사례기저 수치화와 트리구조 관리를 이용한 서비스 컴포지션의 속도 향상 연구

이승훈°, 김건수, 윤태복, 박두경, 이지형
성균관대학교 컴퓨터공학과
{reinblame, kkundi, tbyoon, haderme}@skku.edu, jhlee@ece.skku.ac.kr

A Fast Service Composition Method via Case-Base Numerization and Tree Structure Management

Seunghoon Lee°, Kunsu Kim, Taebok Yoon, Dookyung Park, Jee-Hyung Lee
Department of Information Communication Engineering, Sungkyunkwan University

요 약

유비쿼터스 컴퓨팅의 목표 중 하나는 사용자의 직접적이거나 은연중에 내포된 요청에 따라 적절한 서비스를 제공하는 것이다. 최근에는 사용자의 다양한 요청에 보다 유연하게 대응할 수 있는 연구가 이루어지고 있으며 그 중 단일서비스의 조합을 통해 복합서비스를 제공할 수 있는 서비스 컴포지션(Service Composition)이 주목을 받고 있다. 하지만 기존 연구들은 늦은 처리속도로 인해 실시간 상황인식 서비스에서 빠른 응답을 바라는 사용자의 요구사항을 만족시키기 어렵다. 또한 사례기반추론은 사례기저에 쌓인 사례의 수가 늘어감에 따라 속도가 저하되는 단점이 있다. 이러한 단점을 최소화 하기 위하여 클러스터링 기법이 사용되고 있다. 그러나 클러스터링 기법은 사례기저의 수를 줄여가면서 속도를 유지하기 때문에 기존의 사례가 다시 발생하였을 경우에도 새로운 문제로 인식하게 되는 문제를 가지고 있다. 이러한 문제점을 해결하기 위해서 본 논문에서는 사례기저의 수를 유지하면서 사례기저의 수치화 및 트리구조 관리를 이용하여 기존방법보다 빠른 서비스 컴포지션을 구현하는 방법을 제안한다. 그리고 기존의 서비스 컴포지션 기법과 비교 분석을 통하여 제안하는 기법의 유효함을 확인하였다.

1. 서 론

최근 유비쿼터스 컴퓨팅에 대한 많은 연구들이 진행중이고, 그 중 상황인식 서비스에 대한 연구도 여러 분야에 걸쳐 이루어지고 있다. 특히 대부분의 연구들이 공통적으로 보다 다양한 서비스의 지원을 목적으로 하는 상황인식 프레임워크의 개발을 목표로 하고 있다. 이에 따라 많은 서비스에서 사용자를 만족시킬 수 있는 서비스의 조합을 찾기 위한 방법 중 하나로 서비스 컴포지션[1] 이라는 기술을 사용하고 있다.

서비스 컴포지션은 하나의 서비스를 단일 서비스(primitive service)로 정의하고, 이러한 단일 서비스들을 조합한 복합서비스(composite service)를 생성하여 사용자의 요구사항을 만족시키는 것을 목표로 한다. 예를 들어 여행서비스가 있다고 가정하자. 이 서비스는 사용자의 일정과 예산에 맞춰 비행기 표 예매 서비스와 숙소 예약서비스를 제공한다. 하지만 이 두 가지 서비스를 서로 독립적으로 나누어서 제공한다면, 사용자의 일정 또는 허용된 예산을 벗어나는 경우가 생길 수 있다. 두 가지 서비스에서 최적의 조합을 선택하여 사용자에게는 하나의 복합서비스(compound service)를 생성하는 것이 바로 서비스 컴포지션이다. 서비스 컴포지션 기법으로 주로 사용되는 알고리즘으로 Hierachical Task Network (HTN), Model Checking 등을 이용한 방법들이 제안되었다. 이 중 HTN은 웹 서비스 기반의 서비스 조합에 대한 대표적인 연구인 Maryland대학의 SHOP2[2] 시스템에 사용되고 있으며, SHOP2는 Hierarchical Task Network[3] 기반의 AI Planning 기술을 적용하여 OWL-S[4]로 의미가 기술된 웹 서비스 컴포지션 방법을 수행한다.

하지만 이런 기법들은 모두 탐색을 기반으로 하고 있어서 서비스의 수가 증가함에 따라 긴 탐색 시간을 갖게 된다. 유비쿼터스 환경에서 상황인식 서비스의 대부분이 빠른 응답시간을 요구하는 경향이 있다. 이러한 요구사항을 충족시키기 위해 실시간으로 응답이 가능한 빠른 서비스 컴포지션 기법이 필요하다. 하지만 일반적인 서비스 컴포지션 기법은 이러한 상황인식 서비스의 요구사항을 만족시키기는 어려운 경우도 있다.

그래서 본 논문에서는 사례기반추론을 이용한 서비스 매칭 기법을 이용하여 사용자의 요구에 보다 빠르게 응답할 수 있는 방법을 제안한다. 사례기반추론은 인간의 지적 활동을 모델화한 것으로, 과거문제로부터 얻은 상황 경험이나 지식을 사례 데이터베이스로 구축하여 어떠한 상황이나 문제가 발생하면 기존 사례 데이터베이스에서 똑같거나 또는 가장 유사한 사례를 선택하여 그 사례가 제시하는 해결책으로 현 문제에 대한 답을 제시한다[5].

이러한 사례기반추론의 정확도는 보유하고 있는 사례의 양에 비례한다. 하지만, 사례가 많아지면 많아질수록 사

례를 검색하는데 드는 시간은 늘어난다. 이러한 문제점을 해결하기 위해 본 논문에서는 사례기저에 저장되어 있는 기존 발생 사례들의 각 속성 값을 수치화하고 이를 이용하여 트리구조로 사례를 관리하는 방식을 제안한다.
제안 알고리즘은 새로운 사례 발생시 기존 사례들을 이용하여 구축해놓은 트리를 이용하여 검색하는데 걸리는 시간을 줄일 수 있다. 또한 기존 사례들을 군집화하여 사례의 수를 줄여가며 사례의 정확도를 유지하는 클러스터링 기법과의 비교를 통해 사례의 수를 유지하면서 서비스 매칭의 적중률을 높일 수 있는 방법임을 확인하였다.

2. 기존 사례기반추론 방법

사례기반추론은 지식을 'IF-THEN'의 규칙 형식으로 기술하는 것이 곤란한 경우에 유효한 추론 방식으로 주목받고 있다. 규칙을 사용하여 추론하는 규칙기반 추론(rule-based reasoning)을 채용한 것에는 전문가 시스템이 있으며, 이 방식은 대상이 되는 문제에 관한 전문가와 상의하여 규칙을 추출하는 작업을 필요로 한다. 사례기반의 추론을 하려면 규칙을 작성하는 대신에 사례를 수집하여 데이터베이스에 저장하면 된다. 사례기반추론(case-based reasoning)을 실행하는데 문제가 되는 것은, 유사 사례를 찾기 위한 유사도의 정의와 얻어진 유사한 사례의 해답을 목적하는 문제의 해답으로 변환하기 위한 수정 방법인데, 범용적인 방법은 없기 때문에 개발하는 시스템에 따라 결정하지 않으면 안 된다.
이러한 사례 기반 추론 기법은 크게 4단계로 나누어지며, 각 과정은 다음과 같다. 현재 주어진 문제와 가장 일치하는 사례를 검색하는 과정(이하, 서비스 매칭), 선택된 가장 비슷한 사례의 해결방법을 이용하여 현재 문제를 풀어보는 과정(이하, 시뮬레이션), 문제가 해결되지 않을 경우 해결방법을 수정하여 새로운 해결방법을 모색하는 과정(이하, 어뎁테이션), 마지막으로 새로 발견한 해결법을 이후 새로운 문제 해결에 이용하기위한 저장과정으로 이루어져 있다. 하지만 이렇게 새로 발견한 해결법이 항상 최적의 해결법이라는 보장은 할 수 없다. 사례가 존재하지 않아 유사사례 해법을 수정하여 사용할 경우, 생성된 해법이 최적의 결과를 주지 않을 수 있기 때문에, 이러한 문제 해결을 위해 사례 기반 추론 과정에 최적화 과정을 추가될 수 있다[6].
사용자가 원하는 서비스의 생성에 여러 방법이 제안되었는데 그 중에 사례 기반 추론을 적용한 연구가 있다. 주로 웹서비스를 위한 서비스 생성에 사례 기반 추론을 적용한 연구들이다[7][8]. 하지만 사례기반추론의 경우, 사례기저에 쌓인 사례의 수가 늘어감에 따라 사례기반추론의 속도가 저하된다는 단점이 있다. 이를 극복하기 위한 방법으로 사례를 카테고리 별로 나눈 후 인덱싱을 통해 사례를 빨리 찾는 기법[9]과 사례기저에서 사례의 수를 줄이는 연구가 있다. 사례기저의 수를 줄이는 연구로 많은 연구자들이 클러스터링을 이용한 사례 관리 기법을 제안하였다. 주로 K-means 클러스터링 알고리즘[10]과 K-NN 클러스터링 알고리즘[11]을 이용하여 비슷한 사례끼리 그룹을 형성 후 유사한 사례 중에서 중요한 사례만 남기고 나머지 사례를 삭제하는 방식으로 접근하였다. 그리고 클러스터링 이후 Decision Forest등을 이용하여 사례에 대한 정보가 부족해도 비슷한 사례를 찾을 수 있는 연구가 있다[12]. 그러나 이러한 클러스터링을 적용한 방법들은 사례기저의 수를 줄임으로써 매칭시의 속도를 향상시키고 있기 때문에, 삭제된 사례가 다시 발생하게 되면 새로운 사례로 인식한다는 문제점을 가지고 있다. 서비스 컴포지션에 사례기반 추론 기법을 적용하기 위해 문제, 서비스 그리고 사례에 대한 정의가 필요하다. 본 논문에서는 기존 서비스 컴포지션 기법[6]과 동일한 정의를 이용하여 문제 해결에 적용하였다.

문제 = <시작 상태, 목표 상태>

사용자는 시작 상태에서 출발하여 목표 상태에 도달하는 것을 원한다. 문제를 해결하는 것은 시작 상태에서부터 목표 상태로 도달할 수 있는 방법을 찾는 것을 의미한다. 여기서 상태(State)는 사용자의 현재 위치와 같은 속성 값의 집합을 말한다. 예를 들어 사용자가 집에서 학교까지 가는 방법을 원한다고 하면, 상태는 위치라는 속성을 갖게 되고, 시작 상태는 집을 속성 값으로, 목표 상태는 학교를 속성 값으로 가지게 된다.
서비스는 자원(resource)을 이용하여 한 개 이상의 상태(state)를 변화시킬 수 있는 작업을 말하며, 모든 서비스는 선행상태(precondition)와 후행상태(postcondition)를 기본적으로 갖고 있다. 선행상태는 단일서비스를 수행하는데 있어서 만족해야하는 조건들의 집합이고, 후행상태는 단일서비스가 수행되고 난 후 변화하게 되는 속성 값의 집합이다. 단일서비스는 하나의 자원만을 이용하는 것을 말하며, 복합서비스는 단일 혹은 복합 서비스 조합으로 생성되는 서비스를 말한다. 복합서비스는 특정 목표를 달성할 수 있는 단일서비스의 조합이다.
문제를 해결하는 것은 이러한 서비스의 조합을 통해 해법을 완성하는 것을 의미한다. 물론 단일 서비스로 해결 가능한 문제도 존재 할 수 있다.

문제 해법 = <단일서비스1, 단일서비스2, ···>

마지막으로, 사례는 문제와 그에 따른 해결법의 집합으로 정의한다.

사례 = <문제, 문제 해법>

3. 사례기저 관리(Case Base Management) 방법 제안

사례기반추론은 사례기저에 쌓은 사례의 양이 늘어날수록 보다 정확한 해결법을 찾을 수 있지만, 역설적으로 많은 사례들과 현재 문제를 비교해야 하므로 가장 비슷한 사례를 골라내는 검색시간이 늘어나게 된다. 이러한 단점을 해결하기 위해 클러스터링을 이용한 사례기저 관

리 기법이 사용되고 있다. 하지만 클러스터링을 통해 사례기저를 축소시킴으로서 기존에 발생했던 사례의 손실이 발생할 수 있다. 그래서 클러스터링 기법이 아닌 각 속성 값을 수치로 표현하고, 그 변환된 수치를 이용하여 트리를 구축하는 방법을 제안한다.

3.1 사례기저 수치화

사례기저에 저장되는 사례는 문제와 문제 해결법의 쌍으로 이루어져 있다. 문제 해결법은 발생한 문제에 대해 서비스를 어떻게 제공하여 문제를 해결하였는지에 관한 내용이며, 문제는 실제 발생했던 상황을 해당 도메인에서 정의한 속성들의 속성 값으로 나타내고 있다. 서비스 컴포지션의 매칭은 새로 발생한 상황의 속성 값이 사례기저에 저장되어있는 사례와 일치하는 것이 있는지 찾아내는 과정이다. 따라서 실제 매칭에서 고려되는 부분은 속성 값, 즉 사례의 구성요소 중 문제의 비교이다. 본 논문에서 제안하는 방법 중 하나인 수치화는 속성 값을 수치로 나타내는 방법을 말한다.

수치화의 방법은 속성 값의 종류에 따라 적용 방법을 달리해야 하며 그 과정은 다음과 같다.

표 1. 속성 값의 수치 표현 예

속성	속성 1	속성 2	속성 3	속성 4	속성 5
수치	NO → 0 YES → 1	A → 0 C → 1 D → 2	am1:00 → 0 am1:30 → 1 am3:00 → 2 ...	...	...

표 1의 속성 1의 경우 이분된 값(Binary)을 가진다. 이 경우 두 값을 각각 0과 1로 수치화 하여 나타낼 수 있다. 또한 속성 2의 경우 순서형 값(Ordinal)을 가진다. 예를 들어 속성 값이 A, C, D의 순서로 나타나면 이를 각각 0, 1, 2로 수치화 한다. 이후 새로 발생한 사례에서 B라는 속성 값이 나타나면 B의 순서가 A와 C의 사이이기 때문에 C와 D의 수치화 된 값을 조정해 주어야 한다.(표 2) 마지막으로 속성 3은 연속형 값(Interval)을 갖는다. 이 경우 속성 값을 오름차순으로 나열한 후 차례로 수치화 하는 방법을 이용한다.(표 3)

표 2. 순서형 값의 수치화 예

초기 수치화 값		조정된 수치화 값
A → 0 C → 1 D → 2	B의 삽입 →	A → 0 B → 1 C → 2 D → 3

표 3. 연속형 값의 수치화 예

초기 상태		조정된 수치화 값
am1:30 pm7:00 am5:30 am1:00	오름차순 정렬 →	am1:00 → 0 am1:30 → 1 am5:30 → 2 pm7:00 → 3

수치화 된 값의 범위가 크게 차이가 나는 것은 매칭 과정에서 유사 사례를 찾아내는데 큰 영향을 미친다. 예를 들어 표 4의 속성 1은 문의 개폐여부를 나타내는 속성으로 0은 Open, 1은 Close로 나타내고, 속성 3은 시간을 나타내는 속성이라고 하면, 사례 1과 사례 2는 시간에서 차이를 보이고 있다. 반면 사례 1과 사례 3은 시간 차이는 작지만, 사례 1은 문이 Open되어있고 사례 3은 문이 Close되어 확실히 구분되는 사례이다. 그러나 특정 속성 값이 명확히 차이를 보이고 있음에도 이 사례들을 벡터 평면에 나타내면 사례 1의 유사 사례는 사례 3으로 선택되게 된다. 즉, 각 속성들의 수치화 한 값의 범위가 크게 차이가 나게 되면 특정 속성이 중요함에도 불구하고 반영되지 못하는 경우가 발생할 수 있다. 이러한 사례를 방지하기 위해 수치화 한 값을 정규화 시킨다. 제안하는 알고리즘에서의 정규화는 수치화 한 속성 값의 범위를 동일하게 조정하는 작업으로 볼 수 있다.

표 4. 정규화 예시

	속성 1(Binary)	속성 2(Binary)	속성 3(Interval)
사례1	0	1	30
사례2	0	1	50
사례3	1	1	32

↓

	속성 1(Binary)	속성 2(Binary)	속성 3(Interval)
사례1	0	1	0.6
사례2	0	1	1
사례3	1	1	0.64

3.2 트리 구성

사례기저를 트리구조로 관리하기 위해서 우선 트리의 루트노드를 정해야 한다. 본 논문은 트리의 루트 노드를 정하기 위해 다음과 같은 알고리즘을 제안한다.

① 사례기저에 저장되어 있는 모든 사례들의 속성 값을 수치화

② 수치화 한 값을 정규화

③ 루트 노드 속성 1의 값 $= \frac{\text{모든 사례의 속성 1의 값의 합}}{\text{사례의 수}}$

④ ③의 과정을 속성의 개수만큼 반복

①~④의 과정을 거치고 나면 루트 노드의 각 속성 값이 정해지게 된다. 이 속성 값들은 특정 사례가 어느 서브 트리로 이동해야 하는지 결정하는 인덱스(index)역할을 하게 된다. 본 논문에서는 이러한 루트노드를 제외한 다른 인덱스 역할을 하는 노드를 인덱스 노드라 정의하였다. 본격적인 연산에 앞서 각 속성에 대해 허용오차 값을 설정해야 한다. 허용오차는 특정 값에 대해 어느정도 범위까지는 비슷한 값으로 인정한다는 것을 나타내기 위한 수치로, 문제 도메인에 맞는 유사도 척도(similarity measure)에 대한 정의로 나타난다. 예를 들어 수치 값이 15이고 허용오차가 10이라면 5이상 25이하를 만족하는

수치이면 비슷한 값으로 간주한다. 루트 노드와 허용오차가 결정되면 다음 과정을 거쳐 사례를 분류한다.

① | 사례 1의 속성 1의 값 − 루트 노드의 속성 1의 값 |
② ①의 계산 결과와 속성 1의 허용오차 비교
  ⓐ ①의 결과가 허용오차보다 작거나 같을 경우 : 속성 1의 계산 결과 = ①의 계산 값
  ⓑ ①의 결과가 허용오차보다 클 경우 : 속성 1의 계산 결과 = NULL
③ 모든 속성에 대해 ①②의 과정 반복
④ 사례 1의 NULL을 제외한 모든 계산 결과 합산
  ⓐ ④의 결과가 0 미만 : 사례 1 → 왼쪽 서브트리
  ⓑ ④의 결과가 0 이상 : 사례 1 → 오른쪽 서브트리
⑤ 모든 사례에 대해 ①~④ 반복
⑥ 서브트리 내의 사례의 개수가 일정 개수 이하가 될 때 까지 ①~⑤ 반복

위의 알고리즘에서 특정 속성의 수치와 수치의 평균값이 많은 차이를 보이고 있음에도 차이 값을 그대로 적용할 경우, 이는 유사하지 않은 속성의 큰 차이 값 때문에 다른 유사한 속성들의 차이 값이 영향을 미치지 못하는 경우가 발생할 수 있기 때문이다. 이러한 경우를 방지하기 위해 허용오차를 적용한다. 허용오차를 기준으로 속성의 차이 값이 허용오차를 벗어나면 계산 과정에서 누락시켜 각 사례의 유사도를 보다 정확히 알아낼 수 있다.

표 5. 사례기저

	속성1	속성2	속성3	속성4
사례 1	0	1	0.3	0.5
사례 2	1	1	0.8	0.4
사례 3	1	0	0.3	0.7
사례 4	0	1	0.5	0.1
사례 5	0	1	0.1	0.8

표 5와 같은 사례기저에서 트리를 구축하는 과정을 살펴보자. 허용오차는 표 6과 같고, 서브트리 내의 사례가 2개 이하가 될 때까지 트리를 구축하는 것을 목표로 한다.

표 6. 허용오차

	속성1	속성2	속성3	속성4
허용오차	0.5	0.4	0.3	0.5

사례기저와 허용오차를 적용하여 제안알고리즘을 1회 수행하면 다음과 같은 계산 결과를 얻는다.(표 7)

표 7. 계산결과

	속성1	속성2	속성3	속성4	최종 계산값	서브 트리
사례 1	-0.4	0.2	-0.1	0	-0.3	왼쪽
사례 2	NULL	0.2	NULL	-0.1	0.1	오른쪽
사례 3	NULL	NULL	-0.1	0.2	0.1	오른쪽
사례 4	-0.4	0.2	0.1	-0.4	-0.5	왼쪽
사례 5	-0.4	0.2	-0.3	0.3	-0.2	왼쪽

표 7의 결과를 이용하여 구축된 트리의 구조는 그림 1과 같다.

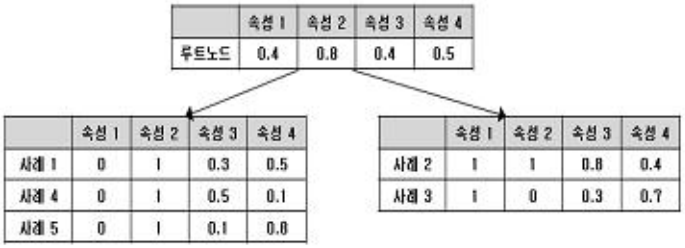

그림 1. 루트노드를 기준으로 분류된 서브트리

또한 최종 목표가 서브트리 내의 사례가 2개 이하가 되는 것이므로 알고리즘은 재귀적으로 실행되며, 실행이 완료된 후의 사례기저의 구조는 그림 2와 같다.

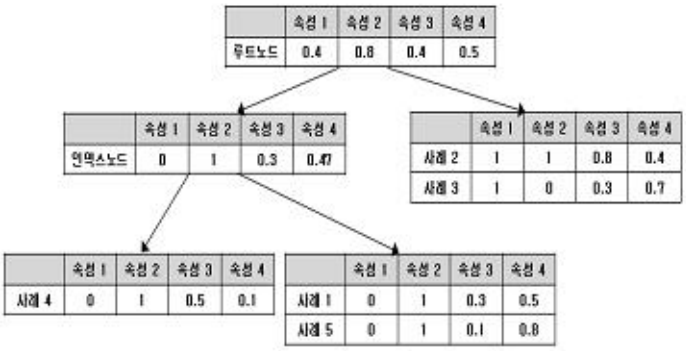

그림 2. 최종 완성된 사례기저의 트리구조

4. 실험 및 분석

본 논문에서 제안하는 알고리즘의 검증을 위해 서로 다른 3가지 클러스터링 기법을 이용한 서비스 조합과 비교하였다. 각 서비스 조합에 사용된 기법은 다음과 같다.

① 사례기저의 수를 줄이지 않는 클러스터링 기법(이하 C1)
② 사례기저의 수를 줄이는 클러스터링 기법[11](이하 C2)
③ 직경제한 클러스터링 기법[6](이하 C3)
④ 제안 알고리즘(이하 PA)

또한 실험을 위한 도메인으로 최단경로 탐색 서비스를 구현하였다. 최단경로 탐색 서비스는 현재 사용자의 위치(시작 위치)와 이동하고자 하는 목적지(목표 위치), 교통비와 시간을 입력받아 교통비를 초과하지 않으면서 제한시간 내에 목적지까지 가장 빠르게 갈 수 있는 경로를 결정해 주는 것이다. 실험의 편의를 위해 다음과 같이 가정하였다.

- 이동 가능한 장소는 그림 3에 원형으로 표시된 A~M까지 13개이며 각 장소를 위도와 경도로 수치화 하였다.

- 교통수단은 지하철과 버스만으로 제한하였으며, 각각의 이동경로는 그림 3과 같다.
- 지하철로 이동하는 경우 한 장소에서 바로 다음 장소까지 걸리는 시간은 180초, 비용은 거리와 상관없이 1000원으로 고정하였다.
- 버스를 타고 이동하는 경우, 한 장소에서 바로 다음 장소까지 걸리는 시간은 300초, 비용은 900원으로 고정하였다.
- 버스의 노선은 총 7개, 지하철의 노선은 3개로 설정하였다.

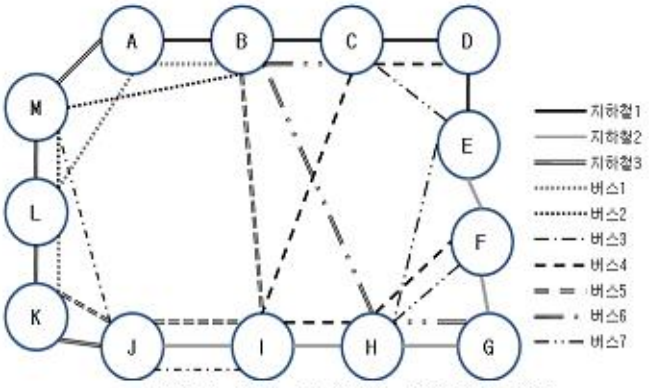

그림 3. 실험 도메인에 적용한 교통망

최단경로 탐색 프로그램에서 사용자가 해결하기 원하는 문제는 다음과 같이 구성된다.

<시작위치, 목표위치, 시간, 교통비>

- 시작 위치 및 목표 위치 : 수치화 한 장소번호
- 시간 : 사용자가 원하는 소요 시간
- 교통비 : 사용자가 이동에 사용할 금액

또한 사례기저에 저장되어있는 사례의 구성은 다음과 같다.

<시작위치, 목표위치, 시간, 교통비, 이용서비스>

- 시작 위치 및 목표 위치 : 수치화 한 장소번호
- 시간 : 소요 시간
- 교통비 : 사용한 금액
- 이용서비스 : 전체 복합서비스 동안 이용한 이동서비스(갈아탄 위치, 사용한 이동 수단).

위의 조건에 따라 사례기저로 사용할 임의의 1000개의 사례를 생성하여 저장하였다. 생성된 사례를 각각 클러스터링을 이용한 서비스 조합과 제안 알고리즘을 이용한 서비스 조합의 사례기저 관리방법에 해당하는 방식으로 사례기저를 구축하였다.

우선 사례기저의 수에 따른 매칭 속도와 매칭 성공률을 비교하였다. 비교할 4개의 알고리즘 중에서 사례기저의 수를 감소시키는 C2와 C3 기법은 각각 50개씩 사례를 줄이면서 실험을 수행하였으며, 사례기저의 수를 유지하는 C1과 PA의 경우, 저장된 사례의 수는 1000개로 고정하였다.

표 8. 평균 매칭 소요시간과 매칭 성공률

	측정 항목	C1	C2	C3	PA
Test 1	사례기저의 수	1000	950	950	1000
	매칭 성공률	30/30	26/30	27/30	30/30
	평균 매칭 소요시간(초)	1.412	0.867	0.321	0.325
	평균 문제 해결 시간(초)	1.412	1.351	1.117	0.325
Test 2	사례기저의 수	1000	900	900	1000
	매칭 성공률	30/30	24/30	25/30	30/30
	평균 매칭 소요시간(초)	1.401	0.842	0.311	0.321
	평균 문제 해결 시간(초)	1.401	2.445	1.645	0.321
Test 3	사례기저의 수	1000	850	850	1000
	매칭 성공률	30/30	21/30	22/30	30/30
	평균 매칭 소요시간(초)	1.415	0.830	0.309	0.322
	평균 문제 해결 시간(초)	1.415	3.256	2.213	0.322
Test 4	사례기저의 수	1000	800	800	1000
	매칭 성공률	30/30	18/30	20/30	30/30
	평균 매칭 소요시간(초)	1.408	0.815	0.297	0.317
	평균 문제 해결 시간(초)	1.408	4.046	2.963	0.317

C1번 기법의 경우, 사례기저의 수를 줄이지 않고 클러스터링을 하였기 때문에 비교해야하는 사례가 다른 클러스터링 기법보다 많아, 평균 매칭 소요시간이 가장 오래 걸리는 것을 확인할 수 있다. C2번 기법은 C1번 기법보다 비교할 사례의 수가 적기 때문에 매칭 소요시간이 단축되었다. C3와 PA의 경우, 매칭 소요시간은 C1, C2에 비해 크게 단축된 것을 확인할 수 있다. 또한 사례기저의 수를 줄이면서 매칭 속도를 향상시키는 C2, C3 기법의 경우, 삭제된 사례의 수가 증가할수록 매칭 소요시간은 감소하지만 매칭 성공률이 크게 떨어지게 된다. 반면 C1과 PA는 사례기저의 수를 1000개로 유지하고 있기 때문에 매칭 성공률은 100%를 보인다. 즉 제안 알고리즘 PA는 클러스터링 기법 중 매칭 소요시간이 가장 짧은 C3와 비슷한 속도를 보이면서도 사례기저의 수를 유지하고 있기 때문에 매칭 성공률이 높게 나타나는 것을 확인하였다. 매칭이 실패하면 발생한 문제에 대한 해결책을 찾기 위해 추가적인 과정(시뮬레이션, 어뎁테이션, 최적화)이 수행되어 답을 제시하게 된다. 매칭성공률이 낮을수록 추가 연산으로 인한 시간소모가 커지기 때문에 최종적인 문제 해결시간이 증가하게 된다. 따라서 사례기반 추론을 이용한 서비스 조합에서는 매칭 성공률이 높을수록 평균 문제 해결시간이 단축된다.

표 8의 결과에서 제안 알고리즘 PA는 매칭 소요시간이 짧고 매칭성공률이 높아 문제 해결에 소모되는 평균 시간이 클러스터링 기법 보다 단축된 것을 확인할 수 있다.

5. 결론

본 논문에서는 빠른 복합 서비스 제공을 위해 사례기반 추론을 이용한 서비스 컴포지션 기법을 제안했다. 하지만 사례의 수가 증가함에 따라 비슷한 사례를 찾는 시간이 오래 걸리는 문제가 발생하기 때문에 사례의 수를 줄이며 사례기저를 관리하는 클러스터링 기법이 주로 사용되고 있다. 그러나 사례기저의 수를 줄이는 것은 기존에 발생했던 사례일지라도 새로 발생한 사례로 인식하게 되어 문제해결에 부수적인 단계가 추가적으로 제공되어야 한다. 이로 인해 평균 문제 해결 시간이 느려지는 문제가 발생하게 된다. 그러나 본 논문에서는 사례기저의 수를 유지하면서 매칭 시에 클러스터링 기법과 동등한 속도를 내면서도 평균 문제풀이 시간을 줄일 수 있는 알고리즘을 제안하였다. 그리고 비교 실험을 분석하여 본 논문에서 제안하는 알고리즘이 기존의 클러스터링을 이용한 서비스 컴포지션 기법보다 더 나은 성능을 보여주었음을 확인했다.

그러나 제안한 알고리즘은 사례기저를 수치화 하는 부분에 있어서 속성 값이 수치화 가능한 도메인에만 적용이 가능하다는 단점이 있다. 따라서 도메인 제한적이 아닌, 일반화 된 사례기저 수치화를 할 수 있는 방법을 찾는 것이 향후 연구 과제이다.

참고문헌

[1] 김재홍 외 6명, "URC를 위한 명령 분석 및 서비스 계획 기술, 전자통신 동향 분석,", 전자통신동향분석, April, 2005

[2] D.S. Nau et al., "SHOP2 : An HTN Planning System, Journal of Artificial Intelligence Research,", Vol. 2870, pp. 195-210, 2003

[3] K. Erol et al., "UMCP : A Sound and Complete Planning Procedure for Hierarchical Task-Network Planning," Artificial Intelligence Planning Systems, pp. 249-254, June, 1994

[4] OWL-S Home Page, "http://www.daml.org/service/owl-s/1.1/overview", 2004

[5] 이재규 외 2명, "전문가시스템 : 원리와 개발, 법영사", 2004

[6] 박두경, "사례기반추론을 이용한 service composition 연구", 성균관대학교 석사학위 논문, 2007

[7] O.Graciela Fragoso Diaz et al., "Searching and Selecting Web Services using Case-based Reasoning," International Conference on Computational Science and Its Applications , Vol. 3983, pp. 50-57, 2006

[8] Bechaphon Limthanmaphon et al., "Web Service Composition with Case-based Reasoning," ADC, pp. 201-208, 2003

[9] B. W. Porter et al., "PROTOS : An experiment in knowledge acquisition for heuristic classification tasks," Proceedings of the 1st International Meeting on Advance in Learning, pp. 159-174, 1986

[10] N. Arshardi et al., "Data Mining for Case-Based Reasoning in High-Dimensional Biological Domain," IEEE Transactions on Knowledge and Data Engineering, Vol. 17, pp. 1127-1137, 2006

[11] Z. Li, Y. Liu, F. Li, S. Yang, "Case Base Maintenance based on Outlier Data Mining," Proceedings of the 4th International Conference in Machine Learning and Cybernetics, Vol. 5, pp. 2861- 2864, 2005

[12] Q. Yang et al,. "Enhancing the Effectiveness of Interactive Case-based Reasoning with Clustering and Decision Forest," Applied Intelligence, Vol. 14, pp. 49-64, 2001