Journal of the Society of Korea Industrial and Systems Engineering
Vol, 29, No. 1, pp.1－8, March 2006.

워크플로우 완료시간 최소화를 위한 실시간 자원할당 알고리즘†

윤상흠* · 신용승**

*영남대학교 경영학부
**(주)SIMedia

A Real-time Resource Allocation Algorithm for Minimizing the Completion Time of Workflow

Sang-Hum Yoon* · Yong-Seung Shin**

*School of Management, Yeungnam University
**(주)SIMedia Co. Ltd

This paper proposes a real-time resource allocation algorithm for minimizing the completion time of overall workflow process. The jobs in a workflow process are interrelated through the precedence graph including Sequence, AND, OR and Loop control structure. A resource should be allocated for the processing of each job, and the required processing time of the job can be varied by the resource allocation decision. Each resource has several inherent restrictions such as the functional, geographical, positional and other operational characteristics. The algorithm suggested in this paper selects an effective resource for each job by considering the precedence constraint and the resource characteristics such as processing time and the inherent restrictions. To investigate the performance of the proposed algorithm, several numerical tests are performed for four different workflow graphs including standard, parallel and two series-parallel structures. In the tests, the solutions by the proposed algorithm are compared with random and optimal solutions which are obtained by a random selection rule and a full enumeration method respectively.

Keywords : workflow, resource allocation, scheduling

1. 서 론

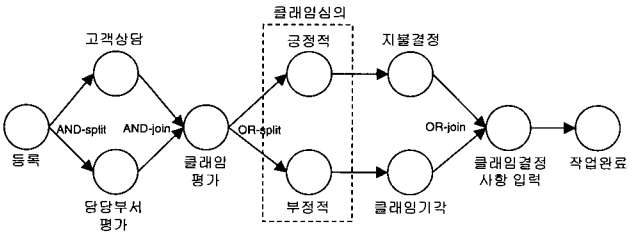
기업의 조직 규모가 방대해지고 업무 프로세스의 복잡도가 증가함에 따라 인터넷 인프라와 정보기술을 기반으로 하여 업무처리의 자동화와 유연성 향상을 지원할 수 있는 업무 프로세스 설계 및 통제도구의 필요성이 증가하고 있다. 또한, 이기종 시스템들이 통신망을 통해 서로 연결되어 기업 간의 통합 프로세스를 처리할 수 있는 분산시스템 환경이 보편화되고 있다. 이렇게 변화된 업무환경에 유연하게 대처하면서 작업프로세스의 효율적인 설계 및 관리를 위한 자동화 및 전산화 도구로써 워크플로우(workflow)와 워크플로우 관리 시스템이 등장하게 되었고 워크플로우 시스템이 가져야 할 공통적인 기능과 제품간의 상호 호환성을 위한 표준을 정의하기 위해 워크플로우 관리 연합체(WfMC: Workflow Management Coalition)가 설립되어 워크플로우 기능 향상을 위한 다양한 표준안들이 활발히 제시되고 있다.

워크플로우에 대한 최근 연구 동향을 살펴보면 워크플로우의 진행과정에서 발생할 수 있는 예외상황에 대한 처리(exception handling) 및 프로세스의 동적변경

† 이 논문은 2005학년도 영남대학교 학술연구조성비 지원에 의한 것임

(dynamic change)에 대하여 유연한 대처방안을 제공하는 고기능 워크플로우 아키텍쳐에 대한 설계(참고문헌 [1] 참조)와 이를 지원할 수 있는 소프트웨어 개발에 관한 연구가 진행되고 있으며 한편에서는 실제 발생된 워크플로우 인스턴스의 특성(실행경로, 인스턴스의 수, 각 인스턴스의 수행시간 등)을 고려한 효율적인 자원할당 및 관리와 진도관리 등의 워크플로우 통제방안에 대한 연구(참고문헌 [2],[4],[6] 참조)가 진행되고 있다. 본 논문에서는 워크플로우 설계 이후 실제 수행단계에서 필요한 시스템 통제분야에 초점을 맞추고 제한된 자원제약 하에서 워크플로우를 효율적으로 수행하기 위한 자원할당 알고리즘을 제시하고자 한다.

워크플로우는 상호 연결된 일련의 작업들로 구성되며 작업들간의 연결구조는 부분적으로 순차구조, AND구조, OR구조, Loop구조 등이 혼재하는 우선순위 그래프를 형성한다. <그림 1>은 보험 클래임의 처리 과정을 나타낸 것으로 워크플로우의 특정 인스턴스가 발생하면 등록작업을 시작으로 클래임에 대한 평가작업 이후 관련조치가 이루어지면서 워크플로우는 종료된다[7]. 여기서 고객상담 작업과 담당부서평가 작업은 AND구조로 병렬로 작업을 수행하게 되고 클래임평가 작업을 통하여 클래임심의 단계로 수행된다. 클래임심의 작업은 클래임평가 작업 이후의 OR-Split 구조에 따른 인스턴스 평가에 따라 긍정적일 경우에는 지불결정 작업을 수행하게 되고 부정적일 경우에는 보험 클래임을 기각하게 된다.

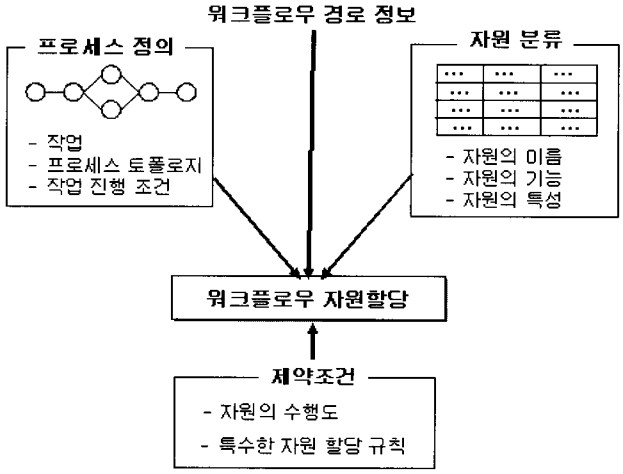


<그림 1> 보험 클래임 처리 과정의 워크플로우

워크플로우 상의 각 작업에 대한 자원할당은 <그림 2>에서 도시된 바와 같이 워크플로우 프로세스 정의 단계와 자원 분류 단계, 각 자원의 작업수행도, 특별한 자원할당 규칙 등의 사항을 바탕으로 이루어진다. 우선, 워크플로우 프로세스 정의 단계에서는 프로세스에 포함된 각 작업에 대한 성격, 수행소요시간을 포함하는 작업 고유정보와 작업간 우선순위 관계를 나타내는 프로세스 토폴로지(Topology), 각 작업에서 연계된 후속 작업으로의 진행조건(Firing Condition) 등이 정의된다. 다음으로 자원 분류 단계에서는 활용 가능한 각 자원의 기능적인 역할과 조직에서의 부서나 직위와 같은 자원의 특성에 따라 자원을 분류하게 된다.

본 연구의 목적은 프로세스 정의에서부터 시작되어 자원분류, 경로정보, 자원할당 제약조건 정의로 이어지는 워크플로우 설계단계의 각종 정보를 바탕으로 실제 워크플로우의 총 소요시간(또는, 워크플로우 완료시간)을 단축할 수 있는 효과적이고 효율적인 자원할당 알고리즘을 개발하는데 있다.

워크플로우의 완료시간은 워크플로우 상의 전체 작업을 모두 처리하는 데 소요되는 시간을 의미하며 각 단위작업에 대한 자원할당에 따라 전체 완료시간이 변하게 된다. 즉, 자원할당 결과에 따라 작업준비시간(Setup Time), 작업소요시간(Job Processing Time), 자원 수행도(Resource Performance) 등과 같은 시간요소가 변하게 되고 결과적으로 전체 워크플로우의 완료시간에 영향을 미치게 되는 것이다.



<그림 2> 워크플로우 설계단계와 자원할당의 관계

만약 워크플로우의 완료시간이 늦어지면 그에 따른 보상처리(Compensating Processing)로 인하여 추가적인 비용이 발생할 뿐만 아니라 부가적인 시스템 과부하와 시스템 자원을 낭비하게 되어 결국 워크플로우 시스템 성능의 저하를 초래한다[3][5]. 따라서 효율적인 자원 할당은 워크플로우의 처리 성능을 향상시키고 소요자원에 대한 비용을 줄이는 효과가 있다. 그러므로 워크플로우 처리의 성능 개선을 위해서는 워크플로우의 완료시간을 최소화하는 알고리즘의 개발이 요구된다.

본 논문에서는 워크플로우에서 자원의 다양한 제약사항과 실시간으로 변하는 시간적인 요소를 반영하는 자원할당 문제를 다루고 있다. 이를 위해 작업 및 자원의 수행시간을 바탕으로 하여 기존의 병렬기계에 대한 작업할당규칙을 기반으로 하여 작업준비시간과 자원수행도 등을 함께 고려한 실시간 자원할당 알고리즘을 제

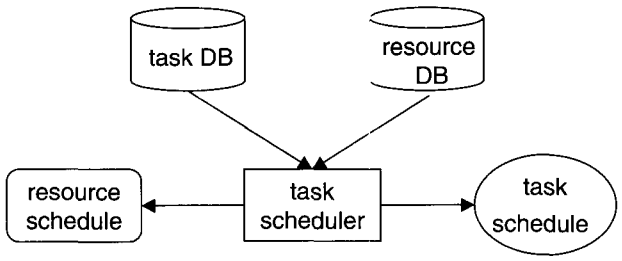
시하였다. 제시된 알고리즘은 발생된 워크플로우 인스턴스의 전체 소요시간을 줄이는 효과 뿐 아니라 자원의 활용도(Utilization)를 높이는 효과도 제공하게 된다.

본 논문은 구성은 다음과 같다 먼저, 2장에서는 워크플로우 완료시간 최소화를 위한 자원할당 문제에 대해 기술한다. 3장에서는 자원할당 문제에 대한 실시간 자원할당 알고리즘 제시하고 다양한 실험을 통해 성능평가 결과를 제시하였다. 마지막으로 4장에는 결론 및 향후 연구방향에 대해 기술한다.

2. 워크플로우의 자원할당 문제

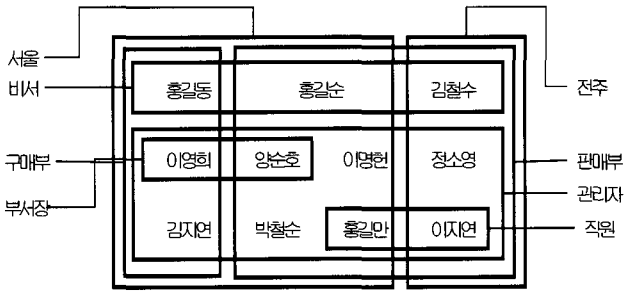
2.1 워크플로우 자원할당 구조

일반적으로 워크플로우 상의 각 작업의 수행을 위해서는 자원의 지원이 필요하며 각 작업의 수행과 자원할당시에 고려해야 할 다양한 제약사항에 대한 자료는 <그림 3>에서와 같이 작업DB와 자원DB를 통해 통합 관리된다. 작업DB에는 워크플로우 설계단계에서 프로세스 정의 과정을 통해 정의된 각 작업별 특성을 담고 있다. 여기에는 작업간 우선순위구조를 반영한 프로세스 라우팅자료와 각 작업의 수행시간을 포함한 작업완료조건 등의 작업특성들이 포함되어 있다. 자원DB에는 활용 가능한 전체 자원리스트와 각 자원의 기능적, 지역적, 직위적 특성을 반영하여 자원을 분류한 자원 분류 자료가 포함되어 있다. 워크플로우의 진행과정은 워크플로우 엔진에 의해 제어되고 모니터링 된다. 특정 시점에서 수행해야 할 작업에 대한 결정과 그 작업에 대한 자원할당은 엔진내의 작업스케쥴러에 의해 실시간으로 수행되며 작업스케쥴러는 작업DB와 자원DB에서 제공하는 자료를 참조하여 이를 결정한다.



<그림 3> 워크플로우 자원할당 구조

각 작업에 대한 자원할당은 <그림 4>과 같은 자원 분류결과에 기초하여 이루어진다. 즉, 자원활용가능 지역에 따라 서울과 전주와 같이 지역적으로 분류할 수 있고 구매부나 판매부와 같이 소속 조직에 따라 분류할 수 있다. 그리고 부서장과 직원과 같이 직위에 따라 분류될 수도 있고 비서와 관리자와 같이 자원의 기능적인 역할에 따라 분류가 가능하다. 이러한 자원 분류 자료는 자원DB에 의해 관리되며 워크플로우 프로세스 정의 단계에서 정의된 작업DB 내의 작업특성정보와 함께 작업스케쥴러에 의해 참조되고 본 논문에서 제시되고 있는 자원할당 결과에 따라 해당 작업의 작업스케쥴과 그 작업에 할당된 자원의 자원스케쥴이 갱신된다[7].

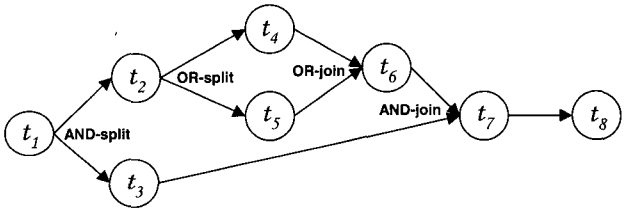


<그림 4> 자원분류의 예

2.2 자원할당 문제의 구성

본 논문에서 고려하고 있는 자원할당문제는 워크플로우의 완료시간을 최소화할 수 있도록 작업스케쥴러가 특정시점에서 수행해야 할 작업에 대한 결정과 그 작업에 대해 특정 자원을 할당하기 위한 규칙을 제공하는 것이다. 이를 위해 본 논문에서는 다음과 같은 문제특성 및 가정을 내재하고 있다.

첫째, 워크플로우 프로세스설계를 통해 <그림 5>에서 예시된 바와 같이 작업간 우선순위 그래프를 미리 알고 있으며 프로세스 진행과정에서 제어구조의 인스턴스에 따른 실제 수행되는 작업리스트는 선행작업이 완료되는 시점에서 알 수 있다. 예를 들어, <그림 5>에서 작업 t_2가 완료되는 시점에서 OR-Split 인스턴스의 평가에 의해 작업 t_4가 수행되는지 작업 t_5가 수행되는지가 결정된다.



<그림 5> 워크플로우 우선순위 그래프

둘째, 특정 작업의 수행가능 시점에서 작업스케쥴러는 작업 및 자원 DB에서 정의된 특성 값을 참조하여 그 작업의 고유작업시간과 그 작업을 실제 수행할 수 있는 자원리스트를 획득할 수 있다. 예를 들어, <표 1>에서 예시된 바와 같이 t_2는 자원 a 또는 자원 c에 의해서 수행가능하며 고유작업시간은 2시간이다.

셋째, 우선순위관계에서 연결된 두 개의 작업을 각각 다른 자원이 수행할 시에는 후속작업에 대해 정해진 준비시간이 부과된다. 예를 들어, <그림 5>에서 작업 t_3과 t_7이 같은 자원에 의해 연속적으로 수행된다면 준비시간이 필요하지 않으나, 각각 다른 자원에 의해 수행될 시에는 후속작업 t_7에 대해 정해진 준비시간이 부과된다. 또한, t_6과 t_7의 관계도 같은 논리가 성립되고, 만약 t_3, t_6, t_7이 모두 다른 자원에 의해 수행된다면 t_7의 작업에는 두 번의 준비시간이 부과된다.

넷째, 각 작업별 소요 작업시간은 자원할당결과에 따라 변할 수 있다. 이는 작업스케쥴러가 <표 2>에 예시된 바와 같은 자원 DB에서 제공하는 업무수행도 자료를 활용하여 소요작업시간을 계산한다. 즉, 작업 t_i의 고유작업시간이 p_i이고 t_i에 자원 r_j를 할당했을 때의 업무수행도 지수를 w_{ij}라고 할 때 작업 t_i를 수행하는 데 필요한 실제 작업수행시간은 p_i/w_{ij} 로 계산된다.

<표 1> 고유 작업시간 및 할당 가능 자원 리스트의 예

작업(t_i)	고유작업시간(p_i)	할당가능자원(r_j)
1	3	b, c
2	2	a, c
3	4	a
4	3	a, b
5	3	a, b
6	5	b, c
7	4	a, b
8	2	a, b, c

<표 2> 작업별 자원할당에 따른 업무수행도의 예

작업 (t_i)	자원 a	자원 b	자원 c
1	0	1.1	1.2
2	1	0	1.3
3	1	0	0
4	1.2	1	0
5	1.2	1	0
6	0	1.2	1
7	1.3	1	0
8	1	1	1.2

향후 자원할당 알고리즘의 도출과정의 설명을 돕기 위해 다음과 같은 기호를 사용한다.

- n : 워크플로우 상의 전체 작업의 수
- m : 워크플로우 상의 활용가능한 자원의 수
- Δ : 두 개의 연결된 작업들을 서로 다른 자원이 수행할 시에 후속작업에 발생되는 작업준비시간
- k : 선행작업과 후속작업이 다른 자원에 할당될 때 후속작업에 부과될 준비시간의 횟수
- σ : 현재 시점에서 수행 가능한 작업리스트
- δ_i : 작업 t_i에 대해 할당 가능한 자원리스트
- $C(t_i)$: 작업 t_i의 완료시간
- $\overline{C}(t_i)$: 작업 t_i의 선행작업들의 최대 완료시간, 즉 $\overline{C}(t_i)=\max_{t_k\in\beta_i}\{C(t_k)\}$로 계산되며, β_i는 작업 t_i에 대한 선행 작업리스트(집합)을 나타냄
- p_i : 작업 t_i의 고유작업시간
- w_{ij} : 작업 t_i에 자원 r_j를 할당했을 때의 업무수행도 지수
- $C(r_j)$: 현재 시점에서 자원 r_j가 수행 완료한 마지막 작업의 완료시간

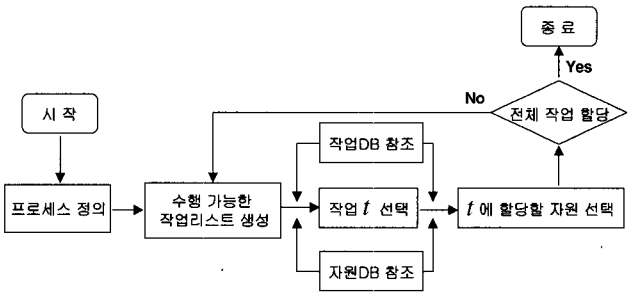
3. 워크플로우의 실시간 자원할당 알고리즘

3.1 실시간 자원할당 알고리즘

워크플로우의 완료시간을 줄이기 위해서는 다음 같은 원리를 적용할 수 있다. 첫째, 우선순위 관계를 위반하지 않는 범위 내에서 작업들은 가능한 병렬(Parallel)로 수행되어야 한다. 서로 독립적으로 할당 가능한 작업들을 같은 자원에 의해 순차적으로 수행하는 것보다는 서로 다른 자원에 의해 병렬로 처리하는 것이 자원활용도를 높임으로써 워크플로우의 완료시간을 줄이는 효과가 있다. 둘째, 서로 연결된 작업의 경우에는 준비시간을 줄이기 위해 가능한 같은 자원에서 순차적으로 처리한다. 하지만, 첫 번째의 병렬처리원리와 두 번째 순차처리원리는 상호 배반 적인 요소를 내포하고 있으며 이를 적절히 조화하기 위해서는 현재 시점에서 각 자원의 활용상태 즉, $C(r_j)$값을 실시간으로 모니터링 함으로써 해결할 수 있다. 마지막으로, 현재 시점에서 여러 개의 작업들이 할당 가능한 경우에는 고유작업시간이 긴 작업, LPT를 가지는 작업을 우선적으로 할당한다.

위에서 설명된 세 가지 원리를 반영하여 순서도를 통

해 실시간 자원할당 알고리즘을 단계별로 표현하면 <그림 6>과 같다. 프로세스 정의를 통해 워크플로우 그래프가 주어진 상황에서 선행 작업들이 이미 종료된 작업리스트 σ를 확보한 후 σ에 속한 작업 중에서 실제 자원을 할당할 작업 t를 작업수행도 및 고유작업시간 등을 고려하여 선택하게 되고 그 작업에 특정자원을 할당할 시에 발생되는 실제 작업완료시간을 바탕으로 가장 유리한 자원을 할당하게 된다. 이와 같은 작업선택과 자원할당의 반복과정이 전체 작업에 대해 수행되면 알고리즘은 종료된다.



<그림 6> 실시간 자원할당 알고리즘 순서도

실시간 자원할당 알고리즘을 단계별로 기술하면 다음과 같다.

(실시간 자원할당 알고리즘)

[step 1] 워크플로우 우선순위 그래프를 생성

[step 2] 현 시점에서 수행 가능한 작업리스트 σ를 구함

[step 3] $t_i \in \sigma$ 인 각 작업 t_i에 대해 할당 가능한 자원리스트 δ_i를 선택

- σ의 요소작업이 하나일 경우 step 5를 수행
- σ의 요소작업이 다수일 경우 step 4을 수행한 후 step 5를 수행

[step 4] $t_i \in \sigma$ 인 각 작업 t_i에 대해

$$y_i = \min_{r_j \in \delta_i} \{p_i / w_{ij}\}$$

를 계산하고 y_i가 최대인 t_i를 선택

[step 5] 작업 t_i에 대한 선행작업들의 완료시간 $\overline{C}(t_i)$와 현 시점의 자원 완료시간 $C(r_j)$ 중 큰 시간과 준비시간, 작업수행시간을 합한 값 중 작은 값을 다음과 같이 계산하여 작업 t_i의 완료시간 $C(t_i)$에 할당

$$\alpha_j = \min_{r_j \in \delta_i} \{\max\{C(r_j), \overline{C}(t_i)\} + k \cdot \Delta + p_i / w_{ij}\}$$

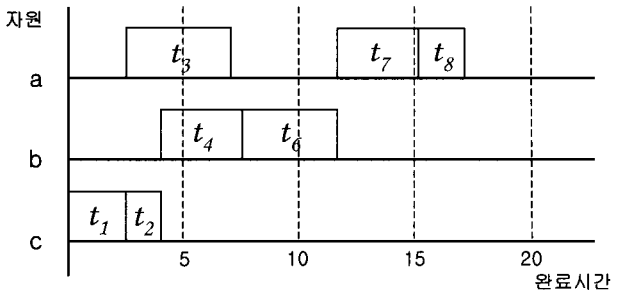
[step 6] 자원의 완료시간 $C(r_j)$에 α_j를 대입

[step 7] 모든 작업이 할당될 때까지 step 2-6을 반복

위와 같이 설명된 알고리즘을 <그림 5>의 워크플로우 우선순위 그래프를 예로 들어 설명하고자 한다. 고유 작업시간과 할당 가능한 자원리스트, 작업별 자원할당에 따른 업무수행도가 <표 1>, <표 2>와 같고 워크플로우 상의 전체 작업 수 n을 8, 활용 가능한 자원의 수 m은 3, 작업준비시간 Δ를 0.5라고 한다면 제안된 알고리즘에 의한 단계별 자원할당 결과를 요약하면 <표 3>과 같고 이를 간트스케쥴(Gantt Schedule)로 표현하면 <그림 7>과 같다. 결과적으로 자원별 완료시간은 자원 a는 17.1이고 자원 b는 11.6 자원 c는 4가 되며 전체 워크플로우의 완료시간은 17.1이 된다.

<표 3> 단계별 자원할당 결과

stage	작업리스트	할당작업	자원리스트	할당자원
1	t_1	t_1	b, c	c
2	t_2, t_3	t_3	a	a
3	t_2	t_2	a, c	c
4	t_4	t_4	a, b	b
5	t_6	t_6	b, c	b
6	t_7	t_7	a, b	a
7	t_8	t_8	a, b, c	a

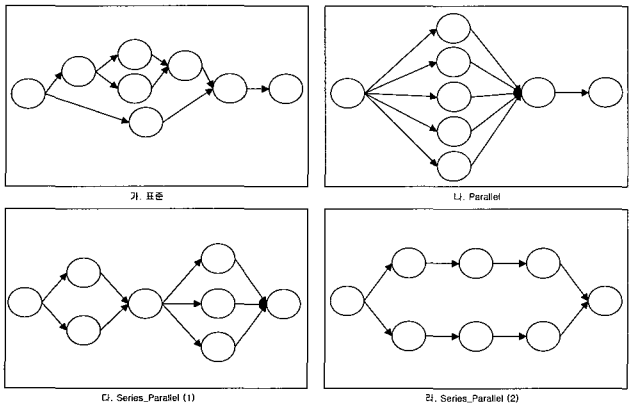


<그림 7> 자원할당결과에 따른 간트스케쥴

3.2 성능 분석

제시한 실시간 자원할당 알고리즘의 성능분석을 위해 본 논문에서는 <그림 8>과 같이 AND 구조와 OR 구조

가 혼재한 네 가지 종류의 우선순위 그래프에 대해 실험을 진행하였다. 각 그림은 표준형태, 병렬성을 강조한 Parallel형태, 마지막으로 직렬과 병렬이 혼합된 Series_parallel(1)과 Series_parallel(2)형태를 나타낸 것이다. 본 논문에서 제시한 자원할당 알고리즘의 성능을 상대적으로 비교할 수 있는 기존의 알고리즘이 없기 때문에 주어진 제약 조건 하에서 자원을 임의로 할당한 랜덤(Random)해, 그리고 모든 가능한 작업할당 스케쥴을 고려하여 찾은(Full Enumeration) 최적해와 비교하였다. 랜덤해의 결과는 총 100번의 반복수행을 통해 얻은 완료시간들의 평균값을 적용하였다. 실험에 사용된 작업별 고유작업시간은 [11, 30] 사이에서 임의로 선택하였으며, 총 자원의 수는 3개이며 각 작업별로 할당가능 자원 리스트는 임의로 선택하였다. 작업수행도는 [1.0, 1.5]사이에서 임의로 선택하였다.



<그림 8> 우선순위 구조에 따른 그래프의 종류

<표 4> 자원할당 알고리즘의 완료시간 비교

	알고리즘	랜덤해	최적해
표 · 준	106.31 (0%)	127.79 (20.2%)	106.31
Parallel	89.07 (0.9%)	90.29 (2.3%)	88.27
Series_Parallel(1)	106.25 (6.6%)	106.52 (6.8%)	99.71
Series_Parallel(2)	96.77 (14.8%)	113.29 (34.4%)	84.27

실험의 결과는 <표 4>에 요약되어 있다. <표 4>는 각 알고리즘의 완료시간을 나타내는 것이고 괄호 안의 수치는 최적해와 비교된 상대적 오차를 나타낸 것이다. 즉, 자원할당알고리즘에 의한 완료시간을 V_{ours}라고 하고 최적해의 완료시간을 V_{opt}라고 표시할 때, 상대적 오차는 $100\times(V_{ours}-V_{opt})/V_{opt}$로 표현된다. 이는 상대적 오차가 낮을수록 최적해와 비교해 완료시간의 차이가 작고 성능이 좋은 것을 나타낸다.

<표 5>는 <그림 8>에서 제시된 네 가지의 우선순위 그래프를 각종 수행시간과 관련된 모수(Parameter)의 변화에 따른 자원할당 알고리즘과 랜덤해, 그리고 최적해를 비교해 얻은 결과를 나타낸 것이다. 준비시간은 0.5에서 3.0의 범위에서 변화를 시켰으며 작업 고유처리시간의 경우는 <표 4>의 실험에서 사용된 처리시간에 비해 상대적인 시간의 변화량을 25%에서 100%로 변화시킨 결과이다. 또한 업무수행도의 경우는 [1.0, 1.5]의 범위에서 변화가 심한 경우와 약한 경우로 나누어 실험이 진행되었다. 표준 그래프에 대해서는 모수의 변화에 따라 모든 경우에 대해 제안된 알고리즘이 최적해를 도출하였다. 하지만 시리즈 구조에 비해 병렬성이 강조된 그래프일수록 다소 나쁜 결과를 나타내고 있다. 즉, 네 가지 종류의 우선순위 그래프의 결과를 비교할 때 병렬성이 강조된 그래프의 상대적 오차가 커짐을 알 수 있다. 마지막으로 각 자원이 얼마나 활용되었는가를 알아보기 위한 자원할당 알고리즘의 상대적 자원활용도를 최적해의 자원활용도와 비교한 결과는 <표 6>과 같다. 상대적 자원활용도는 각 자원별로 준비시간을 포함한 실제소요시간을 구한 후 이중 가장 긴 시간을 작업한 자원을 기준으로(100%) 다른 자원의 소요시간의 상대적 비율을 구한 것이다. <표 6>에서 보는 바와 같이 비록 제안된 알고리즘이 워크플로우의 최종완료시간을 최소화하는 방향으로 설계되었지만, 자원활용도 측면에서도 우수함을 알 수 있다.

4. 결론 및 향후 연구방향

본 논문에서는 워크플로우에서 자원의 다양한 제약사항과 실시간으로 변화하는 시간적인 요소를 반영하고 워크플로우의 완료 시간을 최소화하기 자원할당 알고리즘을 개발하였다.

본 논문에서 제시된 자원할당방법은 중앙의 워크플로우 엔진에 의해 해당 시점에서 각 자원이 수행해야 할 작업들을 강제로 제시하는 Push 방식을 원칙으로 한 것으로서 각 자원이 자신이 수행할 작업들을 선택하는 Pull 방식과는 구별된다. 또한, 본 논문에서는 작업수행을 위한 자원의 종류가 한 개인 경우만을 대상으로 하고 있으나 다양한 종류의 자원을 대상으로 한 연구가 필요하며 작업 수행시간의 경우도 확정적인 경우 뿐 아니라 확률적인 상황을 고려하는 연구가 필요하다.

<표 5> 제약 조건 변화에 따른 비교

그 래 프	기 준	비교방법	자원할당 알고리즘	랜덤한 방법	최적해
표 준	준 비 시 간 변 화	0.5	101.81	120.29 (18.6%)	101.81
		1	103.31	122.79 (18.8%)	103.31
		3	109.14	132.79 (21.6%)	109.14
	처 리 시 간 변 화	25%	29.79	36.73 (23.2%)	29.79
		50%	55.57	66.86 (20.3%)	55.57
		75%	81.23	95.25 (17.2%)	81.23
	업무수행도변화	심한경우	97.43	116.34 (19.4%)	97.43
		약한경우	103.84	119.01 (14.6%)	103.84
Parallel	준 비 시 간 변 화	0.5	84.57 (4.5%)	85.71 (5.9%)	80.94
		1	86.07 (3.8%)	88.01 (6.2%)	82.91
		3	92.07 (0.4%)	94.91 (3.5%)	91.74
	처 리 시 간 변 화	25%	26.77 (0%)	30.11 (12.5%)	26.77
		50%	47.53 (0%)	50.93 (7.2%)	47.53
		75%	68.30 (0.1%)	71.78 (5.2%)	68.20
	업무수행도변화	심한경우	83.14 (5.1%)	91.65 (15.8%)	79.14
		약한경우	88.44 (1.1%)	89.51 (2.3%)	87.49
Series_ parallel(1)	준 비 시 간 변 화	0.5	98.75 (7.1%)	99.61 (8.0%)	92.21
		1	101.25 (6.9%)	101.96 (7.7%)	94.71
		3	111.25 (6.2%)	111.37 (6.4%)	104.71
	처 리 시 간 변 화	25%	25.56 (2.5%)	31.99 (28.3%)	24.93
		50%	50.62 (6.9%)	56.79 (19.9%)	47.35
		75%	82.18 (6.3%)	82.35 (15.1%)	77.28
	업무수행도변화	심한경우	97.08 (8.8%)	111.17 (24.6%)	89.25
		약한경우	102.10 (8.7%)	105.69 (12.6%)	93.85
Series_ parallel(1)	준 비 시 간 변 화	0.5	92.27(16.3%)	106.76 (33.8%)	79.77
		1	93.77(15.4%)	108.92 (34.0%)	81.27
		3	99.77(14.3%)	117.66 (34.8%)	87.27
	처 리 시 간 변 화	25%	24.19(14.8%)	34.89 (65.5%)	21.07
		50%	46.88(15.4%)	61.02 (50.2%)	40.63
		75%	69.58(15.6%)	87.15 (44.8%)	60.2
	업무수행도변화	심한경우	85.18(19.0%)	113.27 (58.3%)	71.55
		약한경우	91.69(15.8%)	110.33 (39.3%)	79.19

<표 6> 자원활용도의 비교

그 래 프	자원할당알고리즘			최 적 해		
	자원 a	자원 b	자원 c	자원 a	자원 b	자원 c
표 준	100%	98.31%	84.20%	100%	98.31%	84.20%
Parallel	100%	31.29%	91.9%	100%	38.60%	41.28%
Series_parallel(1)	52.93%	100%	93.18%	43.37%	100%	60.85%
Series_parallel(2)	100%	20.45%	54.45%	100%	65.28%	48.62%

참고문헌

[1] 김학성; "워크플로우 마이닝 프레임웍과 아키텍쳐에 관한 연구", 경기대학교 박사학위 논문, 2003.

[2] 손진현, 오석균, 이윤준, 김명호; "고성능 분산 워크플로우를 위한 선형 계획법 기반의 워크플로우 작업 할당 방법", 정보과학회논문지 : 데이터베이스,27(3) : 549-557, 2000.

[3] 손진현, 장덕호, 김명호; "워크플로우 임계 경로에 관한 분석", 정보과학회논문지 : 데이터베이스, 28(4) : 677-687, 2001.

[4] 이 헌, 박규석; "워크플로우 엔진을 위한 태스크할당 알고리즘의 설계", 한국정보과학회 2001년도 추계학술대회 발표논문집, pp. 82-84, 2001.

[5] Duk-Ho Chang, Jin Hyun Son, Myoung Ho Kim; "Critical path identification in the context of a workflow", Information and Software Technology, 44 : 405-417, 2002.

[6] Jin Hyun Son, Seok Kyun Oh, Kyung Hoon Choi, Yoon Joon Lee, Myoung Ho Kim; "GM-WTA : An efficient workflow task allocation method in a distributed execution environment," Journal of Systems and Software, 67(3) : 165-179, 2003.

[7] Wil van der Aalst and Kees van Hee; *Workflow Management*, MIT Press, 2002.