데이터 모델을 만드는 연구를 진행했다.

Choi의 연구에서는 유전자의 상호작용 정보와 특정 기법을 이용하여 질병을 진단하였다[23]. Kai의 연구에서는 카신-벡 병(Kashin-beck disease)와 비슷한 질병들에 대한 유전자 진단 애플리케이션을 개발하였다[24]. 이 연구들은 유전자와 특정 질병 간의 상호 작용을 통해 질병이나 유전자를 진단하는 방법을 제안했다.

Maji의 연구는 유전자와 마이크로 어레이 데이터를 이용하여 유전자 중요도를 계산하고, 특정 네트워크를 통해 유전자 간의 유사성을 측정하여 질병 관련 유전자의 내부 식별을 진행하였다[25]. Choi의 연구에서는 유전자 네트워크가 특정 환경일 때 유발되는 질병에 대한 바이오 빅데이터를 제안 및 분석을 하였다[26]. Wu의 연구에서는 다중 유전자 네트워크에 대해 질병 관련 우선순위를 정하는 연구를 하였다[27]. 이 연구들은 질병과 유전자에 대해 분석을 하는 방법을 제안했다.

Choi의 연구는 병렬 프로세서를 이용하여 질병에 대한 유전자 발현 데이터 분석을 진행하였다[28]. Lim의 연구는 분류에 대한 런타임 효율성을 높이기 위해 질병 분류에 불필요한 유전자를 제거하는 기법에 대해 연구를 진행하였다[29]. 이 연구들은 질병 분류 또는 질병 발현 유전자 데이터를 분석 시 런타임 효율성을 높이기 위한 방법에 대해 연구를 진행했다.

위의 선행연구들은 특정 유전자와 특정 질병 간의 통합 모델을 통한 분석 방법을 제안하였고 알고리즘 내 런타임 효율성 증가 방법을 제시하는 부분이 부족하다.

3. 유전자 질병 분석 분류

본 장에서는 플랫폼에서 사용하는 유전자와 질병, 둘 사이의 관련도에 대해 설명한다. 또한 유전자와 질병 사이의 관계에 대해 다양한 분석 요구 사항을 해결하는 분석 방식에 대한 분석 분류체계를 설명한다.

3.1 유전자 질병 데이터 모델

본 플랫폼에서 분석에 사용하는 유전자(Gene)와 질병(Disease), 관련도(Relation)에 대한 객체와 이 객체들의 관계는 Fig. 1로 표현한다.

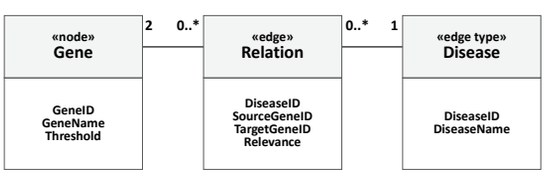

Fig. 1. Gene, Disease and Relation Object Structure

질병 객체는 해당 질병 ID(DiseaseID)와 질병 이름(DiseaseName) 정보를 갖는다.

유전자 객체는 해당 유전자 ID(GeneID)와 유전자 이름(GeneName), 유전자가 질병을 발현시키기 위한 임계값(Threshold)을 갖는다. 임계값의 사용 유무에 따라 플랫폼의 한 분석 방법을 두가지 방법으로 제공한다.

관련도 객체는 유전자 객체와 질병 객체 사이의 관계를 나타낸다. 관계를 맺은 데이터들 중 한 질병에 대해 두 유전자가 질병을 발현시키는 관련도를 나타낸다. 관련도 객체는 한 질병의 ID(DiseaseID), 원인 유전자 ID(SourceGeneID)와 결과 유전자 ID(TargetGeneID), 그리고 두 유전자가 해당 질병의 발현 정도(Relevance)로 구성된다.

3.2 유전자 질병 분석 분류체계

유전자와 질병 데이터를 이용하여 분석할 때 기준이 되는 데이터, 알고 싶은 데이터와 그의 개수 등 많은 요구사항이 발생한다. Table 1은 이러한 요구 사항들을 만족하기 위해 본 플랫폼에서 제공하는 메소드들이다.

Table 1. The Platform Sectors and Methods

Sector	Method
Finding Genes for given Diseases	findRelatedGenes_SD()
	findRelatedGenes_MD()
Finding Diseases for given Genes	findRelatedDiseases_SG()
	findRelatedDiseases_MG()
Finding Genes for given relation	findRelatedGenes()
Finding Diseases for given relation	findRelatedDiseases()

Table 1의 메소드들은 유전자와 질병, 질병과 관련도, 유전자 또는 질병과 관련도 사이의 관계 총 4가지 부문에 대해 분석 방식을 제공한다.

1) Finding Genes for given Diseases

해당 부문의 분석 방식은 구하고자 하는 질병들에 대해 가장 관련도가 높은 유전자 집합을 구한다. 이 분야는 Table 2와 같이 findRelatedGenes_SD() 메소드와 findRelatedGenes_MD() 메소드를 제공한다.

Table 2. Finding Genes for given Diseases Methods

# of diseases	Theshold	Method
1	not use	findRelatedGenes_SD()
	use	findRelatedGenes_SD_TH()
many	not use	findRelatedGenes_MD()
	use	findRelatedGenes_MD_TH()

두 메소드는 각각 x, disease와 x, diseaseSet을 파라미터로 갖는다. x는 찾고자 하는 가장 관련도가 높은 유전자의 개수를 나타낸다. disease는 기준으로 쓰고자 하는 질병을, diseaseSet은 기준으로 쓰고자 하는 질병들의 집합을 나타낸다. 두 메소드를 통해 해당 질병 또는 질병 집합에 대해 가장 높은 관련도를 갖는

x개의 유전자들을 구한다. 또한 findRelatedGenes_SD_TH() 메소드와 findRelatedGenes_MD _TH() 메소드로 임계값을 사용한 방식을 제공한다.

2) Finding Diseases for given Genes

해당 분야의 분석 방식은 구하고자 하는 유전자들에 대해 가장 관련도가 높은 질병 집합을 구한다. 이 분야는 Table 3과 같이 findRelatedDiseases_SG() 메소드와 findRelatedDiseases_MG() 메소드를 제공한다.

Table 3. Finding Diseases for given Genes Methods

# of genes	Theshold	Method
1	not use	findRelatedDiseases _SG()
	use	findRelatedDiseases _SG_TH()
many	not use	findRelatedDiseases_MG()
	use	findRelatedDiseases_MG_TH()

두 메소드는 각각 x, gene과 x, geneSet을 파라미터로 갖는다. gene은 기준으로 쓰고자 하는 한 유전자를, geneSet은 기준으로 쓰고자 하는 유전자 집합을 나타낸다. 두 메소드를 통해 기준으로 사용되는 유전자 또는 유전자 집합이 가장 높은 관련도를 나타내는 x개의 질병을 구한다. 또한 findRelatedDiseases_SG_TH() 메소드와 findRelated Diseases_MG_TH() 메소드로 임계값을 사용하여 결과를 얻는 방식을 제공한다.

3) Finding Genes for given relation

해당 분야의 분석 방식은 모든 관련도 데이터에서 가장 관련도가 높은 유전자를 구한다. 이 분야는 find RelatedGenes() 메소드를 제공한다. 이 메소드는 x를 파라미터로 갖고, 가장 관련도가 높은 x개의 유전자를 구한다. 또한 findRelatedGenes_TH() 메소드로 임계값을 사용한 방식을 제공한다.

4) Finding Diseases for given relation

해당 분야의 분석 방식은 전체 관련도 데이터에서 가장 관련도가 높은 질병을 구한다. 이 분야는 findRel atedDiseases() 메소드를 제공한다. 이 메소드는 x를 파라미터로 갖고, 가장 관련도가 높은 x개의 질병을 구한다. 또한 findRelatedDiseases_TH() 메소드로 임계값을 사용한 방식을 제공한다.

제시한 분석 방식들을 통해 사용자들의 다양한 분석 방식에 대한 요구 사항을 충족시킬 수 있다.

4. 유전자 질병 관련도 확장 그래프 모델 설계

본 장에서는 그래프에서의 노드(Node)와 에지(Edge) 관계를 이용하여 유전자와 질병 그리고 관련도를 표현하는 방식을 제안한다. 본 플랫폼에서 사용하는 그래프 모델은 각 유전자를 노드(Node), 두 유전자 사이의 한 질병에 대한 관련도를 에지(Edge), 질병을 에지 타입(Edge Type)으로 표현한다.

각 에지는 두 유전자 간의 관계에서 해당 질병이 발현될 확률로 0에서 1사이 값을 갖는다. 에지 타입은 시각화 시, 색으로 표현한다. 또한 에지는 연결된 두 노드를 원인 유전자(Source Gene)와 결과 유전자(Target Gene)를 설정하여 방향성을 갖는다. 이러한 방향성은 해당 질병에 대해 원인 유전자가 영향을 미칠 때 결과 유전자가 질병과 관련이 있을 확률을 나타낸다. 또한 임계값 사용 유무에 따라 연산 시 노드 간의 에지를 사용하지 않는 경우가 발생할 수 있다.

Fig. 2는 해당 그래프 모델을 시각화 한 것이다.

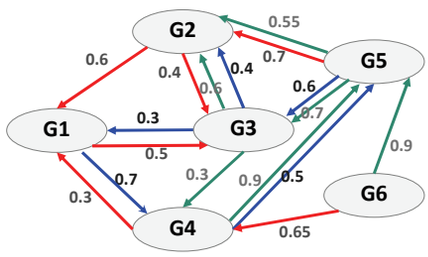

Fig. 2. Gene Disease Relation Extension Graph

Fig. 2에서 유전자 G2와 유전자 G3 관계를 예시로 든다. 이 두 유전자는 질병 Red, Green, Blue에 대해 연관되어 있고 질병 Red, Green, Blue에 대해 각각 0.4, 0.6, 0.4의 관련도를 갖는다.

5. 알고리즘 설계 및 구현

본 장에서는 3장에서 소개된 6가지 분석 방식 메소드에 대한 알고리즘을 소개한다.

5.1 Finding Genes for given Disease

1) findRelatedGenes_SD() 메소드 알고리즘

Table 4는 3.2.1)절에 소개된 findRelatedGenes_SD() 메소드에 대한 알고리즘이다.

Table 4. A Disease and Gene Relation Algorithm

Algorithm 1. A Disease and Gene Relation Algorithm

input: x, disease, allRelation

output: set of Gene genes

```
1  function findRelatedGenes_SD()
2    cmmRel :=allRelation['diseaseID' is disease]
3    cmmGene := cmmRel['sourceGeneID']
4    for each i in cmmGene:
5      Genes :=(average relevance to all path i)
6    genes.sort()
7    return genes[:x]
```

이 알고리즘은 1. 해당 질병의 관련도(cmmRel)에 대한 데이터 추출(2줄), 2. 관련도에 속해 있는 유전자(cmmGene) 추출(3줄), 3. 각 유전자의 관련도 평균 구한 후(4~5줄) 정렬(6줄),

4. x개의 유전자를 반환(7줄) 순서로 진행한다. Table 4의 5번째 줄에서 한 유전자에 대한 관련도를 측정할 때 사용되는 알고리즘은 O(n3)의 시간 복잡도를 갖는다. 위의 알고리즘을 f1(n)으로 표현하게 되면 Equation (1)과 같이 표현된다.

$$f_1(n) = n^4 + 2n + 3 \quad (1)$$

$$O(g(n)) = O(n^4) \quad (2)$$

이 때 $f_1(n)$의 최대 증가율 g(n)은 n^4으로써 시간복잡도는 $O(n^4)$ Equation (2)이다.

2) findRelatedGenes_MD() 메소드 알고리즘

이 알고리즘은 5.1.1)절에 소개된 알고리즘과 같은 방법으로 진행된다. 이 때 Table 4의 2번째줄에서 disease가 아닌 diseaseSet을 사용하여 질병 집합에 대해 관련도가 높은 유전자를 구한다. 따라서 시간 복잡도는 5.1.1)절과 같은 O(n4)이다.

5.2 Finding Diseases for given Genes

1) findRelatedDiseases_SG() 메소드 알고리즘

Table 5는 3.2.2)절에 소개된 메소드 중 findRelated Diseases_SG() 메소드에 대한 알고리즘이다.

Table 5. A Gene and Disease Relation Algorithm

Algorithm 2. A Gene and Disease Relation Algorithm	
input: x, gene, allRelation	
output: set of Diseases diseases	
1	**function** findRelatedDiseases_SG()
2	**for** i **in** 0.. allDisease.length:
3	diseases[i] = 0
4	trg_Rel :=allRelation['TargetGeneID'is gene]
5	**for** i **in** trg_Rel[‘DiseaseID’]:
6	diseases[i] += 1
7	diseases.sort()
8	**return** diseases[:x]

이 알고리즘은 1. 질병 집합(diseases) 초기화(3줄), 2. 입력한 유전자에 관련이 있는 집합(trg_Rel) 추출(4줄), 3. 만들어진 집합을 통해 질병 빈도수 조사(5, 6줄) 및 정렬(7줄), 4. x개의 관련도가 높은 질병을 반환(8줄) 순서로 진행한다.

이 알고리즘을 $f_2(n)$으로 표현하게 되면 Equation (3)과 같이 표현된다.

$$f_2(n) = 4n + 3 \quad (3)$$

$$O(g(n)) = O(n) \quad (4)$$

이 때 $f_2(n)$의 최대 증가율 g(n)은 n으로써 시간복잡도는 O(n) Equation (6)이다.

2) findRelatedDiseases_MG() 메소드 알고리즘

Table 6은 findRelated Diseases_MG() 메소드에 대한 알고리즘이다.

이 알고리즘은 1. 조사하고 싶은 질병의 부분 집합(dis_subset)에서 해당 질병의 관련도(cmmRel)에 대한 데이터 추출(2줄), 2. 관련도에 속해있는 유전자(cmmGene) 추출(3줄), 3. 각 유전자의 관련도 합 구한 후(5~6줄) 정렬(8줄), 4. x개의 유전자를 반환(9줄) 순서로 진행한다.

Table 6. Gene Set and Disease Relation Algorithm

Algorithm 3. Gene Set and Disease Relation Algorithm	
input: x, geneSet, allRelation, allDisease, dis_subset	
output: set of Gene genes	
1	**function** findRelatedDiseases_MG()
2	cmmRel :=allRelation['diseaseID' is dis_subset]
3	cmmGene := cmmRel['sourceGeneID']
4	**for** i **in** cmmRel:
5	**for** each j **in** cmmGene:
6	gene :=(most relevance to all path j)
7	diseases[i] = gene
8	diseases.sort()
9	**return** diseases[:x]

이 알고리즘은 질병의 부분 집합을 만들어 결과를 구한다. 또한 Table 6의 6번째 줄에는 5.1.1)절과 같은 관련도 측정 알고리즘을 사용한다. 따라서 해당 분석 방식 알고리즘의 시간복잡도는 $O(n^4)$이다.

5.3 Finding Genes for given relation

1) findRelatedGenes() 메소드 알고리즘

Table 7은 3.2.3)절에 소개된 findRelatedGenes() 메소드에 대한 알고리즘이다.

Table 7. Gene Set and Relation Algorithm

Algorithm 4. Gene Set and Relation Algorithm	
input: x, allRelation	
output: set of Genes genes	
1	**function** findRelatedGenes ()
2	allRelation[‘relevance’].sort()
3	genes = allRelation[‘SourceGeneID’]
4	**return** genes[:x]

이 알고리즘은 1. 모든 관련도 데이터를 관련도(relevance) 기준으로 내림차순 정렬(2줄), 2. 정렬된 데이터 중 원인 유전자를 추출(3줄), 3. x개의 유전자 집합을 반환(4줄) 순서로 진행한다.

이 알고리즘을 $f_3(n)$으로 표현하게 되면 Equation (5)와 같이 표현된다.

$$f_3(n) = 2n \tag{5}$$

$$O(g(n)) = O(n) \tag{6}$$

이 때 $f_3(n)$의 최대 증가율 g(n)은 n으로써 시간복잡도는 시간 복잡도는 O(n) Equation (6)이다.

5.4 Finding Diseases for given Relation

1) findRelatedDiseases() 메소드 알고리즘

이 알고리즘은 5.3.1)절에 소개된 알고리즘과 같은 방법으로 진행된다. 이 메소드는 질병을 기준으로 하기 때문에 Table 7의 3번째 줄에서 SourceGeneID가 아닌 DiseaseID를 사용한다. 또한 x개의 질병 집합이 반환된다. 이 알고리즘의 시간복잡도는 5.3.1)절의 알고리즘과 같은 O(n)이다.

본 플랫폼의 분석 방식 알고리즘 중 findRelatedGenes_SD(), findRelatedGenes_MD(), findRelatedDiseases_MG() 메소드 알고리즘들의 시간복잡도는 $O(n^4)$이다. 반면에 나머지 메소드 알고리즘들의 시간복잡도는 O(n)이다. 따라서 분석에 필요한 데이터 크기가 커질수록 앞의 세 메소드 알고리즘이 그 외 알고리즘보다 분석에 소요되는 시간이 더욱 증가한다.

6. 알고리즘 효율성 최적화

본 절에서는 플랫폼에서 사용하는 관련도 측정 알고리즘을 소개하고, 해당 알고리즘을 대용량의 데이터 환경에서 사용할 때 생기는 시간 효율성 측면의 문제점을 제시한다. 또한 이 문제점을 해결하기 위한 다이나믹 프로그래밍 기반 알고리즘과 근사 (Approximation) 기반 알고리즘을 소개한다. 이 알고리즘들의 시간복잡도를 통해 원래 측정 알고리즘보다 시간 효율성이 향상되는 것을 보인다.

6.1 관련도 측정 효율성 알고리즘

본 플랫폼은 그래프 모델을 통하여 인접하지 않은 유전자 간의 관련도를 구할 수 있다. 이 때 플랫폼에 맞게 Dijkstra 알고리즘을 변형하여 유전자 간 관련도를 구한다. Table 8은 관련도 측정 효율성 알고리즘이다.

기본적으로 Dijkstra 알고리즘은 한 노드(Node)에서 각각의 노드에 대한 경로(<Table 8> 9줄)를 찾는다. 이 때 현재 노드에서 이웃한 노드를 거쳐 갔을 때의 관련도 비교를 위해 이웃한 노드에 대한 반복을 진행한다(<Table 8> 10줄). 따라서 이 알고리즘을 사용하게 되면 한 유전자에 관련된 n개의 유전자에 대해 평균 관련도를 얻을 때 시간 복잡도가 $O(n^2)$인 Dijkstra 알고리즘을 총 n번 연산하여 구한다. 이 알고리즘을 f4(n)으로 표현하면 Equation (7)과 같이 표현된다.

$$f_4(n) = n^2 * (n) = n^3 \tag{7}$$

이 때 $f_4(n)$의 최대 증가율 g(n)은 n^3으로써 이 알고리즘의 시간 복잡도는 $O(n^3)$이다(Equation (8)).

Table 8. Relation Measurement Algorithm

Algorithm 5. Relation Measurement Algorithm

input: graph, source, target
output: set of distance dist, path of predecessor pred, relevance value score

```
1  function dijstra(graph, source, target)
2    for node in graph:
3      if node == source:
4        pq[node] := 1
5      else
6        pq[node] := 0
7      endif
8      while pq:
9        node, pq := find_current_node()
10       for nghb in graph[node]:
11         if nghb in pq:
12           new_score:= dist[node]*graph[node][nghb]
13           if new_score > pq[nghb]:
14             …
15           endif
16         endif
17       endfor
18       if node == tareget:
19         return dist, pred, dist[target]
20       endif
21   endwhile
```

$$O(g(n)) = O(n^3) \tag{8}$$

따라서 관련도 측정 효율성 알고리즘은 Equation (8)의 시간복잡도에 의하면 데이터 양이 커질수록 시간복잡도가 매우 크게 증가하는 형태의 알고리즘이다.

6.2 다이나믹 프로그래밍 기반 최적화

관련도 측정 알고리즘은 $O(n^3)$의 시간복잡도로 데이터의 크기가 커질수록 런타임이 매우 크게 증가하는 문제점이 생긴다. 따라서 이러한 이슈를 다이나믹 프로그래밍 (Dynamic Programming) 기법을 적용하여 해결하는 알고리즘을 제안한다. Table 9는 해당 최적화 알고리즘을 나타낸다. 다이내믹

Table 9. Dynamic Programming Based Algorithm

Algorithm 6. Dynamic Programming Based Algorithm

input: graph, source gene, target gene, record
output: set of distance, path of predecessor, relevance

```
1  function optimized_algorithm()
2    ...
3    while pq is not empty:
4      node := find_current_node(pq)
5      if (source to target) in record:
6        pred, dist := record[(source to target)]
7        return dist, pred, dist[target]
8      else:
9        for each nghb in node neighbors list:
10         score:=(relevance from node to nghb)
11         if score > (previous relevance path):
12           ...
13       record[(source to target)] := (pred, dist)
14   return dist, pred, dist[target]
```

프로그래밍 중 메모이제이션 (Memoization) 기법을 사용한 알고리즘이다. Table 9의 13줄에 해당 기법을 적용하여 구한 두 유전자 사이의 관련도를 저장한다.

Table 9에 제시한 알고리즘을 사용하여 한 유전자에 대해 관련을 찾을 때, 즉 최적화 알고리즘을 적용하였을 경우를 $f_5(n)$으로 표현하면 Equation (9)와 같이 표현된다.

$$f_5(n) = n^2 + 1 + \cdots + 1 = n^2 + (n-1) \quad (9)$$

한 유전자가 처음 다른 유전자와의 관계를 찾을 때 모든 유전자와 관계를 알아내야 하므로 n^2번 반복한다. 그리고 구해진 관계를 재사용을 위해 저장한다. 나머지의 경우(<Table 9> 5~7줄)는 앞에서 저장된 관계를 이용하여 결과를 추가적인 연산 없이 얻을 수 있다. 따라서 해당 알고리즘은 Equation (9)로 나타낸다.

처음에 한 최적화 알고리즘에서의 f5(n)의 최대 증가율 g(n)은 n2으로써 시간 복잡도는 O(n2)이 된다(Equation (10)).

$$O(g(n)) = O(n^2) \quad (10)$$

본 절의 최적화 알고리즘을 사용하였을 때 6.1의 알고리즘보다 공간 복잡도가 증가하나, 시간 복잡도는 $O(n^2)$로 6.1의 알고리즘의 시간복잡도인 $O(n^3)$보다 n만큼 감소한다. 따라서 이 최적화 알고리즘을 사용하면 6.1의 알고리즘과 같은 결과를 얻지만 데이터가 증가할수록 런타임이 감소되는 효과를 얻는다.

6.3 근사 (Approximation) 기반 최적화

6.2의 최적화 알고리즘을 이용하여 질병 관련도를 얻을 때도 데이터의 크기에 따라 런타임이 다소 증가한다. 따라서 관련도가 적은 데이터를 없애고 연산하는 근사 알고리즘을 통해 이러한 문제를 해결한다. Table 10은 본 논문에서 제시하는 근사 알고리즘이다.

Table 10. Approximation Based Algorithm

Algorithm 7. Approximation Based Algorithm	
input: minimum relevance value, relation data set **output**: relation data set	
1	**function** approximated_algorithm ()
2	**for** data **in** relation_data_set:
3	**if** data['relevance']> min_rele_value:
4	data_set = data
5	**return** data_set

근사 알고리즘은 사용하는 관련도 데이터에서 질병 관련도가 일정 값 (min_rele_value) 이하 데이터를 제외한다(<Table 10> 3, 4줄).

이 알고리즘은 일정 관련도 이상의 데이터들만 사용하므로 연산에 사용되는 데이터의 크기가 앞서 소개한 두 알고리즘보다 줄어들게 된다. 따라서 이 알고리즘의 시간 복잡도는 최적화 알고리즘과 같지만 데이터의 개수가 줄어들기 때문에 비교적 런타임이 감소하게 된다.

7. 실험 및 평가

7.1 실험 데이터 베이스

본 실험에서는 6개의 질병 데이터와 487개의 유전자 데이터, 3,252개의 관련도 데이터를 사용하였다. Fig. 3은 관련도 데이터의 일부를 보여주고 있다.

DiseaseID	SourceGeneID	TargetGeneID	relevance
1	21	16	1
1	16	21	1
1	213	358	0.5
1	358	213	0.5
1	148	230	0.25
1	230	148	0.25
1	85	426	0.5
1	426	85	0.5
1	65	426	0.5

Fig. 3. A Part of Relation Data

7.2 유전자 간 관련도 측정 실험

본 절에서는 5장의 메소드 중 일부에 대해 실험을 진행하였다. 이 실험에서는 각각의 메소드에 대해 임계값을 사용하지 않았을 때와 임계값을 사용 했을 때의 결과 일치율을 확인하였다.

1) findRelatedGenes_SD() 메소드 실험

본 절에서는 findRelatedGenes_SD()와 findRelatedG enes_SD_TH()에 대한 실험을 진행하였다.

findRelatedGenes_SD() 메소드를 이용하여 주어진 6개의 질병에 대해 관련도가 높은 3개의 유전자 집합을 얻는 실험을 하였다. Table 11은 해당 실험에 대한 결과이다.

Table 11. findRelatedGenes__SD() Method Result

Disease ID	Gene ID
1	[213, 401, 323]
2	[39, 35, 53]
3	[426, 323, 153]
4	[249, 250, 154]
5	[474, 473, 136]
6	[42, 449, 301]

질병 1에 대해 유전자 ID 213, 401, 323가 높은 관련도를 보였다. 또한 질병 2에 대해 유전자 ID 39, 35, 53이 높은 관련도를 보였다.

findRelatedGenes_SD_TH() 메소드를 이용하여 위와 같은 조건으로 실험을 진행하였다. Fig. 4는 두 실험의 결과를 비교한 것이다.

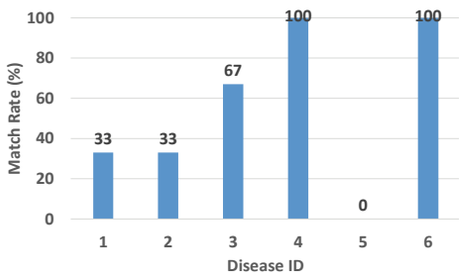

Fig. 4. Comparing findRelatedGenes__SD Methods

질병 4와 6을 제외한 나머지의 질병에서는 임계값을 사용하지 않았을 때와 결과에 차이를 보였다. 특히 질병 5의 경우에 대해서는 일치율이 0%이다.

2) findRelatedGenes_MD() 메소드 실험

본 절에서는 findRelatedGenes_MD()와 findRelated Genes _MD _TH()에 대한 실험을 하였다.

우선 findRelatedGenes_MD() 메소드를 이용하여 3개의 질병과 2개의 질병에서 각각 관련도가 높은 2개의 유전자로 된 집합을 얻는 실험을 진행하였다. Table 12는 해당 실험에 대한 결과이다.

Table 12. findRelatedGenes__MD() Method Result

Disease ID	Gene ID
[2, 5, 6]	[473, 346]
[4, 5]	[11, 294]

질병 2, 5, 6에 대해 유전자 ID 473, 346가 높은 관련도를 보이고, 질병 4, 5에 대해 유전자 ID는 11, 294가 높은 관련도를 보임을 확인할 수 있었다.

findRelatedGenes_MD_TH() 메소드를 이용하여 위와 같은 조건으로 실험을 진행하였다. Fig. 5는 두 실험의 결과를 비교한 것이다.

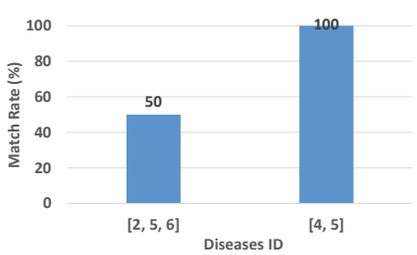

Fig. 5. Comparing findRelatedGenes__MD Methods

임계값을 사용했을 때 질병 4, 5에 대해서 100%의 일치율을 보인 반면 질병 2, 5, 6에 대해서는 50%의 일치율을 보였다.

3) findRelatedDiseases_SG() 메소드 실험

본 절은 findRelatedDiseases_SG()와 findRelated Diseases _SG_TH()에 대해 실험을 하였다.

findRelatedDiseases_SG() 메소드를 사용하여 유전자 ID 4, 35, 173, 288, 300가 영향을 미치는 6개의 질병을 얻는 실험을 하였다. Table 13은 해당 실험의 결과이다.

Table 13. findRelatedDiseases__SG() Method Result

Gene ID	Disease ID
4	[5]
35	[4, 5, 2]
173	[6, 5, 4, 2]
288	[2, 1]
300	[4, 3, 6, 5, 2]

유전자 ID 4는 1개의 질병에, 유전자 ID 300은 5개의 질병에 영향을 주는 것을 확인할 수 있다.

findRelatedDiseases_SG_TH() 메소드를 이용하여 위와 같은 조건으로 실험을 진행한 후 임계값을 사용하지 않았을 때와 비교하였다. Fig. 6은 해당 비교의 결과이다.

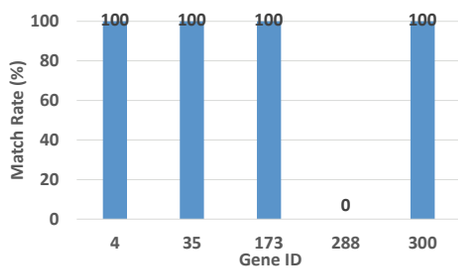

Fig. 6. Comparing findRelatedDiseases__SG Methods

임계값을 사용한 결과는 사용하지 않은 결과와 대부분의 경우에서 일치하게 나왔지만 유전자 ID 288일 때 0%의 일치율을 보였다.

7.3 효율성 최적화 알고리즘 실험

1) 다이나믹 프로그래밍 기반 최적화

본 절에서는 6.2절에서 다이나믹 프로그래밍 최적화 알고리즘을 이용한 실험을 진행한다. 결과를 통해 최적화 알고리즘 사용 유무를 소요시간 기준으로 비교한다.

관련도 측정 알고리즘이 쓰인 메소드에서 시간복잡도가 큰 3가지 알고리즘 중 findRelatedGenes_SD() 메소드를 사용하여 다이나믹 프로그래밍 기반 최적화 알고리즘을 적용하지 않았을 때와 적용했을 때의 소요시간을 측정하였다. Fig. 7은 위의 실험에 대한 결과이다.

Fig. 7에서 해당 최적화 알고리즘을 적용하지 않은 것은 Before로 적용한 것은 After로 표기하였다. 실험에 사용된 유전자 데이터 개수는 질병 4와 5가 219개, 216개로 가장 많았고, 질병 1이 25개로 가장 적었다. 또한 알고리즘 적용 전에서는 질병 1은 0.43초, 질병 4는 158.06초, 질병 5는 174.33초가 소요되었고, 적용 후에서는 질병 1은 0.41초, 질병 4는 37.18초, 질병 5는 36.67초가 소요되었다.

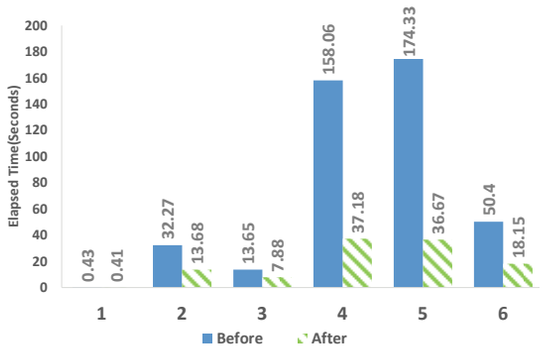

Fig. 7. First Method Elapsed Time

따라서 데이터 개수가 증가할수록 효율성 최적화 알고리즘 적용 전과 후의 소요시간 감소량이 큰 폭으로 증가하는 것을 확인할 수 있었다.

해당 최적화 알고리즘의 실험을 통해 이 알고리즘을 사용했을 때 원래의 알고리즘을 사용했을 때와 같은 결과를 얻을 수 있지만 소요 시간이 단축되는 것을 확인할 수 있었다. 또한 데이터 크기와 소요 시간의 차가 비례함을 볼 수 있었다.

2) 근사 기반 최적화

본 절에서는 6.3절에서 소개한 근사 기반 최적화 알고리즘을 이용한 실험을 진행한다. 아래의 결과는 근사 알고리즘을 사용하는 실험 중 일부이다.

findRelatedGenes_SD() 메소드를 사용하여 근사 알고리즘을 최소 관련도 값을 각각 0.1, 0.2, 0.3, 0.4으로 설정했을 때의 경우에 대해 진행을 하였다. Fig. 8은 위의 실험에 대한 결과이다.

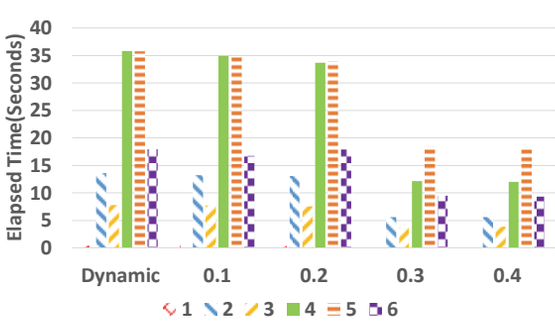

Fig. 8. Comparing Approximated Algorithm Elapsed Time with Optimized Algorithm

이 때 0.3일 때 소요시간이 크게 감소하였다. 데이터의 크기 또한 0.3일 때 3000개에서 1104개로 감소했다. 질병 4와 5의 경우 0.2일 때까지 30초 이상 소요되었는데 0.3부터 질병 4는 12초, 질병 5는 18초가 소요되어 약 40%가량 감소했다.

Fig. 9는 본 실험과 7.3.1)의 결과를 관련도 측정 실험 결과를 정확도 비교한 결과이다.

전체적으로 Fig. 9에서도 0.3일 경우에 큰 변화량을 보였다. 질병 2와 5의 경우 0.4까지 100%를 유지하였으나 질병 1과 3은 40%까지 감소하면서 가장 큰 변화량을 보였다.

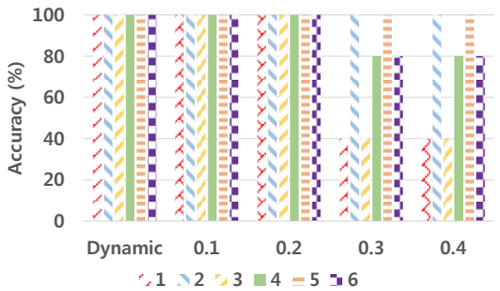

Fig. 9. Comparing Approximated Algorithm Accuracy with Optimized Algorithm

위의 실험을 통해 근사 기반 알고리즘을 적용하였을 때가 적용하지 않았을 때보다 소요 시간과 정확도가 감소하는 것을 볼 수 있었다. 이 때 두가지 요소의 감소율은 데이터의 개수에 비례하였다. 따라서 근사 기반 최적화 알고리즘도 최소 관련도 값 조정을 통해 런타임을 줄일 수 있으나 결과에 대한 정확도가 낮아졌다.

7.2절과 7.3절의 실험결과들은 소요 시간과 정확도 부분에서 차이를 보인다. 소요 시간 측면에서 관련도 측정 알고리즘, 다이나믹 프로그래밍 기반 최적화 알고리즘, 근사 기반 최적화 알고리즘 순서대로 소요 시간이 감소한다. 정확도 측면에서는 다이나믹 프로그래밍 기반 최적화 알고리즘의 경우 관련도 측정 알고리즘과 항상 같은 결과를 보이는 반면, 근사 기반 최적화 알고리즘의 경우 설정한 최소 관련도 값에 따라 정확도가 달라진다. 최소 관련도 값이 커질수록 연산에 제외되는 데이터의 수가 많아지므로 정확도가 낮아진다.

8. 결 론

유전병에 대해 분석을 할 때 유전자와 질병 사이의 관계에서 생길 수 있는 여러 가지 분석 방식 요구 사항과 방대한 양의 데이터들을 처리할 때 생기는 시간 효율성의 저하가 생길 수 있다.

본 논문에서는 유전자와 질병 사이의 관계를 나타내는 방식을 그래프 모델을 통해 제안하고, 이 모델을 이용하여 유전자와 질병 사이의 관계를 여섯 가지 분석 방식을 제공하였다. 또한 방대한 양의 유전자와 질병을 처리하면서 생기는 효율성에 대한 이슈에 대한 해결책으로 최적화 알고리즘을 제안하였다. 그리고 실험을 통해 제안한 접근 방식 및 효율성 최적화 알고리즘의 유효성을 검증하였다.

References

[1] Man-Sun Kim and Jeong-Rae Kim, “Characterization of the Alzheimer’s disease-related network based on the dynamic network approach,” *Journal of Korean Institute of Intelligent Systems*, Vol.26, No.6, pp.529-535, Dec. 2015.

[2] Kai Zheng, Gangquan Si, and Chen Duan., “Diagnosis of kashin-beck disease and other common joint diseases via a

gene model," in *Proceedings of IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC 2017)*, Chengdu, China, Dec. pp.808-812, 2017.

[3] Gui-Qiong Zhu and Pei-Hui Yang, "Identifying the candidate genes for Alzheimer's disease based on the rejection region of T test," *International Conference on Machine Learning and Cybernetics (ICMLC 2016)*, Vol.2, pp.732-736, 2016.

[4] Rohit Gupta, S. M. Fayaz, and Sanjay Singh, "Identification of gene network motifs for cancer disease diagnosis," in *Proceedings of IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER 2016)*, India, pp.179-184, Aug. 2016.

[5] Jahnabi Dutta, Surama Biswas, and Souvik Saha, "Identification of disease-critical genes causing preeclampsia: Meta-heuristic approaches" in *Proceedings of IEEE UP Section Conference on Electrical Computer and Electronics (UPCON 2015)*, Allahabad, India, pp.1-6, Dec. 2015.

[6] Ji-Hwan Ha, Hyun-Jin Kim, and Sang-Hyun Park, "An Extraction Method of Disease-Related miRNA through the Target Gene Based miRNA Network," *Journal of Korean Institute of Information Technology*, Vol.13 No.1, pp.113-119, Jan. 2015.

[7] Jong-Keun Lee, S.S Park, D.W. Hong, et al, "Development of a Gene's Functional Classifying System for a Microarray Data using a Gene Ontology," *Journal of Korea Information Science Society*, Vol.33, No.2C, pp.246-251, Oct. 2006.

[8] Xiwei Tang, Xiaohua Hu, Xuejun Yang, et al., "A algorithm for identifying disease genes by incorporating the subcellular localization information into the protein-protein interaction networks," in *Proceedings of IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2016)*, Shenzhen, China, Dec. pp.308-311, 2016.

[9] Shadi A. Aljawarneh, Reem Jaradat, Abdelsalam M. Maatuk, et al., "Gene profile classification: A proposed solution for predicting possible diseases and initial results," in *Proceedings of International Conference on Engineering & MIS (ICEMIS 2016)*, Agadir, Morocco, pp. 1-7, Sep. 2016.

[10] Shadi A. Aljawarneh, Reem Jarada, et al., "Gene profile classification A proposed solution for predicting possible diseases and initial results," in *Proceedings of IEEE Engineering & MIS (ICEMIS 2016)*, Agadir, Morocco, pp.1-7, Nov. 2016.

[11] Amirhossein Tavanaei, Nishanth Anandanadarajah, et al., "A deep learning model for predicting tumor suppressor genes and oncogenes from PDB structure," in *Proceedings of Bioinformatics and Biomedicine (BIBM 2017)*, MO, USA, pp.613-617, Dec. 2017.

[12] TengJiao Wang, Wei Liu HaiLin, et al., "Predicting potential disease-related genes using the network topological features," in *Proceedings of Human Health and Biomedical Engineering (HHBE 2011)*, Jilin, China, pp.871-876, Sep. 2011.

[13] Mustafa Özgür Cingiz, and Banu Diri, "The inference of gene co-expression networks of breast and colon cancer using miRNA-target gene interactions data," in *Proceedings of Signal Processing and Communications Applications Conference (SIU 2017)*, Antalya, Turkey, pp.1-4, Jun. 2017.

[14] Ashkan Entezari Heravi and Sheridan Houghten, "A methodology for disease gene association using centrality measures," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2016)*, Vancouver, Canada, pp. 24-31, Jul. 2016.

[15] Ashwani Kumar and Tiratha Raj Singh, "Systems biology approach for gene set enrichment and topological analysis of Alzheimer's disease pathway," in *Proceedings of International Conference on Bioinformatics and Systems Biology (BSB 2016)*, Allahabad, India, pp.1-5, Mar. 2016.

[16] Nivit Grewal, Shailendra Singh, and Trilok Chand, "Effect of Aggregation Operators on Network-Based Disease Gene Prioritization: A Case Study on Blood Disorders", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol.14, No.6, pp.1276-1287, Aug. 2016.

[17] Kazutaka Nishiwaki, Katsutoshi Kanamori, et al., "Finding a disease-related gene from microarray data using random forest," in *Proceedings of Cognitive Informatics & Cognitive Computing (ICCI*CC 2017)*, CA, USA, pp.542-546, Feb. 2017.

[18] Khalid M.O. Nahar and Izzat Alsmadi, "Information analysis to study interactions between different genes and diseases," in *Proceedings of 8th International Conference on Information Technology (ICIT 2017)*, Bhubaneswar, India, pp.1-5, Dec. 2017.

[19] Zhen Tian, Maozu Guo, Chunyu Wang, et al., "Constructing an integrated gene similarity network for the identification of disease genes," in *Proceedings of IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2016)*, Shenzhen, China, pp.1663-1668, Dec. 2016.

[20] Kyungsik Ha, Jinmuk Lim, and Hong-Gee Kim, "Integrated Model design of miRNA, PPI and Disease Information," *Journal of Korea Information Science Society*, Vol.39, No.1B, pp.492-494, Jun. 2012.

[21] Kyungsik Ha, Jinmuk Lim, and Hong-Gee Kim, "Integrated Model Design of Microarray Data Using miRNA, PPI, Disease Information," *Journal of Korean Institute of Intelligent Systems*, Vol.22, No.1B, pp.492-494, Jun. 2012.

[22] Jeongwoo Kim and Sanghyun Park, "ISN: Inferring disease-related genes using seed gene and network

analysis," in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC 2017)*, Banff, Canada, pp.2012-2017, Oct. 2017.

[23] Sun-Wook Choi and Chong Ho Lee, "Gene Expression Data Analysis Using Parallel Processor based Pattern Classification Method," *Journal of The Institute of Electronics Engineers of Korea*, Vol.46, No.6, pp.44-55, Nov. 2009.

[24] Kai Zheng, Gangquan Si, and Chen Duan., "Application of a gene diagnosis model in the identification of Kashin-Beck disease and similar joint diseases," in *Proceedings of IEEE International Conference on Computer and Communications (ICCC 2017)*, Chengdu, China, pp.2669-2673, Dec. 2017.

[25] Pradipta Maji and Ekta Shah, "Significance and Functional Similarity for Identification of Disease Genes," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol.14, No.16, pp.1419-1433, Aug. 2016.

[26] Hyung-Seok Choi, "Systematic Integrative Analysis of Heterogeneous Bio Big Data for Identification of Disease-causing Gene Network," *Journal of Korea Information Science Society*, Vol.31, No.5, pp.61-68, Aug. 2013.

[27] Mengmeng Wu, Wanwen Zeng, Wenqiang Liu, et al., "Integrating embeddings of multiple gene networks to prioritize complex disease-associated genes," in *Proceedings of IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2017)*, MO, USA, pp.208-215, Nov. 2017.

[28] Sun-Wook Choi and Chong Ho Lee, "Disease Classfication using Random Subspace Method based on Gene Interaction Information and mRMR Filter," *Journal of Korean Institute of Intelligent Systems*, Vol.22, No.2, pp.192-197, Apr. 2012.

[29] Chae-Gyun Lim, A.K.M. Tauhidul Islam, and Byeong-Soo Jeong, "Efficient Parallel Gene Selection Method based on MapReduce," *Journal of Korea Information Science Society: Databases*, Vol.41, No.3, pp.162-167, Jun. 2014.

송 명 호

https://orcid.org/0000-0001-8581-3382
e-mail : songmho@gmail.com
2017년 연세대학교 원주캠퍼스 컴퓨터공학부(학사)
2018년~현 재 숭실대학교 컴퓨터학과 석사과정
관심분야 : 소프트웨어 디자인(Software Design), 머신러닝 기반 어플리케이션(Machine Learning based Application), 사람 안전 시스템(People Safety System)

김 수 동

https://orcid.org/0000-0001-7585-1801
e-mail : sdkim777@gmail.com
1984년 Northeast Missouri State University 전산학(학사)
1988년/1991년 The University of lowa 전산학(석・박사)
1991년~1993년 한국통신 연구개발단 선임연구원
1994년~1995년 현대전자 소프트웨어연구소 책임연구원
1995년~현 재 숭실대학교 컴퓨터학부 교수
관심분야 : 객체지향 모델링(Object-Oriented Modeling), 소프트웨어 아키텍처(Software Architecture), 컨텍스트 인지 서비스(Context-Aware Service), 사물 인터넷 컴퓨팅(Internet of Things Computing)