Proceedings of KFIS Autumn Conference 2007 Vol. 17, No. 2.

사례 기반 추론을 이용한 서비스 컴포지션 속도향상 연구

A Study for Fast Service Composition with Case-based Reasoning

이승훈, 박두경, 김건수, 윤태복, 이지형

성균관대학교 컴퓨터공학과

reinblame@naver.com, {haderme, kkundi, tbyoon}@skku.edu, jhlee@ece.skku.ac.kr

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 빠른 응답이 필요한 실시간 상황인식 서비스에는 부적합 하다. 또한 사례기반 추론은 사례 기저에 쌓인 사례의 수가 늘어감에 따라 속도가 저하되는 단점이 있다. 이러한 단점을 최소화 하기 위하여 클러스터링 기법이 사용되고 있다. 본 논문은 사례기반 추론을 이용한다. 또한 사례 기저의 수를 유지하면서 사례 기저의 수치화 및 트리구조 관리를 이용하여 기존방법보다 빠른 서비스 컴포지션을 구현하는 방법을 제안한다. 그리고 기존의 서비스 컴포지션 기법과 비교 분석을 통하여 제안하는 기법의 유효함을 확인하였다.

Key Words : Service Composition, Case-based reasoning

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 방법으로 서비스 컴포지션[1] 이라는 기술이 개발되었다.

서비스 컴포지션은 하나의 서비스를 단일 서비스(primitive service)로 정의하고, 이러한 단일 서비스들 간의 연계를 고려해서 조합을 한 복합서비스(composite service)를 생성하여 사용자의 요구사항을 만족시키는 것을 목표로 한다. 서비스 컴포지션 기법으로는 Hierachical Task Network(HTN), Model Checking 등을 이용한 방법들이 제안되었다. 이 중 HTN은 Maryland 대학의 SHOP2[2] 기반의 웹 서비스 컴포지션 시스템으로 이는 계층적 업무 네트워크 (Hierarchical Task Network)[3] 기반의 AI Planning 기술을 적용하여 OWL-S[4]로 의미가 기술된 웹 서비스 컴포지션을 수행하는 것이다. 하지만 이런 기법들은 모두 탐색을 기반으로 하고 있어서 서비스의 수가 증가함에 따라 긴 탐색 시간을 갖게 된다. 유비쿼터스 환경에서는 상황인식 서비스의 대부분이 빠른 응답시간을 요구하기 때문에, 실시간으로 응답이 가능한 빠른 서비스 컴포지션 기법이 필요한데, 이러한 기법들은 상황인식 서비스의 요구사항을 만족시키지 못한다.

그래서 본 논문에서는 사례기반추론을 이용한 서비스 매칭 기법을 이용하여 사용자의 요구에 보다 빠르게 응답할 수 있는 방법을 제안한다. 사례기반 추론은 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]. 이는 지식을 'IF-THEN'의 규칙 형식으로 기술하는 것이 곤란한 경우에 유효한 추론방식으로 주목받고 있다.

서비스 컴포지션에 사례기반추론을 적용한 연구가 있다. 주로 웹서비스를 위한 서비스 컴포지션에 사례기반추론을 적용한 연구들이다[6][7]. 하지만 사례기반추론의 경우, 사례 기저에 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[8]과 사례기저에서 사례의 수를 줄이는 연구가 있다. 사례기저의 수를 줄이는 연구로 많은 연구자들이 클러스터링을 이용한 사례 관리 기법을 제안하였다. 주로 K-means 클러스터링 알고리즘[9]과 K-nn 클러스터링 알고리즘[10]을 이용하여 비슷한 사례끼리 그룹을 형성 후 유사한 사례 중에서 중요한 사례만 남기고 나머지 사례를 지우는 방식으로 접근하였다. 그리고 클러스터링 이후 Decision Forest[11]를 이용하여 사례에 대한 정보가 부족해도 비슷한 사례를 찾을 수 있는 연구가 있다.

사례기반추론의 정확도는 보유하고 있는 사례의 양에 비례한다. 하지만, 사례가 많아지면 많아질수록 사례를 검색하는데 드는 시간은 늘어난다. 이러한 딜레마를 해결하기 위해 사례기저의 각 속성 값을 수치화하고 이를 이용하여 트리구조로 사례를 관리함으로서 검색하는데 걸리는 시간을 줄이고 서비스 매칭의 적중률을 높일 수 있이는 방법을 제안한다.

2. 사례기반 추론을 이용한 서비스 컴포지션

문제를 입력받으면 서비스 컴포지션 모듈은 사례기저(Case Base)에서 가장 비슷한 사례를 찾는다. 완전 일치하는 사례를 찾으면 사례의 해결방법을 그대로 새로운 문제에 사용한다. 하지만 일치하는 사례가 없다면, 가장 비슷한 사례의 해결 방법을 적용시켜 본다. 적용이 불가능하면 주어진 문제에 맞게 해결방법을 수정한다. 그 후, 최적화 과정을 통해 새로운 해결방법이 이상이 없음을 파악한 뒤 해결법을 제공하고, 사례 기저에 현재 문제와 함께 저장한다.

2.1 용어(Terms)

서비스 컴포지션에 사례기반 추론 기법을 적용하기 위해 문제, 서비스 그리고 사례에 대해 정의하였다. 문제는 다음과 같이 정의된다.

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 상태는 사용자의 현재 위치와 같은 속성 값의 집합을 말한다. 예를 들어 사용자가 집에서 학교까지 가는 방법을 원한다고 하면, 시작 상태는 집을 속성으로 가지게 되고 목표 상태는 학교를 속성으로 갖게 된다. 서비스는 자원(resource)을 이용하여 한 개 이상의 상태(state)를 변화시킬 수 있는 작업을 말한다. 자원과 상태 정보는 Attribute-value 쌍으로 기술된다. 그리고 서비스는 단일서비스(Primitive service)와 복합서비스(Composite service) 두 종류가 있다. 단일서비스는 선행상태(precondition)와 후행상태(postcondition)를 기본적으로 갖고 있다. 선행상태는 단일서비스를 수행하는데 있어서 만족해야하는 조건들의 집합이고, 후행상태는 단일서비스가 수행됨으로서 변화하는 정보의 집합이다. 그리고 복합서비스는 서비스 컴포지션의 결과로 생성되는 서비스를 말한다. 복합서비스는 특정 목표를 달성할 수 있는 단일서비스의 조합이다.

문제를 해결하는 것은 이러한 단일 서비스의 조합을 통해 복합서비스를 완성하는 것을 의미한다. 물론 단일서비스로 해결 가능한 문제도 존재 할 수 있다.

문제 해결 = <단일서비스1, 단일서비스2, …>

마지막으로, 본 논문에서 제안하는 모델에서 사례는 문제와 그에 따른 해결법의 집합으로 정의한다.

사례 = <문제, 문제 해결>

2.2 사례 기저 관리(Case Base Maintenance)

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 관리 기법이 있다. 하지만 클러스터링을 통해 사례기저를 축소시킴으로서 기존에 발생했던 사례의 손실이 일어날 수 있다. 그래서 클러스터링이 아닌 각 속성값을 수치로 표현한다고 했을 때, 그 수치를 이용하여 트리를 구축하는 방법을 제안한다.

표 1. 속성 값의 수치 표현 예

속성	속성1	속성2	속성3	속성4	속성5
수치	0 → 0 1 → 1	0 → 0 1 → 1 NULL → 2	am1:00 → 0 am1:30 → 1 am3:00 → 2 …	…	…

속성 값을 수치로 표현할 때 가장 높은 자리수가 나오는 속성의 자리수와 동일한 자리수를 모든 속성이 가질 수 있도록 수치로 표현한다.

표 2. 자리수를 맞춘 속성 수치 값

	속성1	속성2	속성3	속성4	속성5
사례1	0	0	78	80	0
사례2	100	50	92	70	0
사례3	0	100	12	90	100

모든 속성의 값을 수치로 표현한 후 각 속성의 평균값을 구한다. 평균값은 전체사례에서 해당 속성 값을 모두 더한 후 전체 사례 수로 나눈 값이다. 이 평균값이 트리의 루트가 된다. 또한 각 속성에 대해 허용오차 값을 설정한다. 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.

이후 다음 식을 이용하여 사례를 분류한다.

IF(| 속성1의 수치 - 속성1의 수치 평균 |)
<= 허용오차
THEN 속성1의 수치차이 = 속성1의 수치 - 속성1의 수치 평균
ELSE 속성1의 수치차이 = NULL

이 식으로 한 사례의 모든 속성에 대해 수치 차이를 구하고 난 후 그 값을 전부 더한다. 만약 그 값이 음수일 경우 루트의 왼쪽으로 이동되며 양수일 경우 오른쪽으로 이동된다. 위의 과정을 통해 루트와 왼쪽, 오른쪽으로 사례들이 나누어지면 왼쪽에 있는 사례들만으로 평균값을 구하는 과정부터 반복한다. 반복은 나누어진 사례의 개수가 일정 개수 이하가 될 때 까지 이루어진다. 이 과정을 통하여 인덱스가 평균값으로 이루어지며 리프노드는 사례들의 리스트로 이루어지는 트리를 만들 수 있다.

2.3 매칭(Matching)

사례기반 추론 Retrieve에 해당하는 과정으로 주어진 문제의 속성 값을 사례기저에 들어 있는 사례의 속성값과 비교하는 과정이다. 비교를 위해서는 문제 도메인에 맞는 유사도 척도 (similarity measure)에 대한 정의가 필요하다. 정의된 유사도 척도와 주어진 문제의 속성값을 이용하여 사례기저에 구축되어있는 트리에서 비교과정을 거친다. 2.2에 나타난 식을 이용하여 평균값과의 차이로 이동방향을 정하고 최종 리프노드에 도달 시에 해당 리스트에 들어있는 기존 사례와 매칭을 한다. 매칭 결과 현재 문제와 일치하는 사례 혹은 가장 비슷한 사례가 선정되며, 선정된 사례가 일치하는 사례일 경우 사례의 해결 방법이 주어진 문제의 해결방법으로 그대로 사용되고, 선정된 사례가 가장 비슷한 사례일 경우 다음 과정인 시뮬레이션 단계로 넘어가게 된다.

2.4 시뮬레이션(Simulation)

시뮬레이션은 사례기반추론 Reuse 단계에 해당한다. 이 과정을 통해 가장 유사한 사례의 해결 방법이 현재 문제에 그대로 적용가능한지 알 수 있다. 문제의 현재 상태에서 사례의 해결방법인 복합서비스를 적용하여 목표 상태에 도달할 수 있는지 검증하는 과정이며, 시뮬레이션 결과로 사례의 해결법이 현재 문제에도 적용가능하면 현재 문제에 대한 해결법으로 그대로 사용하고 그렇지 않으면 다음 과정인 어뎁테이션 단계로 넘어가게 된다.

2.5 어뎁테이션(Adaptation)

주어진 문제와 가장 비슷한 사례의 해결법이 문제에 직접 적용이 불가능 할 때 어뎁테이션 단계에서 가장 비슷한 사례의 해결법을 주어진 문제에 맞게 수정하는 작업을 한다. 먼저 가장 비슷한 사례의 해결법을 순차적으로 현재 상태에서부터 적용시킨다. 그 후 중간에 조건을 만족시키지 못하는 상태에 도달하게 되면 바로 직전상태로 복귀 후 목표상태로 도달할 수 있는 사례를 매칭, 시뮬레이션 과정을 통해 재귀적으로 찾는다. 사례를 찾은 경우 중간까지 만족한 서비스와 새로 발견한 서비스를 조합하여 새로운 조합서비스를 생성하고, 찾지 못한 경우 만족되지 못한 부분부터 탐색과정을 이용하여 새로운 조합 서비스를 생성한다. 이렇게 새롭게 생성된 해결법은 마지막 단계인 최적화 과정으로 넘어간다.

2.6 최적화(Optimization)

사례기반추론의 특징상 비슷한 사례의 해결법을 새로운 문제의 해결에 그대로 적용하는데 오류는 발생하지 않지만, 자원의 낭비가 발생할 수도 있다.

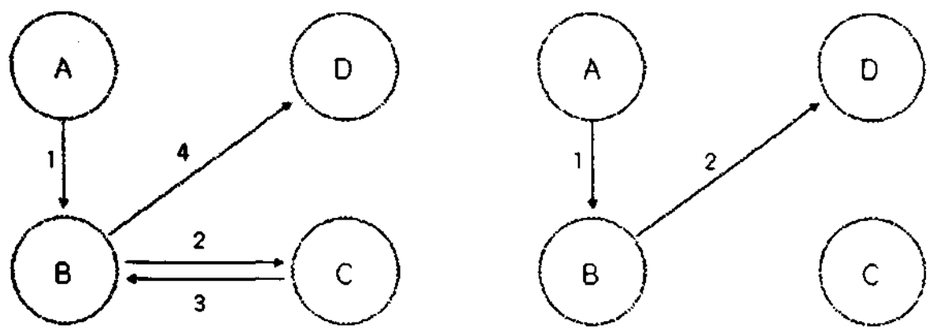

그림 1. 최적화의 예

그림 1과 같이 A에서 출발하여 D에 도착하는 것이 목표인 문제가 있다고 하자. 그림 1의 좌측 예는 A에서 B를 거쳐 C로 이동하였다가 다시 B로 돌아오는 불필요한 행동을 하게 된다. 하지만 우측 예와 같이 A에서 B를 거쳐 바로 D에 도착하는 경로를 이용하면 불필요한 자원의 낭비를 막을 수 있다. 이처럼 최적화 단계에서는 최종적으로 결정된 해결법의 오류뿐만 아니라 해결법으로 결정된 서비스 컴포지션에 불필요한 서비스가 없는지 검사하고 수정하는 일을 한다.

3. 실험 및 분석

본 논문에서 제안하는 알고리즘의 검증을 위해 클러스터링을 통한 서비스 컴포지션과 비교하였다. 또한 비교하기 위한 도메인으로 최단경로 탐색 서비스를 구현하였다. 최단경로 탐색 서비스는 현재 사용자의 위치와 목적지, 교통비와 시간을 입력받아 교통비를 초과하지 않으면서 제한시간 내에 목적지까지 가장 빠르게 갈 수 있는 경로를 결정해주는 프로그램이다. 실험의 편의를 위해 다음과 같이 가정하였다.

- 이동 가능한 장소는 10개이며 각 장소에 1~10까지의 번호를 배정하였다.
- 교통수단은 버스와 지하철을 사용하며, 버스와 지하철의 이동 경로는 임의로 설정하였다.
- 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 2개로 설정하였다.

최단경로 탐색 프로그램에서 사용자가 해결하기 원하는 문제는 다음과 같이 구성된다.

<시작위치, 목표위치, 시간, 교통비, 중요요소>

- 시작 위치 및 목표 위치 : 1~10의 장소번호
- 시간 : 사용자가 원하는 소요 시간
- 교통비 : 사용자가 이동에 사용할 금액
- 중요요소 : 사용자가 시간과 교통비 중 우선시할 사항. 교통비 우선시 교통비는 초과하지 않으면서 시간은 약간의 초과를 허용, 시간 우선시는 반대.

또한 사례 기저에 저장되어있는 사례의 구성은 다음과 같다.

<시작위치, 목표위치, 시간, 교통비, 이용서비스>

- 시작 위치 및 목표 위치 : 1~10의 장소번호
- 시간 : 소요 시간
- 교통비 : 사용한 금액
- 이용서비스 : 전체 복합서비스 동안 이용한 이동서비스(갈아탄 위치).

위의 조건에 따라 사례기저에 200개의 사례를 생성하여 저장하고 트리를 구축하였다.

```
leap node number 26's data
3 5 180 2900
3 1 180 2800
3 3 180 3900
3 5 180 2900
3 5 180 3800
3 3 180 2800
3 5 180 3800
3 3 180 3800
3 3 180 3800
```

그림 2. 구축된 트리의 26번 노드 내용

트리는 그림 2와 같이 비슷한 사례가 한 노드에 모이는 형태를 보인다. 그림 2의 첫 번째 열은 시작위치를 나타내고 두 번째 열은 도착위치를 나타낸다. 세 번째 열과 네 번째 열은 각각 시간과 사용한 교통비를 나타낸다. 동일한 200개의 사례를 클러스터링을 통해 172개로 줄이고 200개의 사례 중 20개를 임의로 선택하여 사례기반 추론을 하였다. 우선 사례기저의 수를 줄인 클러스터링 기법을 사용했을 때와 사례기저의 수를 유지하면서 트리를 구축했을 때에 유사사례를 찾아내는 매칭에 걸리는 시간을 비교해 보았다.

표 3. 매칭 결과

(단위 : 초)

문제 번호	클러스터링	제안 알고리즘
1	0.189	0.191
2	0.187	0.193
3	0.190	0.195
4	0.187	0.190
5	0.195	0.192
6	0.190	0.187
7	0.191	0.188
8	0.186	0.187
9	-	0.201
10	0.194	0.195
11	0.189	0.187
12	-	0.190
13	0.191	0.189
14	0.187	0.192
15	0.190	0.194
16	0.192	0.195
17	0.190	0.195
18	0.189	0.191
19	0.194	0.192
20	0.189	0.196
평균 소요시간	0.190	0.192
매칭 성공률	18/20	20/20

클러스터링에서 줄어든 28개중 2개가 임의로 선택한 20개에 속해 매칭 성공률이 제안 알고리즘에 떨어지고 있다. 또한 매칭에 걸린 시간은 거의 차이가 없는 것을 확인할 수 있다.

두번째 실험은 한 개의 상태 값을 임의로 변경하였을 경우 걸린 시간과 매칭 성공률을 측정해보았다. 매칭이 실패하였을 경우, 문제를 해결하기 위한 방법을 찾아내는데 걸리는 시간을 측정하였다.

표 4. 상태 값 임의 변경 시

(단위 : 초)

문제 번호	클러스터링	제안 알고리즘
1	0.191	0.193
2	0.195	0.192
3	0.874	0.195
4	0.201	0.195
5	0.955	0.961
6	1.026	1.031
7	0.187	0.190
8	0.942	0.187
9	0.191	0.192
10	0.190	0.193
11	0.886	0.881
12	0.195	0.192
13	0.189	0.195
14	0.191	0.189
15	0.198	0.192
16	0.197	0.194
17	0.949	0.952
18	0.188	0.190
19	0.190	0.194
20	0.197	0.193
평균 소요시간	0.416	0.345
매칭 성공률	14/20	16/20

제안한 알고리즘이 매칭 성공률이 높아 문제를 해결하는데 걸린 평균 소요시간이 단축됨을 볼 수 있다. 사례기저의 수를 줄이고 군집화를 통해 속도를 향상시키는 클러스터링 기법과 수치화를 통한 트리구조를 이용하는 제안 알고리즘이 매칭 시에 걸리는 시간이 거의 유사하지만 문제를 해결하는데 걸리는 시간은 제안 알고리즘에서 향상된 것을 확인할 수 있다.

4. 결 론

본 논문에서는 빠른 복합 서비스 제공을 위해 사례기반추론을 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 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.

참 고 문 헌

[1] 김재홍 외 6명, "URC를 위한 명령 분석 및 서비스 계획 기술, 전자통신 동향 분석," 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," AIPS, 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] O.Graciela Fragoso Diaz et al., "Searching and Selecting Web Services using Case-based Reasoning," ICCSA, Vol. 3983, pp. 50-57, 2006

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

[8] 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

[9] 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

[10] 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

[11] 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