워크플로우 분할에 기반한 복합 웹 서비스의 빠른 선택

(Fast Selection of Composite Web Services Based on Workflow Partition)

장 재 호 † 신 동 훈 † 이 경 호 ††
(Jae-Ho Jang) (Dong-Hoon Shin) (Kyong-Ho Lee)

요 약 복합 웹 서비스 선택은 서비스의 기능만을 명시한 추상 워크플로우에 바인딩 정보를 추가하여 주어진 QoS 요구사항을 만족하는 복합 웹 서비스를 구성하는 과정이다. 웹 서비스의 급격한 증가와 QoS가 동적으로 변하는 웹 서비스 환경을 고려할 때 주어진 QoS 요구사항을 만족하는 복합 웹 서비스의 빠른 선택이 중요하다. 본 논문은 워크플로우 분할에 기반하여 복합 웹 서비스를 빠르게 선택하는 방법을 제안한다. 제안된 방법은 추상 워크플로우를 두 개의 서브 워크플로우로 분할하여 선택 과정에서 고려되는 후보 서비스의 수를 줄인다. 분할된 워크플로우의 QoS 요구사항은 원래의 QoS 요구사항을 분해함으로서 생성한다. QoS 요구사항의 분해는 휴리스틱에 기반하기 때문에 워크플로우 분할 시 적절한 복합 웹 서비스를 선택하지 못할 가능성이 있다. 이러한 실패를 줄이기 위해 주어진 QoS 요구사항의 제약정도를 정의하고 적절한 제약정도를 가지는 요구사항에 한해 워크플로우를 분할한다. 서비스 선택은 mixed integer linear programming을 이용하여 해결한다. 실험 결과 제안된 워크플로우 분할 방법은 99% 이상의 성공률을 보였다. 특히 워크플로우 분할 시 모든 경우에 대해서 복합 웹 서비스를 보다 빠르게 선택하였으며 이때 선택된 복합 웹 서비스의 QoS는 최적 서비스와 5%미만의 차이를 보였다.

키워드 : 웹 서비스 선택, QoS, 워크플로우 분할, mixed integer linear programming

Abstract Executable composite Web services are selected by binding a given abstract workflow with the specific Web services that satisfy given QoS requirements. Considering the rapidly increasing number of Web services and their highly dynamic QoS environment, the fast selection of composite services is important. This paper presents a method for quality driven comosite Web services selection based on a workflow partition strategy. The proposed method partitions an abstract workflow into two sub-workflows to decrease the number of candidate services that should be considered. The QoS requirement is also decomposed for each partitioned workflow. Since the decomposition of a QoS requirement is based on heuristics, the selection might fail to find composite Web services. To avoid such a failure, the tightness of a QoS requirement is defined and a workflow is determined to be partitioned according to the tightness. A mixed integer linear programming is utilized for the efficient service selection. Experimental results show that the success rate of partitioning is above 99%. Particularly, the proposed method performs faster and selects composite services whose qualities are not significantly different (less than 5%) from the optimal one.

Key words : Web services selection, QoS, workflow partition, mixed integer linear programming

1. 서 론

웹 서비스(Web services)는 HTTP와 SOAP과 같은 표준 프로토콜을 기반으로 접근 가능한 소프트웨어 컴포넌트로서 이종 플랫폼 간의 상호운용성을 지원한다. 최근들어 SOA(Service-Oriented Architecture)의 보급과 더불어 고수준의 비즈니스 프로세스에 대한 요구가 증가함에 따라 단일 웹 서비스를 조합하여 복합 웹 서

· 본 지식재산권은 정통부 및 정보통신연구진흥원의 지원을 받아 수행된 연구결과임(06-정책-118, 적응형 모바일 웹 서비스 기술)
† 학생회원 : 연세대학교 컴퓨터과학
jhjang@icl.yonsei.ac.kr
dhshin@icl.yonsei.ac.kr
†† 종신회원 : 연세대학교 컴퓨터과학 교수
khlee@cs.yonsei.ac.kr
논문접수 : 2006년 4월 5일
심사완료 : 2007년 3월 14일

비스(composite Web services)를 구성하는 방법에 대한 관심이 높아지고 있다.

복합 웹 서비스는 서로 상호작용하는 다수의 웹 서비스들로 구성되어지며 단일 웹 서비스로는 제공하기 힘든 복잡한 기능을 제공한다. 일반적으로 복합 웹 서비스의 비즈니스 로직은 추상 워크플로우(abstract workflow)로 기술된다. 추상 워크플로우는 비즈니스 프로세스를 구성하는 기능적 모듈에 대응하는 태스크(task)와 이들 간의 제어흐름을 나타내는 트랜지션(transition)으로 구성된다. 복합 웹 서비스 선택은 추상 워크플로우의 각 태스크에 웹 서비스를 바인딩하여 실행 가능한 워크플로우(executable workflow)를 구성하는 과정이다. 한편 동일한 기능을 제공하지만 QoS는 서로 다른 다수의 후보 서비스들이 존재하기 때문에 복합 웹 서비스의 QoS는 선택된 웹 서비스에 따라 달라질 수 있다. 따라서 서비스 선택을 통해 사용자가 요구하는 특정 QoS를 만족하는 복합 웹 서비스를 구성하는 것은 매우 중요하다. 그림 1은 복합 웹 서비스의 선택 과정을 보여준다.

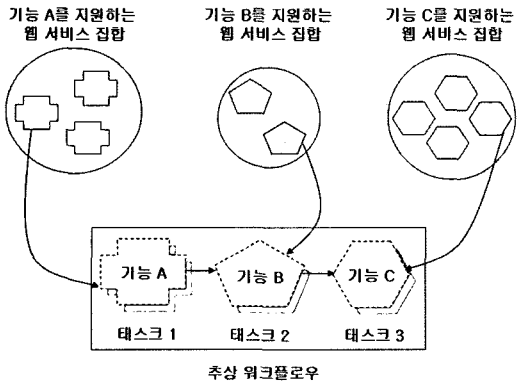


그림 1 복합 웹 서비스의 선택 과정

일반적으로 QoS 요구사항은 하나 이상의 품질요소에 관한 제약조건(constraint)들로 구성된다. 이때 제약조건은 복합 웹 서비스를 구성하는 단일 웹 서비스에 대한 로컬 제약조건이거나 복합 웹 서비스 전체에 대한 글로벌 제약조건일 수 있다. 글로벌 제약조건은 복합 웹 서비스의 품질을 단일 웹 서비스처럼 쉽게 표현할 수 있다는 장점을 가지는 반면 서비스 선택과정을 더욱 복잡하게 만든다. 다수의 품질요소에 대한 최적 서비스를 선택하는 과정은 NP-Complete 및 NP-Hard 문제와 같이 선형시간에 해결이 불가능한 조합 최적화(combinatorial optimization)[1] 문제이다. 기존의 서비스 선택에 관한 몇몇 접근법은 복잡도를 낮추기 위해 각 태스크에 대한 후보 서비스를 개별적으로 고려한다. 하지만 이 경우 글로벌 제약 조건을 고려할 수 없다는 제약이 따른다. 따라서 복합 웹 서비스 선택에 관한 기존 연구의 대부분은 글로벌 제약 조건을 만족하는 최적 서비스를 선택하는데 초점을 맞추고 있다. 한편 웹 서비스의 수가 빠르게 증가하고 QoS가 동적으로 변하는 웹 서비스 환경을 고려할 때 주어진 글로벌 제약조건을 만족하면서 적절한 QoS를 가지는 복합 웹 서비스를 빠르게 선택할 수 있는 방법이 필요하다.

현재 웹 서비스는 이용은 EAI(enterprise application integration)과 같은 조직 내 프로세스의 통합에 주로 이용되고 있다. 이는 웹 서비스가 성능 및 상호 운용성에 있어서 아직 시장의 요구사항을 반영 하고 있지 못함에 기인한다. 하지만 향후 웹 서비스 환경은 관련 표준 기술의 확립과 기술적 발전에 따라 다수의 조직이 서로 상호작용 하는 개방된 환경으로 옮겨 갈 것으로 기대된다. 앞서 언급한 바와 같이 다수의 서비스들이 상존하는 환경에서는 어떠한 서비스를 이용하여 비즈니스 프로세스를 구현할 것인지에 대한 고려가 필수적이다.

본 논문은 워크플로우의 구조, QoS 요구사항, 그리고 후보 웹 서비스의 QoS를 고려하여 워크플로우를 분할하여 복합 웹 서비스를 빠르게 선택할 수 있는 방법을 제안한다. 제안된 방법은 다수의 글로벌 제약을 고려한다. 한편 제안된 방법은 휴리스틱에 기반하기 때문에 워크플로우 분할 시 적절한 복합 웹 서비스를 선택하지 못할 가능성이 있다. 따라서 서비스 선택 실패를 줄이기 위해 주어진 QoS 요구사항의 제약정도를 계산하여 워크플로우의 분할 여부를 적절히 판단한다.

제안된 복합 웹 서비스의 선택 과정은 QoS 요구사항의 제약정도 계산, 워크플로우 분할, 그리고 서비스 선택의 세 단계로 이루어진다. 첫 번째 단계에서는 QoS 요구사항의 제약정도를 계산하여 워크플로우의 분할 여부를 결정한다. 두 번째 단계에서는 주어진 추상 워크플로우를 분할한 후, QoS 요구사항을 분해하여 분할된 추상 워크플로우의 QoS 요구사항을 생성한다. 마지막으로 MILP(Mixed Integer Linear Programming)[2]을 사용하여 복합 웹 서비스를 선택한다.

실험을 통해 제안된 방법의 성공률, 선택 속도, 그리고 선택된 복합 서비스의 품질의 세 가지 측면에서 성능을 평가했다. 실험 결과, 제안된 워크플로우 분할 방법은 99% 이상의 성공률을 보였다. 특히 워크플로우 분할 시 모든 경우에 대해서 제안된 방법이 복합 웹 서비스를 보다 빠르게 선택하였으며 이때 선택된 복합 웹 서비스의 QoS는 최적 서비스와 5%미만의 차이를 보였다.

본 논문의 구성은 다음과 같다. 2절은 복합 웹 서비스의 선택에 관한 기존 연구 결과를 간략히 기술한다. 3절은 제안된 복합 웹 서비스 선택 방법을 요구사항의 제약정도 계산, 워크플로우 분할, 그리고 서비스 선택의

세 단계로 나누어 자세히 기술한다. 4절은 실험을 통해 제안된 방법의 성능을 평가한다. 끝으로 5절은 결론 및 향후 연구 방향에 대하여 기술한다.

2. 관련 연구

웹 서비스의 QoS에 관한 기존 연구는 품질모델[3-5], 단일 웹 서비스 선택[6-8], 그리고 복합 웹 서비스 선택[9-17]으로 구분할 수 있다. 특히 복합 웹 서비스의 선택과 관련한 최근 연구의 대부분은 다수의 글로벌 제약조건을 대상으로 한다. 표 1은 QoS 기반 복합 웹 서비스의 선택에 관한 기존 연구의 특징을 요약한 것이다.

Cardoso 등[16]은 단일 서비스의 QoS로부터 워크플로우를 구성하는 순차, 병렬, 조건, 루프, 그리고 고장감내(fault-tolerant) 구조의 QoS를 계산하는 방법을 제안한다. 제안된 방법은 워크플로우의 중첩된 제어 구조를 가장 안쪽부터 재귀적으로 환원(reduction)하여 전체 프로세스의 서비스 품질을 갖는 단일 태스크를 생성한다. 그러나 서비스 선택을 자동화하기 위한 노력은 이루어지지 않았다.

GrØnmo와 Jaeger[14]는 컴포지션 패턴(composition pattern)에 기반하여 복합 웹 서비스를 선택한다. 따라서 전체 워크플로우를 고려하는 다른 방법들보다 서비스 선택 문제의 크기가 작은 반면 글로벌 제약조건을 고려하지 못하는 제약을 갖는다.

Zeng 등[17]은 QoS를 고려하는 복합 웹 서비스 선택을 위한 미들웨어를 제안한다. 특히 복합 웹 서비스 선택 문제를 ILP(Integer Linear Programming) 문제로 공식화한다. 특히 신뢰성과 같은 곱셈기반의 품질 모델을 ILP에 적용하기 위해 로그함수를 사용한다. 조건 분기 구조의 경우 모든 개별적인 분기를 독립적으로 해결한다. 따라서 선택에 소요되는 비용이 분기의 개수에 비례하여 증가하며 분기를 다시 병합할 경우 글로벌 제약조건을 위반할 가능성이 있다. ILP에 기반한 또 다른 방법으로 Gao[9] 등은 서비스 사이의 충돌이 발생할 수 있는 상황을 고려한다. Yu와 Lin[15]은 end-to-end 지연시간에 대한 제약조건을 가지는 서비스 선택 문제를 linear programming을 이용하여 효율적으로 해결할 수 있는 방법을 제안한다. 단일 제약조건만을 고려하는 경우 매우 빠르게 동작하지만 다중 제약조건에 대하여 적용할 수 없다.

Canfora 등[13]은 유전자 알고리즘(genetic algorithm)을 이용한 복합 웹 서비스 선택 방법을 제안한다. 유전자 알고리즘은 조합 최적화 분야에서 근사 최적해(approximate optimal solution)를 구하기 위한 대표적인 방법이다. 유전자 알고리즘에 사용된 염색체는 워크플로우의 각 태스크를 표현하며 각 유전자는 해당 태스크에 할당된 웹 서비스를 나타낸다. 한편 제약 사항을 고려하기 위해 적합도 함수에 제약 사항을 위한 기법을 적용한다. Liu 등[11]은 복합 웹 서비스를 표현하는 SCG(Service Composition Graph)를 생성한 뒤 유전자 알고리즘을 이용해 SCG상의 최적의 경로를 찾는다. 이 방법은 제약사항으로 평판도(reputation)와 신뢰성만을 고려한다.

Yang 등[12]은 TCNN(Transiently Chaotic Neural Networks)을 이용하여 QoS와 웹 서비스의 인터페이스에 대한 매칭정도를 동시에 고려하여 복합 웹 서비스를 선택한다. 병렬적으로 수행되는 서비스의 시간을 순차구조와 동일하게 처리한다는 문제점이 있으며 제약사항을 고려하지 않는다. Chen 등[9]은 가능한 모든 복합 웹 서비스의 품질을 2차원 배열로 표현한 뒤 서비스에 대한 가중합(weighted sum)의 결과를 이용하여 서비스를 선택한다.

관련연구 [9,17]에서 사용한 ILP와 관련 연구 [11,13]에서 사용한 유전자 알고리즘은 조합 최적화 문제를 효율적으로 해결하기 위해 일반적으로 사용되는 방법이다. ILP는 목적 함수(objective function)가 선형이어야 하며 변수의 개수에 대하여 기하급수적인 시간복잡도를 갖는다. ILP는 실제 조합 최적화를 비롯한 다양한 최적화 문제에 널리 적용되고 있으며 적절한 크기의 문제의 경우 빠른 시간 내에 결과를 얻을 수 있다. 유전자 알고

표 1 QoS 기반 복합 웹 서비스 선택에 관한 연구

저자	연도	특징
Gao 등 [9]	2006	서비스 사이의 충돌을 고려할 수 있는 방법을 제안
Chen 등 [10]	2006	Multi-Object Programming 에 기반하여 복합 웹 서비스 선택
Liu 등 [11]	2005	유전자 알고리즘을 이용하여 웹 서비스를 선택
Yang 등 [12]	2005	Transiently chaotic neural network를 이용하여 복합 웹 서비스를 선택
Canfora 등 [13]	2005	제약조건을 처리하는 적합도 함수를 사용하는 유전자 알고리즘에 기반
GrØnmo와 Jaeger [14]	2005	컴포지션 패턴을 기반으로 복합 웹 서비스를 선택
Yu와 Lin [15]	2004	지연시간에 대한 제약조건을 가지는 서비스 선택 문제의 효율적인 해결 방법을 제안
Cardoso 등 [16]	2004	워크플로우의 QoS에 대한 모델을 제안
Zeng 등 [17]	2003	integer linear programming 기반으로 복합 웹 서비스를 선택하는 미들웨어를 제안

리즘은 조합 최적화 방법의 근사 최적해를 효율적으로 구하기 위한 방법으로 지금까지 많은 연구가 이루어져 왔으며 실제 적용에 있어서도 주목할 만한 성과를 보여주었다. 하지만 복합 웹 서비스 선택에의 적용에 있어서는 아직 초기 단계에 머물러있다.

한편 QoS 요구사항, 워크플로우의 구조, 그리고 후보 서비스의 QoS는 실제 웹 서비스 선택 과정에 중요한 영향을 미친다[14]. 특히 전체 후보 서비스 집합 크기는 선택 속도에 있어서 직접적인 영향을 미치는 인자이다. 위에서 언급한 기존 연구들은 주로 최적 서비스를 찾는 방법에 집중되어 있으며 선택에 걸리는 시간을 줄이기 위한 노력은 부족한 실정이다. 본 논문은 워크플로우 분할을 통해 선택 문제를 나누어 해결함으로서 선택 속도를 향상시킬 수 있는 방법을 제안한다. 제안된 분할 방법은 선택 과정과 독립적으로 이루어지기 때문에 어떠한 선택 방법에도 적용가능하다는 장점을 갖는다.

3. 제안된 복합 웹 서비스 선택 방법

제안된 방법은 그림 2와 같이 추상 워크플로우와 QoS 요구사항을 입력으로 받아 사용자 요구사항을 만족하는 복합 웹 서비스를 선택한다. 선택 과정은 크게 QoS 요구사항의 제약정도 계산, 워크플로우 분할, 그리고 MILP 기반의 서비스 선택의 세 단계로 구성된다.

3.1 QoS 요구사항의 제약정도 계산

워크플로우의 분할에 앞서 주어진 QoS 요구사항의 제약정도를 계산한다. 사용자가 높은 수준의 QoS를 요구할수록 주어진 요구사항을 만족하는 복합 웹 서비스의 개수는 감소하며 워크플로우 분할시 적절한 복합 서비스를 찾지 못할 가능성은 보다 높아진다. 따라서 제약정도는 주어진 QoS 요구사항과 후보 서비스들로 구현 가능한 최고 품질 간의 차이로 계산된다. 전술한 바와 같이, 제안된 방법은 사용자가 요구한 QoS의 제약정도에 따라 워크플로우의 분할 여부를 결정한다. 워크플로우는 주어진 요구사항의 제약정도가 임계값 $TH_{tightness}$ 보다 작은 경우에만 분할되며 이때 임계값은 워크플로우 분할시 웹 서비스 선택의 실패기준을 나타내는 값이다. 임계값의 설정에 관한 자세한 내용은 3.3절에서 설명한다. 본 절에서는 복합 웹 서비스의 QoS 계산 모델과 요구사항의 제약정도를 계산하는 방법을 기술한다.

3.1.1 복합 웹 서비스의 QoS 계산 모델

복합 웹 서비스의 QoS는 서비스를 구성하는 단일 웹 서비스의 QoS로부터 도출된다. 일반적으로 서비스의 QoS는 상수가 아닌 범위 값으로 표현된다. 제안된 방법은 QoS의 계산을 단순화하고 서비스의 QoS가 변하더

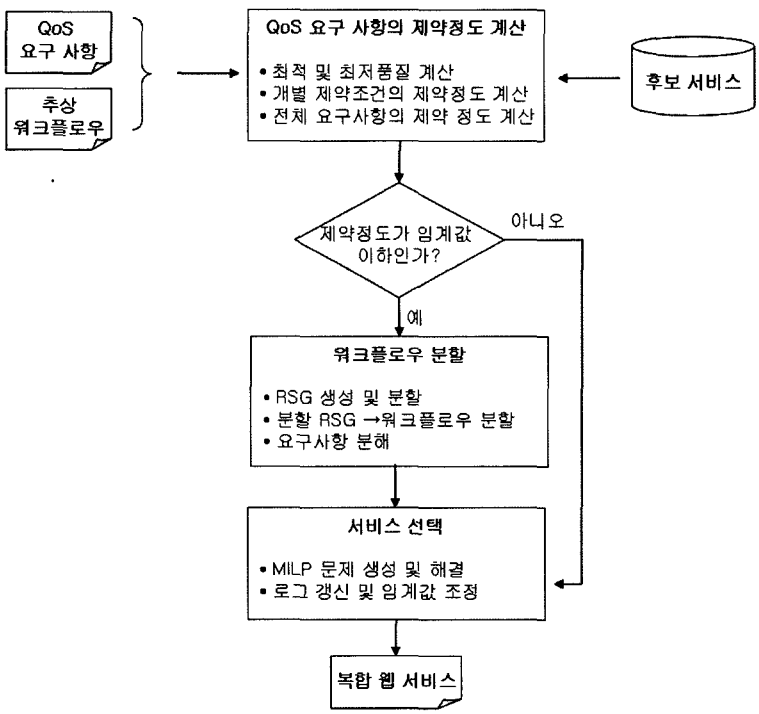


그림 2 제안된 복합 웹 서비스 선택 과정

라도 요구사항을 만족시킬 수 있도록 QoS의 최저 품질 값이 전체 범위를 대표한다고 간주한다. 예를 들어, 어떤 서비스의 지연시간이 10ms 이상 20ms 이하일 때 이 서비스의 지연시간은 최대값인 20ms로 간주한다. 본 논문은 표 2와 같이 QoS의 종류를 4개 범주로 분류하고 각 범주의 대표값을 정의한다. 제안된 방법은 표 2에 명시된 품질을 대상으로 한다. 하지만 그 외의 품질도 제시된 품질 범주에 속할 경우 동일한 방법으로 해결할 수 있다.

제안된 방법은 순차, 병렬, 조건, 루프구조를 가지는 워크플로우에 적용 가능하다. 단, 루프구조는 해당 루프의 최대 반복 횟수만큼 루프 구간을 복제하여 미리 제거한다. 예를 들어, 그림 3과 같이 최대 3회 수행된 루프의 경우 해당 루프 내부의 구조가 3회 반복되는 구조로 변환한다. 이때 최대 수행 횟수는 과거 로그를 통하여 알 수 있다고 가정한다.

순차, 병렬, 그리고 조건구조에 대하여 제안된 품질 계산 방법은 각각 그림 4, 그림 5, 그리고 그림 6과 같다. 여기서 $time(t_i)$, $cost(t_i)$, $prob(t_i)$, 그리고 $cap(t_i)$

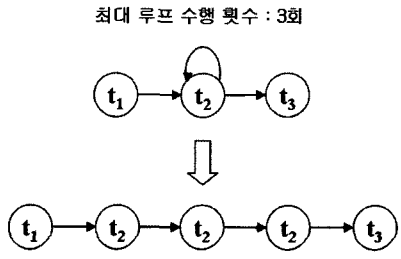


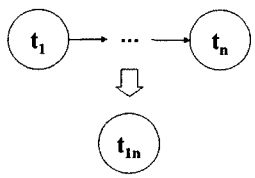
그림 3 루프구조 제거의 예

는 각각 태스크 t_i에 대한 시간, 비용, 확률, 그리고 용량 범주의 품질에 해당한다.

그림 4의 순차구조는 태스크 t_1부터 t_n까지 순차적으로 실행되는 구조이다. 따라서 순차 구조의 시간과 비용 범주는 수행되어야 할 모든 태스크에 대한 QoS의 합으로 확률범주의 경우 QoS의 곱으로 계산될 수 있다. 용량 범주의 QoS는 태스크 중 가장 작은 용량을 가지는 태스크의 QoS에 대해 제한된다. 그림 5의 병렬구조는 태스크 t_1부터 t_n이 동시에 실행된 후 그 결과가 동기화

표 2 품질 범주와 대표값

품질 범주	대표값	품질의 예	정의
시간	최대값	지연시간(latency)	서비스 요청 후 서비스가 제공되기까지 걸린 시간
비용		비용(cost)	서비스 요청에 소요되는 비용
확률	최소값	신뢰성(reliability)	서비스 요청을 성공적으로 수행할 확률
		가용성(availability)	서비스가 동작하고 있을 확률
용량		처리량(throughput)	단위 시간 동안 성공적으로 수행된 서비스의 수
		커패서티(capacity)	요구된 성능 하에 동시에 처리 가능한 서비스의 수



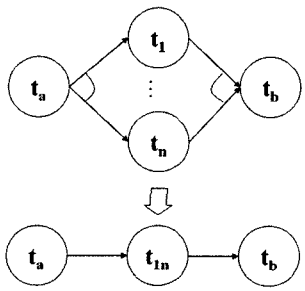
$$time(t_{1n}) = \sum_{i=1}^{n} time(t_i)$$

$$cost(t_{1n}) = \sum_{i=1}^{n} cost(t_i)$$

$$prob(t_{1n}) = \prod_{i=1}^{n} prob(t_i)$$

$$cap(t_{1n}) = \min\{cap(t_1), ..., cap(t_n)\}$$

그림 4 순차구조의 품질 계산



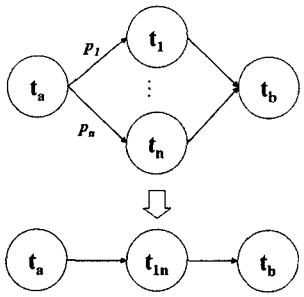
$$time(t_{1n}) = \max\{time(t_1), ..., time(t_n)\}$$

$$cost(t_{1n}) = \sum_{i=1}^{n} cost(t_i)$$

$$prob(t_{1n}) = \prod_{i=1}^{n} prob(t_i)$$

$$cap(t_{1n}) = \min\{cap(t_1), ..., cap(t_n)\}$$

그림 5 병렬구조의 품질 계산



$$time(t_{1n}) = \sum_{i=1}^{n} p_i \times time(t_i)$$

$$cost(t_{1n}) = \sum_{i=1}^{n} p_i \times cost(t_i)$$

$$prob(t_{1n}) = \sum_{i=1}^{n} p_i \times prob(t_i)$$

$$cap(t_{1n}) = \sum_{i=1}^{n} p_i \times cap(t_i)$$

그림 6 조건구조의 품질 계산

되는 구조이다. 간선에 걸친 호(arc)는 해당 간선이 모두 활성화되어야 함을 의미한다. 병렬 구조의 시간 범주 QoS의 경우 태스크들이 동시에 실행되므로 그 중 가장 QoS를 가지는, 즉, 가장 오랜 시간을 가지는 태스크의 값에 의해 결정되게 된다. 그림 6의 조건 구조는 t_a의 결과에 따라 t_1부터 t_n까지의 태스크 중 하나만 실행되는 구조이다. 각 간선에 부여된 p_i, $p_1+\ldots+p_n=1$,는 t_i가 실행될 확률이다. 조건 구조는 순차 구조가 확률에 따라 선택적으로 실행되는 구조로 볼 수 있으며 따라서 확률적 기대치를 통해 QoS를 계산한다. 이와 같이 제안된 방법은 제어구조를 단일 태스크로 환원한다. 환원된 태스크의 QoS는 환원 전의 서비스 품질과 같다. 중첩된 제어구조의 경우 가장 안쪽부터 재귀적으로 환원한다. 결과적으로 전체 워크플로우는 하나의 태스크로 환원되며 해당 QoS는 전체 워크플로우의 QoS와 같다.

3.1.2 제약정도 계산

전체 QoS 요구사항은 각 개별 품질에 대한 제약조건들로 구성된다. 따라서 전체 QoS 요구사항의 제약정도는 개별 제약조건에 대한 제약정도를 이용하여 계산한다. 개별 품질 요소 q에 대한 제약정도 $Tightness\,Of\,Constraint(q)$의 계산은 식 (1)을 이용한다. 식 (1)은 추상 워크플로우와 후보 서비스로 구성 가능한 품질 요소 q의 정규화 된 값을 의미하며 $Tightness\,Of\,Constraint(q)$의 값이 클수록 더욱 제약정도가 크다고 볼 수 있다. 용량 범주 품질의 경우, 단일 태스크 수준에서 QoS 요구사항을 만족하는 서비스를 선택할 수 있기 때문에 제약정도를 계산하지 않는다.

$$Tightness\,Of\,Constraint(q) = \frac{Constraint(q) - Worst\,Quality(q)}{Best\,Quality(q) - Worst\,Quality(q)}. \quad (1)$$

$$where\ Worst\ Quality(q) \le Constraint(q) \le Best\,Quality(q)$$
$$\text{if } q \in prob \text{ or } \cap$$

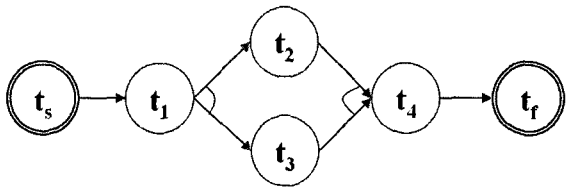
$$Best\,Quality(q) \le Constraint(q) \le Worst\,Quality(q)$$
$$\text{if } q \in time \text{ or } cost$$

이때 $Constraint(q)$와 $Best\,Quality(q)$는 각각 품질 요소 q에 대한 제약조건과 최고의 품질에 해당한다. $Best\,Quality(q)$는 각 태스크에 대한 후보 서비스 집합 중 품질 요소 q의 값이 가장 좋은 서비스를 선택한 복합 웹 서비스의 QoS이다. 예를 들어, 그림 7은 태스크당 8개의 후보 서비스가 존재하며 지연시간과 신뢰성에 대한 제약조건이 주어졌다고 가정한다. 음영 처리된 값은 후보 서비스 집합 내에서의 최고 품질이다. $Worst\,Quality(q)$는 품질요소 q에 대한 최저 품질로 $Best\,Quality(q)$와 유사한 방식으로 계산된다. 식 (1)에서 $Best\,Quality(q)$와 $Worst\,Quality(q)$는 각각 만족시키기 가장 어려운 QoS와 항상 만족시킬 수 있는 QoS에 해당한다. 즉, 주어진 제약조건이 $Worst\,Quality(q)$보다 낮은 품질을 요구한다면 그 요구는 서비스 선택에 관계없이 항상 만족되며 반대로 $Best\,Quality(q)$ 보다 높은 품질을 요구한다면 그 요구는 항상 만족될 수 없다. 따라서 식 (1)은 $Constraint(q)$가 $Worst\,Quality(q)$와 $Best\,Quality(q)$사이에 존재하는 경우만을 가정한다.

일반적으로 전체 요구사항을 구성하는 제약조건 중 높은 제약정도를 가지는 제약조건이 상대적으로 낮은 제약정도를 가지는 제약조건에 비해 전체 요구사항의 제약정도에 미치는 영향이 더 크다고 볼 수 있다. 전체 요구사항에 n개의 제약조건이 존재하고, 개별 제약조건의 제약정도 $Tightness\,Of\,Constraint(q)$ 중 i번째로 큰 값을 $Tightness_i$라고 할 때, 전체 요구사항의 제약 정도 $Tightness\,of\,Requirement$는 식 (2)를 이용해 계산한다. 식 (2)는 각 개별 제약조건의 제약정도를 제약정도의 순위를 나타내는 i의 제곱으로 나눔으로서 비중을 달리하여 합한다.

$$Tightness\,of\,Requirement = \frac{\sum_{i=1}^{n} \frac{Tightness_i}{i^2}}{\sum_{i=1}^{n} \frac{1}{i^2}} \quad (2)$$

개별 제약조건의 제약정도와 마찬가지로 $Tightness\,Of\,Requirement$가 크면 클수록 만족시키기 어려운 요구사항을 의미한다. $Tightness\,Of\,Requirement$는 주어진 요



t_i / s_{ij}	t_1		t_2		t_3		t_4	
	지연시간	신뢰성	지연시간	신뢰성	지연시간	신뢰성	지연시간	신뢰성
s_{i1}	20	0.995	113	0.992	10	0.992	25	0.992
s_{i2}	15	0.997	150	0.994	88	0.996	33	0.995
s_{i3}	30	0.998	200	0.996	75	0.998	50	0.991
s_{i4}	10	0.991	170	0.996	66	0.991	45	0.994
s_{i5}	11	0.995	130	0.991	81	0.992	83	0.994
s_{i6}	25	0.997	120	0.995	35	0.998	21	0.993
s_{i7}	18	0.992	119	0.992	90	0.999	50	0.992
s_{i8}	33	0.993	155	0.993	24	0.993	48	0.992

$$BestQuality(time) = time(s_{14}) + Max\{time(s_{21}), time(s_{31})\} + time(s_{46})$$
$$= 10 + 113 + 21 = 143$$

$$BestQuality(rel) = rel(s_{13}) \times rel(s_{23}) \times rel(s_{37}) \times rel(s_{42})$$
$$= 0.998 \times 0.996 \times 0.999 \times 0.995 = 0.988$$

그림 7 $BestQuality(q)$의 계산 예

구사항의 제약정도를 가늠하는 기준이 되며 임계값 $TH_{tightness}$와 비교하여 작은 경우에 대해 워크플로우 분할을 수행한다. $TH_{tightness}$는 과거 실패하였던 요구사항들의 제약사항을 이용하여 설정한다. 이에 관한 자세한 설명은 3.3절에서 기술한다.

3.2 워크플로우 분할

복합 웹 서비스 선택 문제는 모든 후보 서비스 집합의 조합을 고려해야 하기 때문에 선택에 걸리는 시간은 전체 후보 서비스 개수에 가장 크게 영향을 받는다. 워크플로우의 전체 후보 서비스 집합은 개별 태스크의 후보 서비스 집합으로 구성된다. 따라서 워크플로우의 분할은 동시에 후보 서비스 집합의 분할을 의미하며 분할된 문제를 각각 해결하는 것이 전체 문제를 해결하는 것보다 빠르다고 기대할 수 있다. 한편 워크플로우 분할이 이루어지면 글로벌 제약조건을 만족하는 서비스 선택을 위해 QoS 요구사항 또한 분해되어야 한다. 본 절은 워크플로우를 분할한 후, 분할된 워크플로우의 QoS 요구사항을 재설정 하는 방법을 기술한다.

3.2.1 RSG 생성 및 분할

분할의 첫 번째 과정으로 RSG(Reduced Sequential Graph)를 생성한다. RSG는 추상 워크플로우에 속한 병렬 및 조건 구조를 순차구조로 환원한 그래프이다. 초기 RSG는 추상 워크플로우의 DAG(directed acyclic graph) 표현과 동일하며 각 정점(vertex)은 대응되는 태스크의 후보 서비스 개수를 저장한다. 초기 RSG는 그림 8의 규칙을 따르는 환원과정을 거쳐 최종 RSG로 변환된다.

규칙 1. 추상 워크플로우의 각 조건 및 병렬구조의 태스크에 대응되는 정점들은 단일 정점으로 환원된다.

규칙 2. 추상 워크플로우의 순차구조가 조건 및 병렬구조 내에 중첩되어 있는 경우 해당 순차구조의 태스크에 대응되는 정점들은 단일 정점으로 환원된다.

규칙 3. 환원된 정점은 환원 전 제어구조의 앞에 위치한 정점으로부터 들어오는 간선과 뒤에 위치한 정점으로 향하는 간선을 갖는다.

규칙 4. 추상 워크플로우의 제어구조가 중첩된 경우 가장 안쪽부터 재귀적으로 환원한다.

규칙 5. 환원과정을 통해 생성된 정점은 환원된 정점들의 후보서비스 개수의 합과 동일한 후보서비스 개수를 갖는다.

그림 8 RSG 환원 규칙

환원과정은 3.1.1절의 내용과 유사하다. 최종 RSG는 단일 경로만을 가지며 각 정점은 추상 워크플로우의 한 개 이상의 태스크에 대응된다.

그림 9는 초기 RSG와 환원과정을 거친 최종 RSG의 예를 보여준다. 각 정점 아래에 표시된 수는 대응되는 태스크의 후보 서비스 개수이다. 초기 RSG에서 t_3, t_4, 그리고 t_5로 구성된 조건 구조에 대응하는 정점들을 환원하여 정점 t_{345}를 생성하면 최종 RSG가 된다. RSG는 워크플로우가 분할되더라도 조건 및 병렬구조가 그대로 유지되도록 하며, QoS 계산 및 분해를 단순화시킨다. 또한 병렬구조의 시간 범주의 품질은 구조내의 모든 서비스에 의해 결정되기 때문에 분할될 수 없다. 최종 RSG를 생성하는 과정은 추상 워크플로우가 n개의 태스크를 가질 때 O(n)의 시간복잡도를 갖는다.

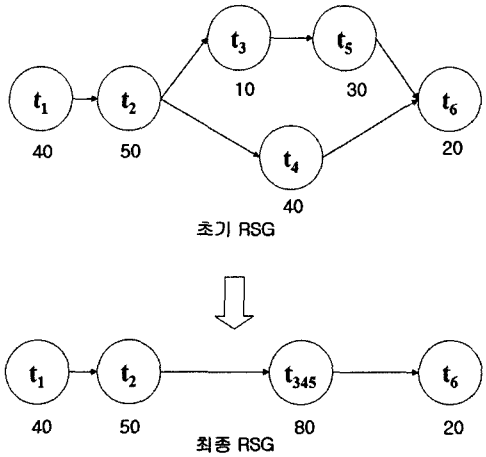


그림 9 초기 RSG와 최종 RSG

최종 RSG가 완성되면 각 정점에 저장된 후보 서비스 개수를 원소로 갖는 집합을 생성한다. 생성된 집합은 근사 부분 합 (approximate subset sum) 알고리즘 [18]을 이용하여 전체 후보 서비스 집합의 절반에 근사한 크기를 가지는 두 문제로 분할된다. 이러한 분할 기준은 제약조건 재설정에 따른 선택 실패 가능성을 최소화하면서 분할된 문제의 크기가 가능한 한 비슷하도록 한다.

분할 과정은 최종 RSG의 정점 수에 대하여 다항시간(polynomial-time)내에 해결된다. 그림 10은 그림 9의 최종 RSG에 대한 분할결과이다. 특히 분할된 두 집합, p'과 p'',에 포함된 후보 서비스 개수의 비가 일정 임계값 $TH_{partition}$ 이상일 경우 $(\frac{p'}{p''} \geq TH_{partition},\ p' \leq p'')$ 워크플로우를 분할한다. 마지막으로 분할된 집합으로부터 분할된 워크플로우를 생성한다. 분할된 워크플로우의 생성과정은 그림 10과 같이 각 원소에 대응하는 RSG의 정점을 함께 저장하여 쉽게 이루어질 수 있다. RSG는 순차구조로만 구성되어 있으며 순차구조의 QoS 계산은 교환법칙이 성립하는 연산을 통해 이루어진다. 따라서 각 집합의 원소에 대응하는 정점들을 순서에 상관없이

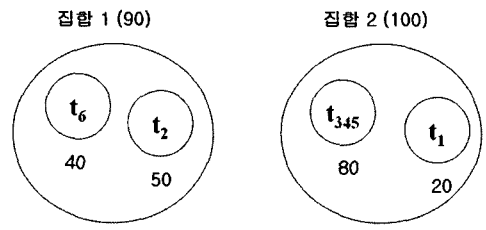


그림 10 근사 부분 합의 결과

순차적으로 연결한 뒤, 정점에 대응하는 추상 워크플로우의 구조로 치환하여 분할된 워크플로우를 생성한다. 이렇게 분할된 추상 워크플로우는 복합 웹 서비스의 QoS 계산을 위한 워크플로우 구조만을 표현한다.

그림 11은 그림 10의 두 집합에 대한 RSG 및 분할된 최종 워크플로우를 보여준다. 태스크 t_s와 t_f는 각각 전체 프로세스의 시작과 끝을 표현하기 위해 삽입되는 더미(dummy) 태스크이다. 그림의 예제에서 왼쪽(또는 오른쪽) 워크플로우의 t_f에서 오른쪽(또는 왼쪽) 워크플

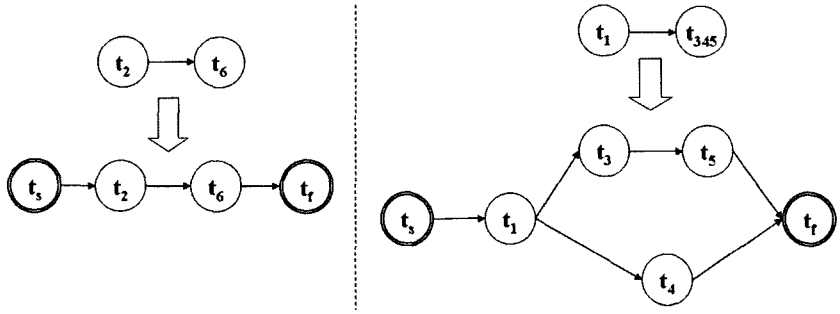


그림 11 분할된 워크플로우의 생성 예

로우의 t_s로 향하는 간선을 삽입하여 두 워크플로우를 연결하면 분할 전과 동일한 서비스 품질을 가지는 워크플로우가 생성됨을 알 수 있다. 이 사실은 분할된 두 RSG를 순차형태로 연결하면 좀 더 간단히 확인할 수 있다. 연결된 RSG는 분할 전의 RSG와 태스크의 순서만이 다르므로 서로 같은 품질을 갖는다.

현재 제안된 RSG 환원규칙은 완전히 중첩된 구조(fully-nested)만으로 구성된 워크플로우에 한해 적용할 수 있다는 단점이 있다. 부분적으로 중첩된(partially-nested) 구조의 경우 일반화 된 QoS 계산 방법을 적용할 수 없으며 개별 워크플로우에 대하여 각 제어 구조를 분석하여 QoS를 계산하여야 한다. 본 논문에서는 자동화된 서비스 선택이 가능한 완전히 중첩된 구조를 가지는 워크플로우를 대상으로 한다.

3.2.2 QoS 요구사항의 재설정

분할된 워크플로우를 통해 선택된 복합 웹 서비스는 주어진 QoS 요구사항을 만족하여야 한다. 따라서 원래의 QoS 요구사항을 적절히 분해하여 분할된 각 워크플로우에 대한 QoS 요구사항으로 재설정한다. 전술한 바와 같이 전체 워크플로우의 QoS는 분할된 두 워크플로우를 순차구조로 연결하였을 때의 QoS로 계산되어지므로 분할된 두 워크플로우의 품질요소 q에 대한 제약조건 $Constraint'(q)$과 $Constraint''(q)$에 대해 식 (3)이 성립한다.

$$Constraint(q)=\begin{cases} Constraint'(q)+Constraint''(q) & \text{if } q\in time \text{ or } cost \\ Constraint'(q)\times Constraint''(q) & \text{if } q\in prob \\ Constraint'(q)=Constraint''(q) & \text{if } q\in \cap \end{cases} \quad (3)$$

식 (3)에서 알 수 있듯이 분할된 워크플로우의 용량 범주 품질의 경우 분할 전의 제약사항과 동일하다.

$Constraint'(q)$ (또는 $Constraint''(q)$)에 대해 선택된 복합 웹 서비스의 품질은 $Constraint'(q)$ (또는 $Constraint''(q)$)과 동일하거나 더 좋으며 따라서 분할된 두 문제를 결합한 전체 품질은 $Constraint(q)$와 동일하거나 더 높은 품질을 갖는다. 즉, 분해된 제약조건을 통해 항상 분할 전의 글로벌 제약조건을 만족함을 보장할 수 있다. 그러나 실제 QoS는 선택된 후보 웹 서비스의 모든 품질 요소에 의해 결정되는데 반해 식 (3)은 전체 요구사항을 구성하는 각각의 제약조건을 분리하여 계산하였기 때문에 다수의 제약조건을 가지는 요구사항에 대해 분할 전 요구사항을 만족하는 서비스가 존재할지라도 재설정된 요구사항을 만족하는 서비스가 존재함을 보장하지는 못한다.

제안된 방법은 재설정된 요구사항에 의해 복합 웹 서비스를 선택하지 못하는 경우를 줄이기 위해 분할된 각 워크플로우에 대해 가장 제약정도가 높은 제약조건인 $BestQuality'(q)$과 $BestQuality''(q)$을 이용한다. $BestQuality'(q)$과 $BestQuality''(q)$은 분할된 워크플로우의 품질 요소 q에 대한 최적의 품질이다. 전술한 식 (3)의 $Constraint'(q)$ (또는 $Constraint''(q)$)가 가지는 값은 품질 요소 q의 도메인에 속하며 이와 같은 도메인에 속하는 $BestQuality'(q)$와 $BestQuality''(q)$로 치환하게 되면 식 (4)가 성립한다.

$$BestQuality(q)=\begin{cases} BestQuality'(q)+BestQuality''(q) \\ \quad \text{if } q\in time \text{ or } cost \\ BestQuality'(q)\times BestQuality''(q) \\ \quad \text{if } q\in prob \end{cases} \quad (4)$$

한편 식 (1)에 의해 $BestQuality(q)$는 정규화 범위내의 품질 값 중 가장 큰 값이 되며 $BestQuality(q)$ = $Constraint(q)$일 경우 $TightnessOfConstraint(q)$의 값은 1이 된다. 따라서 품질 요소 q에 대한 가장 큰 $TightnessOfConstraint(q)$의 값을 갖는 제약조건, 즉, 가장 높은 제약정도의 $Constraint(q)$와 동일하다. 마찬가지로 $Constraint'(q)$ (또는 $Constraint''(q)$)이 $BestQuality'(q)$ (또는 $BestQuality''(q)$)보다 더욱 높은 품질을 요구한다면 해당 요구사항을 만족할 수 있는 복합 웹 서비스는 존재하지 않으며 서비스 선택은 실패하게 된다.

제안된 방법은 분할된 워크플로우의 최적의 품질인 $BestQuality'(q)$과 $BestQuality''(q)$의 비율에 따라 $Constraint'(q)$과 $Constraint''(q)$ 값을 설정한다. 즉, 식 (5)를 만족하는 값 중 하나로 설정된다. 이렇게 분할된 $Constraint'(q)$ 또는 $Constraint''(q)$는 식 (1)의 전제에 따라 각 워크플로우의 최적 및 최저 품질 사이의 값을 갖는다. 또한, 식 (3), (4), 그리고 (5)에 의해 식 (6)이 성립한다.

$$\begin{aligned} &BestQuality'(q):BestQuality''(q) \\ &=Constraint'(q):Constraint''(q) \end{aligned} \quad (5)$$

$$\begin{cases} BestQuality(q):(BestQuality'(q)+BestQuality''(q)) \\ =Constraint(q):(Constraint'(q)+Constraint''(q)) \\ \quad \text{if } q\in time \text{ or } cost \\ BestQuality(q):(BestQuality'(q)\times BestQuality''(q)) \\ =Constraint(q):(Constraint'(q)\times Constraint''(q)) \\ \quad \text{if } q\in prob \end{cases} \quad (6)$$

이원 일차 방정식을 이용해 식 (5)를 이용하여 식 (6)에서 $Constraint''(q)$을 제거하게 되면 식 (7)을 얻을 수 있으며 식 (7)을 통해 $Constraint'(q)$을 계산할 수 있다. $Constarint'(q)$을 계산하면 $Constraint''(q)$은 식

(3)을 이용하여 쉽게 계산할 수 있다. 이때 분할된 두 워크플로우 중 어느 쪽이 $Constraint'(q)$이 되던지 결과는 동일하다.

$$Constraint'(q)=\begin{cases} Constraint(q)\times\dfrac{BestQuality'(q)}{BestQuality(q)} & \text{if } q\in time \text{ or } cost \\ \sqrt{Constraint(q)\times\dfrac{BestQuality'(q)}{BestQuality''(q)}} & \text{if } q\in prob \\ Constraint(q) & \text{if } q\in \cap \end{cases} \tag{7}$$

분할된 워크플로우에 대해 서비스 선택이 실패하였을 경우, 원래 요구사항을 만족하는 서비스가 없기 때문인지와 분할 후 재설정된 요구사항을 만족하는 서비스가 없기 때문에 실패한 경우를 구분할 수 없다. 따라서 분할된 문제에 대한 선택이 실패하였을 경우, 원래의 요구사항을 만족하는 복합 웹 서비스의 존재 여부를 확인하기 위해 분할 전 워크플로우에 대한 선택과정을 적용해야 한다.

3.3 서비스 선택

제안된 방법은 효율적인 서비스 선택을 위해 MILP 기법을 사용한다. 앞서 언급한 바와 같이 다수의 품질 요소를 고려하는 서비스 선택 문제는 선형시간에 해결이 불가능한 조합 최적화 문제이다. MILP는 변수, 목적함수, 그리고 제약사항으로 구성되며, 주어진 제약사항을 만족하면서 목적함수를 최대화 또는 최소화하는 변수의 값을 찾는다. 이때 목적함수와 제약사항은 선형, 즉, 덧셈으로 구성된 식이어야 한다는 제약을 갖는다. MILP는 ILP와 달리 실수 값을 취하는 일반 변수와 정수 값을 취하는 정수 변수를 모두 사용한다. 제안된 방법은 ILP에 기반한 기존의 방법과 달리 병렬 구조의 지연시간 계산을 위해 CPA(critical path algorithm)를 사용하지 않는다. 또한 모든 조건 구조를 동시에 고려하기 때문에 글로벌 제약조건이 만족함을 보장한다.

서비스 선택 문제는 다음과 같은 MILP 문제로 표현될 수 있다. 목적함수는 복합 웹 서비스의 QoS를 최대화(최적화)하며 다음과 같이 정의된다.

$$Maximize: \sum_{q\in\forall}\frac{Quality(q)-WorstQuality(q)}{BestQuality(q)-WorstQuality(q)}\times W_q. \tag{8}$$

이때 Quality(q)는 복합 웹 서비스의 품질 요소 q에 대한 품질이며, BestQuality(q)와 WorstQuality(q)는 각각 품질 요소 q에 대한 최고 및 최저의 품질 값이다. W_q, $\sum W_q=1$,는 품질 요소 q에 대한 가중치이다. 위의 목적함수는 다수의 결정 요소에 대한 스코어(score)를 정규화된 가중합을 통해 계산하는 SAW(simple additive weighting)[20]에 기반한다.

MILP 제약사항은 복합 웹 서비스의 QoS와 글로벌 QoS 제약사항을 표현한다. CS_{ij}를 태스크 t_i의 j번째 후보 서비스라고 할 때, 모든 CS_{ij}는 이진 변수 s_{ij}, $s_{ij}\in\{0,1\}$,에 매핑된다. 따라서 다음의 제약사항은 태스크 t_i의 후보 서비스 중 오직 하나만이 선택될 수 있음을 표현한다.

$$\sum_{j=1}^{CS(i)} s_{ij}=1. \tag{9}$$

이때 CS(j)는 태스크 t_i의 후보 서비스 수이다. s_{ij}의 값은 CS_{ij}가 선택 되었을 때 1이 되며, 선택되지 않았을 때 0이 된다. (9)를 이용하여 각 태스크 당 하나의 서비스만이 선택될 수 있음을 보장할 수 있다. 따라서 태스크 t_i의 QoS에 대한 제약사항은 다음과 같다.

$$Quality(q,t_i)=\sum_{j=1}^{CS(i)} Quality(q,CS_{ij})\times s_{ij}. \tag{10}$$

이때, Quality(q, CS_{ij})는 CS_{ij}의 품질 요소 q에 대한 실제 QoS이다.

복합 웹 서비스의 QoS는 MILP 제약사항을 통해 표현된다. 표 3은 각 워크플로우의 구조에 대해 QoS를 표현하는 MILP 제약사항을 기술한다.

확률 범주의 경우 품질들의 곱으로 계산되기 때문에 곧바로 MILP 제약사항으로 표현할 수 없다. 따라서 Zeng 등이 제안한 방법과 같이 로그 함수를 사용하여 덧셈형태로 바꾸어 표현한다. 확률 범주의 경우 식 (10) 대신 다음의 MILP 제약사항을 사용하여 태스크 ti의 품질을 표현한다.

$$Quality'(probability,t_i)=\sum_{i=1}^{n}\ln(Quality(probablity,CS_{ij}))\times s_{ij}. \tag{23}$$

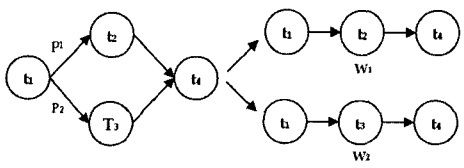
순차 및 병렬 구조의 용량 범주 품질 그리고 병렬 구조의 시간 범주 품질의 경우 각각 최소 그리고 최대 값으로 계산되며 직접적으로 MILP 제약사항으로 표현될 수 없다. 따라서 표 3의 식 (14)와 같이 각 태스크마다 개별적으로 MILP 제약사항을 정의한다. 용량 범주의 경우, Qualitiy(capacity)가 증가할수록 목적 함수 값도 커지므로, Qualitiy(capacity)는 MILP 해결과정을 통해 Quality(capacity, t_i) 중 가장 작은 값에 바운드(bound) 될 때까지 최대화 된다. 반대로 시간 범주의 경우 Quality(time)은 Quality(time, t_i)중 가장 큰 값과 같아지게 된다.

조건 구조의 경우, 태스크 t_i로 분기할 확률 p_i에 대한 기댓값 형태로 표현된다. 확률 범주 품질의 경우, 로그 함수를 변수들의 합에 적용해야 하므로 곧바로 MILP 제약사항으로 표현될 수 없다. 따라서 모든 분기에 대해

표 3 QoS에 대한 MILP 제약사항

워크플로우 구조	품질 범주	MILP 제약사항	식 번호
순차	시간	$Quality(time) = \sum_{i=1}^{n} Quality(time, t_i)$	(11)
	비용	$Quality(\cos t) = \sum_{i=1}^{n} Quality(\cos t, t_i)$	(12)
	확률	$Quality'(probability) = \sum_{i=1}^{n} Quality(probability, t_i)$	(13)
	용량	$Quality(\cap acity) \le Quality(\cap acity, t_i), 1 \le i \le n$	(14)
병렬	시간	$Quality(time) \ge Quality(time, t_i), 1 \le i \le n.$	(15)
	비용	$Quality(\cos t) = \sum_{i=1}^{n} Quality(\cos t, t_i)$	(16)
	확률	$Quality'(probability) = \sum_{i=1}^{n} Quality(probability, t_i)$	(17)
	용량	$Quality(\cap acity) \le Quality(\cap acity, t_i), 1 \le i \le n$	(18)
조건	시간	$Quality(time) = \sum_{i=1}^{n} Quality(time, t_i) \times p_i,$	(19)
	비용	$Quality(\cos t) = \sum_{i=1}^{n} Quality(\cos t, t_i) \times p_i.$	(20)
	확률	$Quality(probability) = \sum_{k=1}^{b} Quality(probability, W_k).$	(21)
	용량	$Quality(\cap acity) \le \sum_{i=1}^{n} Quality(\cap acity, t_i) \times p_i$	(22)

각각 MILP 제약사항을 정의한다. 예를 들어, 그림 12는 두 개의 분기를 가진 워크플로우에 대한 확률 범주의 품질을 표현하는 MILP 제약사항을 보여준다. 그림의 QoS(probability, W_k)는 분기 k에 대한 확률 범주의 품질이다.



$$Quality(probability, W_1) = \sum_{i\in\{1,2,4\}} \sum_{j=1}^{CS(i)} \ln(p_{ij} \cdot Quality(probabiliity, CS_{ij})) \cdot s_{ij},$$

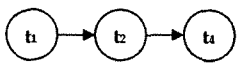
$$Quality(probability, W_2) = \sum_{i\in\{1,3,4\}} \sum_{j=1}^{CS(i)} \ln(p_{ij} \cdot Quality(probabiliity, CS_{ij})) \cdot s_{ij}.$$

그림 12 확률 범주 품질에 대한 MILP 제약사항의 예

그림 12의 예와 같이 각 분기에 대한 확률 범주의 품질은 식 (13)를 기반으로 하되 분기 확률이 곱해진 형태로 표현된다. 전체 워크플로우의 확률 범주 품질에 대한 MILP 제약사항은 식 (21)과 같이 모든 분기에 대한 워크플로우의 확률 범주 품질을 더한 값이 된다. 이때 b는 모든 분기에 대한 워크플로우의 수이다.

표 3의 MILP 제약사항의 좌변은 해당 구조의 품질을 나타내는 변수이다. 따라서 중첩된 구조의 경우 해당하는 변수를 사용하여 품질을 표현한다. 결과적으로 복합 웹 서비스의 품질은 전체 워크플로우에 대한 MILP 제약사항으로 표현된다. 복합 웹 서비스의 품질을 나타내는 변수들은 (8)의 목적 함수를 표현하는데 사용된다.

전술한 바와 같이, QoS 제약사항 또한 MILP 제약사항으로 표현된다. 만약 워크플로우에 어떠한 분기도 없다면 복합 웹 서비스의 품질을 표현하는 각 변수 값의 범위를 제한함으로써 QoS 제약사항을 표현할 수 있다. 반면 분기가 존재하는 경우 모든 분기 가능한 경로의 품질을 표현하는 MILP 제약사항을 통해 QoS 제약사항을 표현한다. 그림 13은 그림 12의 분기 W1에 대한 QoS 제약사항에 대한 예를 보여준다. QoS 제약사항에 있어서는 각 분기가 실제 실행될 상황을 고려하기 때문에 분기 확률은 곱하지 않는다. 용량 범주의 경우 각 태스크에 대해 제약사항을 만족하지 못하는 후보 서비스를 제외하여 제약사항이 만족됨을 보장할 수 있기 때문



$$\sum_{i\in\{1,2,4\}} Quality(time, t_i) \le Constraint(time),$$

$$\sum_{i\in\{1,2,4\}} Quality(cost, t_i) \le Constraint(cost),$$

$$\sum_{i\in\{1,2,4\}} Quality'(probability, t_i) \ge \ln(Constraint(probability)).$$

그림 13 QoS 제약사항에 대한 MILP 제약사항의 예

에 MILP를 이용하지 않는다.

분할된 워크플로우의 경우 두 MILP가 모두 성공하지 않으면 전체 서비스 선택은 실패하게 된다. 이때 두 문제의 해결 순서는 임의로 정하며 먼저 시도된 문제가 해결될 경우 남은 문제의 QoS 요구사항은 식 (3)을 이용하여 다시 계산한다. 이때 Constraint'(q)은 앞서 해결된 문제의 실제 QoS를 사용한다. 이 방법은 나중에 해결되는 문제의 제약사항을 완화시켜 서비스 선택의 성공률을 높이게 된다.

서비스 선택에 실패한 경우, 해당 QoS 요구사항의 제약정도를 로그에 기록한 뒤 임계값 $TH_{tightness}$를 갱신한다. $TH_{tightness}$는 로그를 기반으로 ε $(0 \le \varepsilon \le 1)$만큼의 실패를 허용하도록 설정된다. 예를 들어, 100개의 로그 데이타가 존재하며 ε = 0.05일 때, 임계값은 로그 데이타 중 5번째로 작은 제약정도가 된다. ε값이 증가하면 성공률은 증가하는 반면 재현율은 줄어든다. 반대로 ε값이 감소하면 성공률은 감소하는 반면 재현율이 증가한다.

4. 실험 결과

성능 평가를 위해 제안된 서비스 선택 방법의 성공률 및 재현율, 선택 속도, 결과 서비스의 QoS의 측면에서 실험하였다. 실험 결과 제안된 워크플로우 분할 방법은 99% 이상의 성공률을 보였다. 특히 워크플로우 분할 시 모든 경우에 대해서 복합 웹 서비스를 보다 빠르게 선택하였으며 이때 선택된 복합 웹 서비스의 QoS는 최적 서비스와 5%미만의 차이를 보였다.

4.1 실험 환경

실험은 표 4와 같이 고정된 크기의 후보 서비스 집합에 대하여 워크플로우의 태스크 수를 증가시키는 경우와 고정된 태스크 개수를 가지는 워크플로우에 대하여 후보 서비스 집합의 크기를 증가시키는 두 가지 경우에 대하여 수행되었다. 실험 1의 경우 태스크 수의 증가에 따른 성능의 변화를 파악하기 위해 순차구조만을 포함한 워크플로우를 사용하였으며 태스크는 4개부터 20개까지 5단계에 걸쳐 4개씩 증가시켰다. 전체 후보 서비스의 수는 1000개로 동일하다. 실험2의 경우 후보 서비스 수의 증가에 따른 성능의 변화를 파악하기 위해 후보 서비스 수를 500개부터 2500개까지 5단계에 걸쳐 500개씩 증가시켰다. 실험 2에 사용된 워크플로우의 구조는 그림 14와 같다.

표 4 실험에 사용된 워크플로우의 특징

구분	워크플로우 구조	태스크 수	후보 서비스 수
실험 1	순차	단계별로 증가	변화 없음
실험 2	순차, 병렬, 조건	변화 없음	단계별로 증가

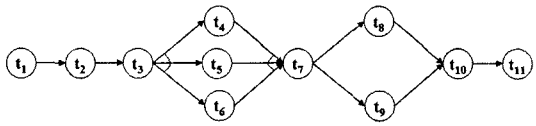


그림 14 실험 2에 사용된 워크플로우

각 태스크는 표 5의 5가지 품질 클래스 중 임의의 한 가지에 속하게 된다. 각 후보 웹 서비스의 품질은 후보 웹 서비스가 속한 태스크의 품질 클래스의 기준을 따른다. 서비스 선택에 사용된 QoS 요구사항은 2개, 3개, 그리고 4개의 제약사항을 가지는 경우에 대해 각각 5000개를 생성하였다. 각 QoS 요구사항에 포함된 품질 요소는 무작위로 선택되며 그에 대한 품질은 $BestQuality(q)$와 $WorstQuality(q)$사이에서 무작위로 설정되었다. 이때 무작위로 생성된 요구사항의 성격에 따라 실험 결과는 변할 수 있으나 두 방법에 공평한 요구사항을 생성하는 것은 매우 어려운 문제이므로 각 경우에 있어 5000개라는 많은 요구사항을 생성하여 실험의 일반성을 높이고자 하였다. 이때 실험에 사용된 $TH_{partition}$과 ε은 각각 0.3과 0.05였으며, 각 품질 요소에 대한 가중치 Wq는 동일하게 설정하였다. 실험은 Intel Pentium™4 2.4GHz, 1GB RAM에서 수행되었다.

표 5 실험에 사용된 품질 클래스와 품질 요소 설정 기준

품질 요소	품질 클래스				
	1	2	3	4	5
지연 시간	0-40	40-80	80-120	120-160	160-200
가격	80-100	60-80	40-60	20-40	0-20
가용성	0.99-0.999	좌동	좌동	좌동	좌동
신뢰성	0.99-0.999	좌동	좌동	좌동	좌동

4.2 성능 평가

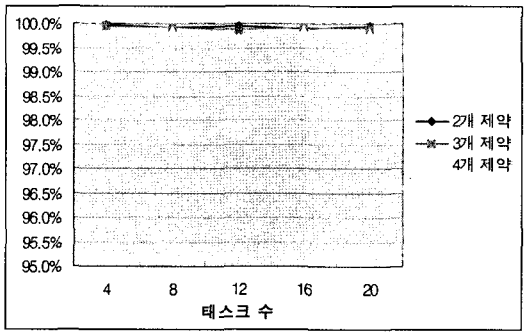
본 절에서는 제안된 분할 방법의 성공률 및 재현율, 선택 속도, 그리고 선택된 복합 서비스의 품질의 세 가지 측면에서 실험 결과를 자세히 기술한다.

4.2.1 워크플로우 분할의 성공률 및 재현율

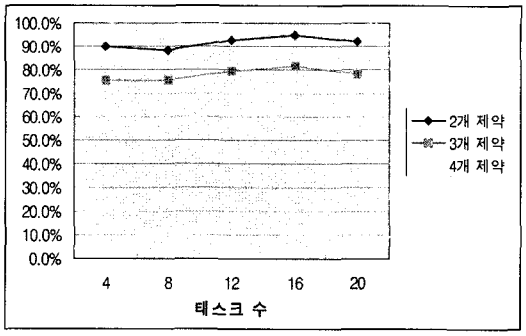
제안된 방법이 워크플로우의 분할 여부를 적절히 판단하는지를 측정하기 위해서 식 (24)와 식 (25)와 같이 성공률과 재현율을 계산하였다. 이때 분할 가능한 요구사항이란 분할된 워크플로우로부터 QoS 요구사항을 만족하는 복합 웹 서비스를 선택할 수 있는 경우이다. 그림 15와 그림 16은 각각 실험 1과 실험 2의 성공률 및 재현율을 나타낸다.

$$\text{성공률} = \frac{\text{분할하여 서비스 선택에 성공한 횟수}}{\text{총 분할한 횟수}} \quad (24)$$

$$\text{재현율} = \frac{\text{분할하여 서비스 선택에 성공한 횟수}}{\text{총 분할 가능한 요구사항의 수}} \quad (25)$$

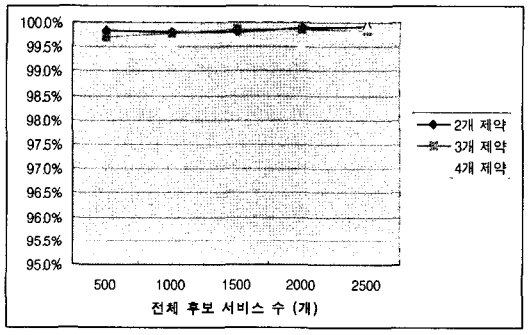


(a) 성공률

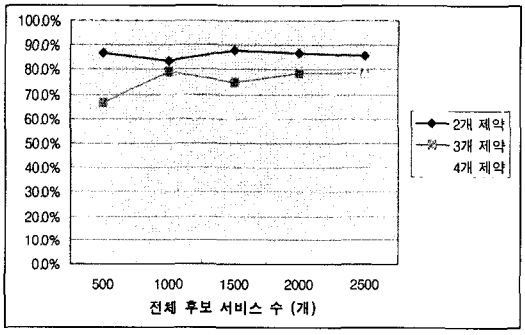


(b) 재현율

그림 15 실험 1의 성공률과 재현율



(a) 성공률



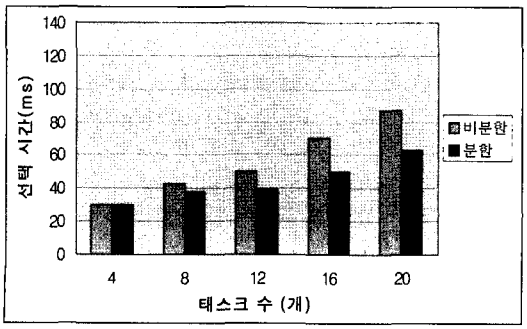
(b) 재현율

그림 16 실험 2의 성공률과 재현율

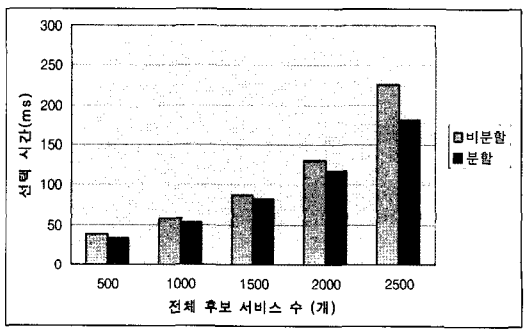
실험 결과, 성공률 및 재현율은 태스크나 후보서비스의 수 보다 제약조건의 수에 영향을 받는 것으로 나타났다. 성공률의 경우 제약조건의 수가 증가할수록 점차 낮아졌지만 전체적으로 99% 이상의 결과를 보였다. 한편 재현율의 경우 제약조건이 늘어남에 따라 비교적 큰 폭으로 낮아졌다. 이러한 현상이 발생하는 이유는 전체 요구사항에 포함된 제약조건의 수가 증가함에 따라 제약정도를 하나의 값으로 표현하기 어려워지기 때문이며 따라서 성공하는 요구사항이 실패하는 요구사항과 유사한 제약정도를 갖는 것으로 계산되었다고 볼 수 있다.

4.2.2 복합 웹 서비스의 선택 속도

선택 속도를 평가하기 위해 제안된 방법 분할 방법을 적용한 경우와 그렇지 않은 경우를 비교하였다. 그림 17, 그림 18, 그리고 그림 19는 각각 2개, 3개, 그리고 4개의 제약조건을 가지는 QoS 요구사항에 대한 선택 속도를 나타낸다. 실험 결과, 모든 경우에 있어서 제안된

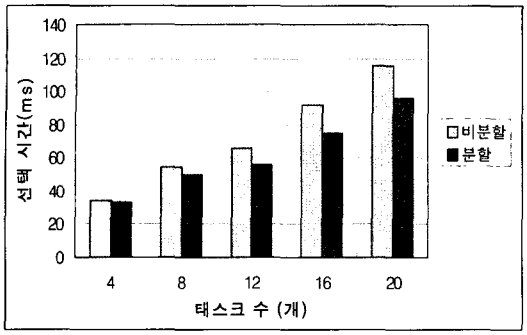


(a) 실험 1

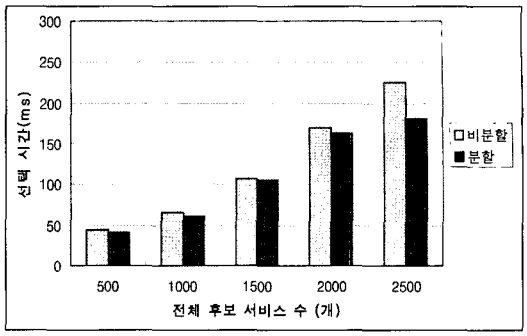


(b) 실험 2

그림 17 선택 속도 (제약조건이 2개인 경우)

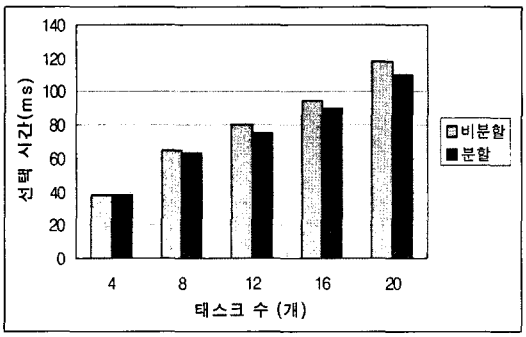


(a) 실험 1

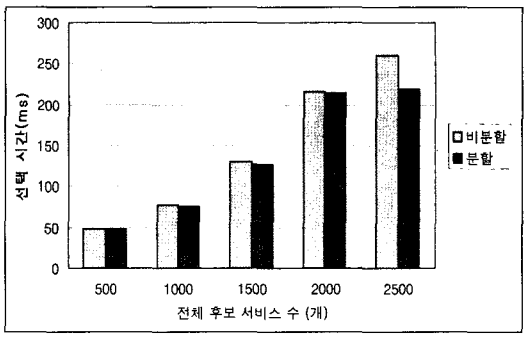


(b) 실험 2

그림 18 선택 속도 (제약조건이 3개인 경우)



(a) 실험 1



(b) 실험 2

그림 19 선택 속도 (제약조건이 4개인 경우)

방법의 복합 웹 서비스 선택 속도가 보다 빨랐다. 한편 태스크의 수와 후보 서비스의 수가 늘어남에 따라 선택에 걸리는 시간이 증가하였으며, 두 경우 모두 후보 서비스 수의 증가에 대해 기하급수적인 증가율을 보였다. 특히 태스크 및 후보 서비스의 수가 증가할 경우 더욱 큰 속도 차이를 나타내었다.

4.2.3 선택된 복합 웹 서비스의 품질

표 6과 표 7은 제안된 분할방법을 통하여 선택된 복합 웹 서비스의 QoS를 최적 QoS와 비교한 결과이다. 이때 음수는 최적 QoS보다 저하된 품질을 의미하며 반대로 양수는 최적 품질보다 향상된 경우를 의미한다. 여기서 최적 품질보다 향상되었다는 것은 각 품질 요소를 개별적으로 고려하였을 경우를 의미한다. 실험 결과, 모든 품질 요소에 있어 최적의 경우와 비교하여 5%미만의 품질 저하를 보였다.

표 6 실험 1에 대한 QoS 차이 비교

태스크 수	시간	비용	가용성	신뢰성
4	-0.56%	0.21%	-0.024%	-0.039%
8	1.34%	-2.72%	-0.13%	0.12%
12	-0.52%	-0.25%	-0.062%	-0.01%
16	-0.61%	-0.26%	-0.1%	0.07%
20	-0.34%	-0.35%	-0.05%	0.033%
평균	-0.138%	-0.674%	-0.073%	0.035%

표 7 실험 2에 대한 QoS 차이 비교

후보 서비스 수	시간	비용	가용성	신뢰성
500	-4.54%	-0.08%	-0.20%	0.12%
1000	-2.71%	1.47%	-0.10%	-0.10%
1500	-3.77%	3.18%	-0.06%	-0.33%
2000	-4.08%	4.32%	-0.18%	-0.24%
2500	0.22%	2.86%	-0.18%	-0.12%
평균	-2.976%	2.35%	-0.144%	-0.134%

5. 결론 및 향후 연구 방향

단일 웹 서비스로 제공할 수 없는 복잡한 비즈니스 프로세스에 대한 요구가 점차 증가함에 따라 사용자가 원하는 QoS를 만족하는 복합 웹 서비스를 선택하는 것이 중요하다. 한편 웹 서비스의 증가에 따른 선택 속도의 저하와 QoS가 동적으로 변하는 웹 서비스 환경을 고려할 때 주어진 QoS 요구사항을 만족하면서 적절한 QoS의 복합 웹 서비스를 빠르게 선택할 수 있는 방법이 요구된다.

본 논문은 워크플로우의 구조, QoS 요구사항, 그리고

후보 웹 서비스의 QoS를 고려하여 워크플로우를 분할하고 분할된 워크플로우에 대해 QoS 요구사항을 적절히 재설정하여 복합 웹 서비스를 빠르게 선택할 수 있는 방법을 제안한다. 제안된 방법은 분할에 의한 서비스의 선택 실패를 줄이기 위해 주어진 QoS 요구사항의 제약정도를 계산하여 제약정도가 적절한 경우에 한해 선택적으로 분할한다. 한편 복합 웹 서비스의 선택을 위해 MILP를 사용한다.

실험을 통해 제안된 서비스 선택 방법의 성공 및 재현율, 선택 속도, 그리고 선택된 복합 서비스의 품질을 평가하였다. 실험 결과, 제안된 방법은 99% 이상의 성공 확률을 보였으며 분할을 적용하지 않았을 경우에 비해 언제나 빠른 선택 속도를 나타내었다. 특히 태스크 및 후보 서비스의 증가에 대하여 좋은 확장성을 보였다. 결과 서비스의 QoS 측면에 있어서도 5% 미만의 차이를 보였다. 향후 실험에서 나타난 재현율의 저하를 개선하기 위해 더욱 정교한 제약정도 계산법을 개발하고 현재 RSG 환원 규칙이 적용될 수 없는 부분적으로 중첩된 구조에 대한 QoS 계산 방법을 자동화 된 방법과 통합시킬 수 있는 방법을 고안할 예정이다. 이와 함께 실제 비즈니스에서 사용되는 복잡한 워크플로우를 대상으로 후보 웹 서비스의 QoS가 동적으로 변하는 환경에 대한 실험을 수행하여 제안된 방법의 실용성을 검증하는 실험도 함께 수행할 계획이다.

참 고 문 헌

[1] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver, Combinatorial Optimization, p. 355, John Wiley & Sons, New York, 1998.

[2] L. A. Wolsey, G. L. Nemhauser, Integer and Combinatorial Optimization, 1st Ed., p. 763, Wiley-Interscience, 1999.

[3] S. Ran, "A model for web services discovery with QoS," ACM SIGecom Exchanges, Vol. 4, No. 1, pp. 1-10, 2003.

[4] S. Kalepu, S. Krishnaswamy, and S.W. Loke, "Verity: a QoS metric for selecting web services and providers," Proc. 4th Int'l Conf. WISE Workshops, pp. 131-139, 2003.

[5] E. M. Maximilien and M. P. Singh, "Conceptual Model of Web Service Reputation," ACM SIGMOD Record, Vol. 31, No. 4, pp. 36-41, 2002.

[6] M. Tian, A. Gramm, T. Naumowicz, H. Ritter, and J. Schiller, "A concept for QoS integration in web services," Proc. Int'l Conf. 4th WISE, pp. 149-155, 2003.

[7] A. S. Bilgin and M. P. Singh, "A DAML-based repository for QoS-aware semantic web service selection," Proc. Int'l Conf. IEEE Web Services, pp. 368-375, 2004.

[8] E. M. Maximilien and M. P. Singh, "A Framework and Ontology for Dynamic Web Services Selection," Internet Computing, Vol. 8, No. 5, pp. 84-93, 2004.

[9] A. Gao, D. Yang, S. Tang, and M. Zhang, "QoS-Driven Web Service Composition with Inter Service Conflicts," Proc. 8th APWeb Conf. (LNCS 3841), pp. 121-132, 2006.

[10] Y. Chen, Z. Li, Q. Jin, and C. Wang, "Study on QoS Driven Web Services Composition," Proc. 8th APWeb Conf. (LNCS 3841), pp. 702-707, 2006.

[11] S. Liu, Y. Liu, N. Jing, G. Tang, and Y. Tang, "A Dynamic Web Service Selection Strategy with QoS Global Optimization Based on Multi-objective Genetic Algorithm," Proc. 4th Int'l Conf on Grid and Cooperative Computing (LNCS 3795), pp. 84-89, 2005.

[12] L. Yang, Y. Dai, B. Zhang, and Y. Gao, "Dynamic Selection of Composite Web Services Based on a New Structured TCNN," Proc. IEEE Int'l Workshop on Service-Oriented System Engineering, pp. 149-158, 2005.

[13] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "An Approach for QoS-aware Service Composition based on Genetic Algorithms," Proc. Conf. Genetic and Evolutionary Computation, pp. 1069-1075, 2005.

[14] R. GrØnmo and M. C. Jaeger, "Model-Driven Methodology for Building QoS-Optimized Web Service Compositions," Proc. 5th Int'l Conf. on DAIS (LNCS 3543), pp. 68-82, 2005.

[15] T. Yu and K. Lin, "Service selection algorithms for web services with end-to-end QoS constraints," Proc. IEEE Int'l Conf. e-Commerce Technology, pp. 129-136, 2004.

[16] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of Service for Workflows and Web Service Processes," Web Semantics Journal, Vol. 1, No. 3, pp. 281-308, 2004.

[17] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," IEEE Trans. on Software Engineering, Vol. 30, No. 5, pp. 311-327, 2004.

[18] O. H. Ibarra, and C.-E. Kim, "Fast approximation algorithms for the knapsack and sum of subset problems," Journal of the ACM, Vol. 22, No. 4, pp. 463-468, 1975.

[19] M. Pinedof, Scheduling: Theory, Algorithms, and Systems, 2nd Ed., p. 586, Prentice Hall, 2001.

[20] C. L. Hwang and K. Yoon, "Multiple Attribute Decision Making," Lecture Notes in Economics and Mathematical Systems, Vol. 186, 1981.

장 재 호

2004년 2월 성균관 정보통신공학부 졸업(학사). 2006년 8월 연세대학교 컴퓨터과학과 졸업(석사). 2006년 9월~현재 TMAX soft 연구원. 관심분야는 Internet Computing, Web Services QoS

신 동 훈

2003년 2월 연세대학교 컴퓨터과학과 졸업(학사). 2005년 2월 연세대학교 컴퓨터과학과 졸업(석사). 2005년 3월~현재 연세대학교 컴퓨터과학과 박사과정. 관심분야는 Internet Computing, Web Services Composition

이 경 호

1995년 2월 연세대학교 전산과학과 졸업(학사). 1997년 2월 연세대학교 컴퓨터과학과 졸업(석사). 2001년 2월 연세대학교 컴퓨터과학과 졸업(박사). 2001년 3월 National Institute of Standard and Technology(NIST) 객원연구원. 2002년~현재 연세대학교 컴퓨터과학과 부교수. 관심분야는 Internet Computing, Service-Oriented Computing, Multimedia *Document Engineering*