유전자 알고리즘을 활용한 효율적인 워크플로우 업무처리에 관한 연구

이승욱* · 하귀룡** · †윤상흠***

A Study on the Efficient Workflow Processing Procedure by Genetic Algorithm

Seung Wook Lee* · Gui Ryong Ha** · †Sang Hum Yoon***

■ Abstract ■

This paper considers a genetic algorithm for sequencing activities and allocating resources to reduce the overall completion time of workflow in the presence resource constraints. The algorithm provides an integrated solution for two sub-problems. The first is to decide the priority for the activities which require the same resource. The other problem is to select one among available resources for each activity by considering the incurred setup time and the performance factor of each resource. We evaluate the algorithm performance for three different kinds of workflows including parallel structures. Computational results show that the proposed algorithm is more effective than a previous work.

Keywords : Workflow, Genetic Algorithm, Activity Sequencing, Resource Allocation

논문접수일 : 2008년 03월 27일 논문게재확정일 : 2008년 09월 21일

* (주)나노텍
** 경북대학교 경영학부
*** 영남대학교 경영학부
† 교신저자

1. 서 론

현대 기업조직의 규모가 방대해지고 부서간 협업과 업무 프로세스의 복잡도가 증가함에 따라 인터넷 인프라와 정보기술을 기반으로 하여 업무처리의 자동화와 유연성을 지원할 수 있는 업무 프로세스 설계 및 통제 도구의 역할이 중요해지고 있다. 또한, 이기종 시스템들이 통신망을 통해 서로 연결되어 기업 간의 통합 프로세스를 처리할 수 있는 분산시스템 환경이 보편화되고 있다[4].

이러한 환경에서 기업의 다양한 업무프로세스를 모델링하고 통제한다는 것은 오류발생 가능성이 매우 높고 복잡하다. 더욱이 업무프로세스 내의 개별 활동들은 일반적으로 그것들을 처리하기 위한 자원이 필요하고 가용한 자원이 충분치 않을 경우 자원 제약이라는 특성이 업무 프로세스의 효율을 크게 떨어뜨리게 된다. 예를 들어 두 가지 활동이 동일한 자원을 요구하고 현재시점에서 가용한 자원의 수가 충분치 않다고 할 때, 통제 시스템이 이러한 상황에 대해 명백한 해를 제공하지 못한다면 분명 두 가지 활동 사이에서는 동일한 자원을 할당받기 위한 충돌이 발생할 것이다. 따라서 통제 시스템은 이러한 잠재적인 충돌을 예상하고 자원 제약 상황에 대한 명백한 해답을 준비해야 한다. 더 나아가 이러한 잠재적인 충돌을 완화시킬 수 있는 시스템화 된 알고리즘과 통제시스템이 필요하다[10].

이러한 요구사항을 충족시키기 위한 대안으로 워크플로우(Workflow)와 워크플로우 관리 시스템(WfMS : Workflow Management System)이 등장하였고 워크플로우 시스템이 가져야 할 공통적인 기능과 제품 간의 상호 호환성을 위한 표준을 정의하기 위해 워크플로우 관리 연합체(WfMC : Workflow Management Coalition)가 설립되어 워크플로우 기능 향상을 위한 다양한 표준안들이 활발히 제안 · 채택되고 있다[8, 9, 10].

워크플로우에 대한 최근 연구 동향을 살펴보면 워크플로우의 진행과정에서 발행할 수 있는 예외상황에 대한 처리(exception handling) 및 프로세스의 동적 변경(dynamic change)에 대하여 유연한 대처방안을 제공하는 고기능 워크플로우 아키텍처에 대한 설계와 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 관리(resource allocation and management)와 진도관리 등의 워크플로우 통제방안에 대한 연구가 진행되고 있다[1, 2, 3, 4, 5].

그러나 워크플로우에서 자원 관리가 관리시스템의 중요한 의사결정 요소라고 인식되고 있음에도 불구하고 현재까지의 연구는 워크플로우의 설계나 다양한 자원에 대한 구조화에 초점이 맞춰져 왔고, 자원제약 상황하의 실제 워크플로우를 처리하는 과정에서 개별업무에 대한 자원할당이나 업무처리 순서를 결정하는 'Resource Allocation and Activity Scheduling' 분야는 아직 많이 다루어지지 않았다[6, 10]. 이러한 자원할당 및 업무처리순서에 대한 의사결정은 워크플로우 관리시스템의 엔진개발을 위한 핵심내용이며 자원활용률(resource utilization)과 전체 워크플로우의 완료시간과 같은 핵심성과(performance)에 큰 영향을 미치게 된다.

따라서 본 연구에서는 워크플로우 설계와 자원에 대한 구조화 작업 이후에 실제 업무프로세스를 진행하는 과정에서 필요한 시스템 통제 분야에 초점을 맞추고 유전자 알고리즘(genetic algorithm)을 활용한 자원제약을 가지는 워크플로우의 효율적인 업무처리(자원할당과 업무처리 순서 결정) 알고리즘을 제시하고자 한다. 업무처리 순서 결정과 자원할당이란 '처리해야할 여러 업무 중에 어떤 업무를 먼저 처리할 것인가 그리고 그 업무를 어느 자원이 수행하게 할 것인가'를 결정하는 것이다.

워크플로우 관리시스템을 통한 업무처리과정은 다음과 같다. 먼저, 워크플로우 정의 단계에서는 워크플로우에 포함된 각 단위활동에 대한 성격과 수행소요시간(activity processing time)을 포함하

는 활동고유정보와 앞에서 언급한 프로세스 토폴로지(topology), 각 업무에서 연계된 후속업무로의 진행조건(firing condition) 등을 명시한다. 다음으로 자원 분류 단계에서는 활용 가능한 각 자원의 기능적인 역할과 조직에서의 소속 부서나 직위와 같은 자원의 개별적인 특성에 따라 자원을 분류하게 된다. 이러한 워크플로우 정의와 자원분류의 두 가지 기본 정보와 실제 발생된 워크플로우 인스턴스의 경로정보, 활동준비시간(setup time), 각 자원의 활동수행도(resource performance), 특별한 자원할당 규칙 등의 기타 고려사항을 바탕으로 워크플로우 상의 각 업무에 대한 처리순서의 결정과 자원할당이 이루어진다.

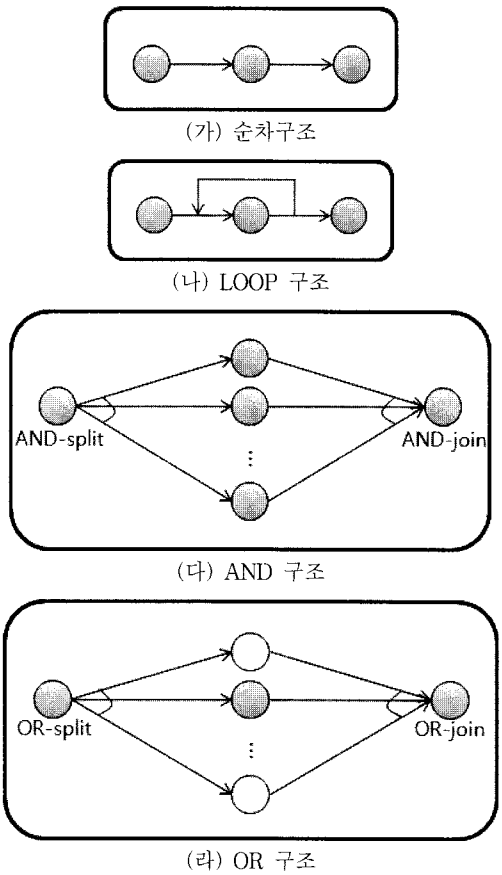
본 연구는 워크플로우 정의, 자원분류의 기본정보와 활동준비시간, 활동수행도의 부가정보를 입력으로 받아서 유전자 알고리즘을 이용한 최적의 업무처리순서와 자원할당의 두 가지 의사결정을 통합적으로 탐색하여 워크플로우 완료시간을 최소화하는 방안을 제시한다. 워크플로우 완료시간이 단축되면 개별 활동들의 유휴시간(idle time)이 줄어들게 되므로 발생된 워크플로우 인스턴스의 전체 소요시간을 줄이는 효과뿐만 아니라 자원의 활용도를 높이는 효과도 제공하게 된다.

본 연구의 구성은 다음과 같다. 제 2장에서는 본 연구에서 고려하는 문제를 보다 구체적으로 사용기호와 함께 소개하고 제 3장에서는 문제해결을 위한 유전자 알고리즘을 기술한다. 제 4장에서는 다양한 토폴로지와 모수(parameter)의 변화에 따른 유전자 알고리즘의 성능을 평가한다. 마지막으로 제 5장에서는 결론과 추후 연구과제를 기술한다.

2. 자원할당과 업무처리 순서 결정 문제

워크플로우 관리시스템이 제공하는 통제 기능은 업무프로세스의 수행시간, 노력, 비용 등을 절감시킨다. 일반적인 업무 프로세스는 여러 개의 하위 업무들로 구성되며 일련의 절차(또는, 업무간 제어구조)에 따라 수행된다. 한편 워크플로우에서는 이러한 업무 프로세스를 여러 개의 활동(activity)과 업무간 제어구조에 해당하는 워크플로우 토폴로지로 구조화한다. 워크플로우 토폴로지를 구성하는 기본적인 제어 구조에는 순차(SEQUENCE) 구조, 병행(AND) 구조, 선택(OR) 구조, 반복(LOOP) 구조가 있다(<그림 1> 참조).

순차구조는 직렬로 연속된 활동들의 집합이고, AND 구조는 AND-split 노드에 의해 여러 경로로 분지되고 개별 경로별로 다른 부분구조가 연결된다. AND 구조의 마지막에는 AND-join 노드가 위치한다. OR 구조는 OR-split 노드에 의해 분지되고 실제 인스턴스는 여러 경로들 중 어느 하나의 분지에 연결된 활동들만 수행된다. OR 구조의 마지막에는 OR-join 노드가 위치한다. 마지막으로 LOOP 구

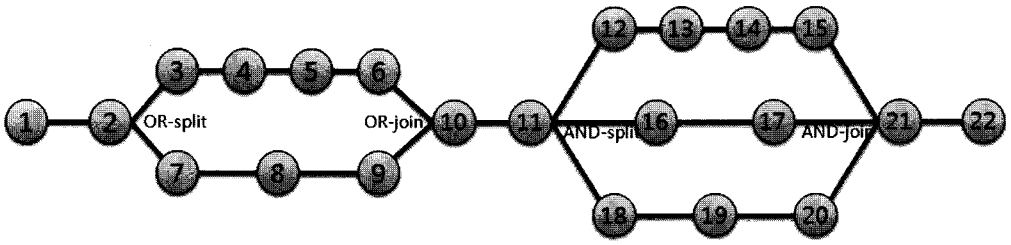


<그림 1> 워크플로우 기본 제어 구조

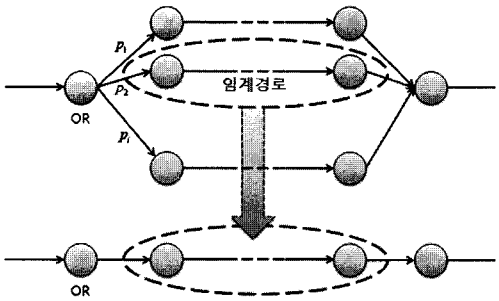
조는 위의 기본구조들의 반복적 수행을 의미한다.

본 연구에서 고려하고 있는 자원할당 및 업무처리순서 결정문제는 워크플로우의 완료시간을 최소화할 수 있도록 스케줄러가 활동의 처리순서와 그 활동을 수행한 자원을 할당하는 최적 혹은 최적에 가까운 해를 찾아내는 것이다. 이를 위해 본 연구에서는 다음과 같은 문제특성 및 가정을 내재하고 있다.

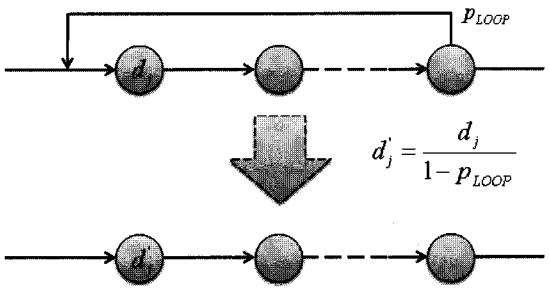
첫째, 워크플로우 설계를 통해 <그림 2-(가)>에서 예시된 것과 같이 워크플로우 토폴로지가 정해



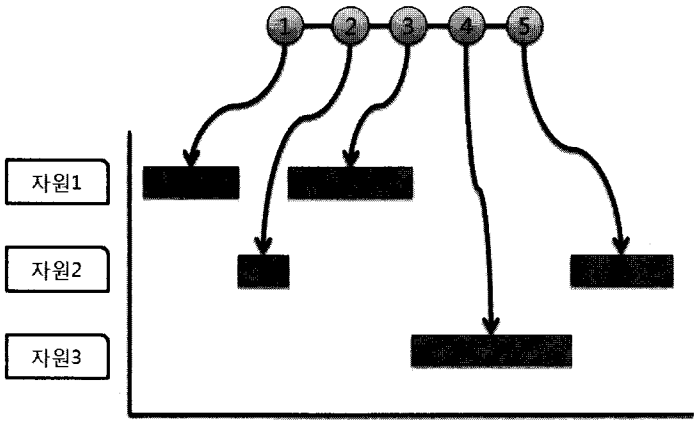
(가) 워크플로우 토폴로지



(나) OR 구조의 재구성



(다) LOOP 구조의 재구성



(라) 순차구조에서의 자원할당

〈그림 2〉 자원할당 및 업무처리순서 결정 문제특성

지고 알고리즘의 입력자료가 된다. 둘째, 본 연구는 실제 인스턴스가 발생한 이후의 실시간 처리가 아닌 사전 업무처리 자원할당을 고려하고 있으며 OR 구조와 LOOP 구조에서와 같이 업무인스턴스가 확률적으로 변동이 가능한 경우에는 사전에 이를 유사한 순차구조로 재구조화하여 분석한다. 먼저 OR 구조는 여러 경로 중 하나가 선택되는 구조이므로 가능한 경로 중에서 가장 많은 업무수와 가장 긴 수행시간을 가지는 경로(critical path, 임계경로)가 해당 OR 구조를 대표하게 한다(<그림 2-(나)> 참조). 또한 LOOP 구조는 분기율(LOOP 확률)에 따라 분기하며 그 분기율을 기준으로 순차구조로 단순화할 수 있다. 따라서 LOOP 구조에 속한 어떤 활동을 순차구조로 변경할 경우의 그 활동의 실제 소요시간은 다음과 같이 추정될 수 있다(<그림 2-(다)> 참조).

LOOP 구조 내 각 활동의 활동처리시간 :

$$\begin{aligned} d_j' &= \lim_{n\to\infty}\left(d_j + p_{LOOP}\cdot d_j + p_{LOOP}^2\cdot d_j + \cdots + p_{LOOP}^n\cdot d_j\right) \\ &= \lim_{n\to\infty}\frac{d_j(1-p_{LOOP}^n)}{1-p_{LOOP}} \\ &= \frac{d_j}{1-p_{LOOP}}, \end{aligned}$$

여기에서 d_j는 활동 j를 수행하는데 소요되는 고유업무시간을 의미하고 P_{LOOP}는 LOOP 구조의 마지막 활동을 수행한 이후에 다시 LOOP 구조상의 활동들을 반복하게 될 확률이다. 셋째, 순차구조는 선행 활동이 완료되는 시점 이후에 후행 활동을 시작할 수 있다(<그림 2-(라)> 참조). 넷째, 워크플로우에서 연속된 두 개의 활동을 각각 서로 다른

〈표 1〉 유전자 알고리즘에 사용되는 기호들

기 호	의 미
$POPsize$	모집단의 크기, 즉 모집단에 속하는 개체의 수
j	활동(activity)번호, $j = 1, 2, \cdots, N$
r	자원(resource)번호, $r = 1, 2, \cdots, R$
d_j	활동 j를 수행할 때 필요한 고유업무시간
α_{jr}	활동 j를 자원 r이 처리할 때의 활동수행도
$P_{j_1j_2}$	활동들의 선후관계 표시를 위한 이진값, j_1이 j_2보다 선행하면 1, 아니면 0
s	두 개의 연속된 활동들을 각기 다른 자원이 수행할 때 후속활동에 추가되는 준비시간
β	순차구조로 연속되는 두 활동에 할당되는 자원의 동일성 여부, 동일하면 0, 동일하지 않으면 1
C_{jr}	활동 j까지 처리했을 때의 자원 r의 완료시간(completion time)
$C_{\max}$	현재 시점에서의 전체 자원의 최종 완료시간, 즉, 만약 현재 활동 j까지 완료되었다면 $C_{\max} = \max_{r=1,\cdots,R}(C_{jr})$
l_j	활동 j가 순차구조에 속한 노드인지, AND-split 또는 AND-join 노드인지를 식별하기 위한 식별자, l_j = {0, 쌍이 되는 AND-join 활동 번호, 쌍이 되는 AND-split 활동번호의 음수값}
λ	전체 활동의 1차원 리스트
λ'	부분 문제(partial problem)에 해당하는 활동들의 부분 리스트
λ^I	개체 I에 해당되는 활동 리스트
λ_{best}	λ^I 중 가장 높은 적합도를 가지는 활동 리스트
λ_{rsc}	[단계 2]에서 λ'을 처리할 수 있는 자원 리스트

자원이 수행할 시에는 후속활동에 대해 일정한 준비시간이 부과된다. 다섯째, 각 활동별로 소요되는 업무시간은 할당된 자원의 활동 수행도에 따라 변할 수 있다. 따라서 활동 j를 자원 r이 수행할 경우 실제 소요시간은 $\alpha_{jr} \cdot d_j$가 된다. 여기서 α_{jr}은 활동 j에 대한 자원 r의 활동수행도 값이며 수행도가 높을수록 값이 작다. 이러한 개별 활동 대비 자원의 활동 수행도는 자원 DB에서 제공하는 업무수행도 자료를 활용하여 추정된다.

또한, 본 연구에서는 다음 장의 알고리즘 도출과정에 대한 설명을 돕기 위해 <표 1>의 기호들을 사용한다.

3. 유전자 알고리즘을 활용한 업무처리 알고리즘

다양한 메타 휴리스틱 방법들로부터 유전자 알고리즘을 구분하는 확고한 정의는 없다. 그러나 유전자 알고리즘이라고 불리는 대부분의 방법들은 최소한 공통적으로 가지는 요소들이 있는데, 그것은 개체 모집단 정의, 적합도(fitness)에 따른 선택과정, 새로운 자손을 생성하기 위한 교배(crossover), 그리고 새로운 자손을 위한 돌연변이이다.

유전자 알고리즘은 문제의 잠재 해(solution)를 표현한 개체들로 이루어진 모집단을 가지고 시작한다. 모집단은 각 세대마다 일정수의 개체를 유지하고 각 세대에서 각 개체의 적합도를 평가하고 이를 기준으로 다음 세대에 생존할 개체들을 확률적으로 선택한다. 선택된 개체들 중 일부의 개체들이 확률에 의해 임의로 짝을 지어 교배하여 자손 해를 생성한다. 이 때 교차에 의해 부모 해의 속성(또는 유전자)이 자손 해에게 상속된다. 교차를 통해 생성된 자손 해는 그 적합도를 평가하여 부모의 것보다 뛰어나면 모집단에 소속되어 있는 부모 해를 대체하게 된다. 자손은 부모로부터 좋은 유전형질을 상속받는다고 가정함으로써 다음 세대의 잠재해들은 평균적으로 전 세대보다 더 좋아진다고 본다. 다음으로 모집단의 개체들은 일정한 확률로 돌연변이가 발생한다. 이러한 진화과정은 정해진 종료조건을 만족할 때까지 반복한다. 위의 요소들 중 선택, 교배, 그리고 돌연변이 이 세 가지를 유전자 알고리즘의 연산자(GA Operators)라고 부른다.

3.1. 자원할당 및 업무처리순서 결정[단계 1]

본 연구에서는 알고리즘의 효율을 높이면서 위크플로우 완료시간을 줄이기 위해 다음과 같은 기본 원리를 사용한다.

첫째, OR 구조와 LOOP 구조는 사전에 순차구조로 재구성한다. 순차구조로의 재구성 방법은 앞서 제 2장에서 거론한 바와 같다. 이때, OR 구조의 경우에는 재구성된 임계경로 이외의 경로로 실제 워크플로우 인스턴스가 발생하는 경우를 대비해야 한다. 따라서, 본 연구에서는 본 절의 [단계 1]에서 일단 임계경로 위주로 초기해를 구하고, 다른 경로에 대해서는 다음 제 3.2절의 [단계 2]에서 초기해를 변경하는 방법을 사용한다. 둘째, 병렬관계에 있는 복수의 활동들은 선후관계를 유지하는 범위 안에서 서로 다른 자원에 의해 병렬로(동시에) 처리하는 것이 전체적인 자원 활용도를 높이고 완료시간을 줄이는 효과가 있다. 셋째, 순차구조의 경우 연속된 활동들의 처리에 발생되는 준비시간들을 줄이기 위해 가능한 동일한 자원이 연속된 활동들을 처리하도록 할당한다. 하지만 위에서 기술한 병렬처리 원리와 순차처리 원리는 상호배반적인 요소를 내포하고 있으므로 이를 적절히 조화시키기 위해서 적절한 처리순서를 유전자 알고리즘의 적합도 평가를 통해 찾는다. 넷째, 순차구조와 병렬처리는 구분하여 처리한다. 순차구조의 경우에는 앞의 활동이 종료한 이후에 다음 활동을 시작할 수 있으므로 활동 리스트(activity list)를 변화시킬 필요가 없다. 하지만 병렬처리의 경우 위의 두 번째 원리와 같이 다양한 처리순서가 존재하고 이 중에서 최선의 활동 리스트(처리 순서)를 탐색할 수 있으므로 병렬처리가 가능한 활동들의 집합은 이를 부분 문제(sub-problem)로 분할하고 개별적으로

유전자 알고리즘을 적용하여 최선의 해를 찾는다. 이러한 네 가지 원리를 반영하여 본 연구에서 제안하는 자원할당 및 업무처리 순서결정 알고리즘을 기술하면 다음과 같다.

〈자원할당 및 업무처리 유전자 알고리즘

[단계 1]〉

입력정보 : 워크플로우 토폴로지, 활동 기본정보, 활동 부가정보(활동 수행도, 준비시간)

(1) OR 구조와 LOOP 구조의 재구성

워크플로우 토폴로지 상의 OR와 LOOP의 부분구조가 존재하면 이를 모두 순차구조로 재구성한다.

(2) 인코딩(encoding)

(2)-1. 재구성된 워크플로우에서 활동간 선후관계 $p_{j_1 j_2}$를 인코딩한다.

(2)-2. 재구성된 워크플로우에서는 임의의 1차원 배열 λ를 생성한다. 1차원으로 배열하는 규칙은 순차관계의 경우 활동간 선후관계에 따라 배열하고 병렬관계는 선후관계를 지키는 범위 내에서 임의로 정렬한다. 예를 들어 <그림 3>의 워크플로우에서 활동 리스트를 1차원 배열로 나타낸 것을 예로 들면 λ = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)가 된다.

(2)-3. AND 관계를 구조화하기 위해 각 활동의 l_j값을 <표 1>에서 기술된 바와 같이 입력한다. 예를 들어 (2)-2번에서 기술한 λ에 대한 l_j를 표시하면 <표 2>와 같다.

〈표 2〉 l_j값의 예

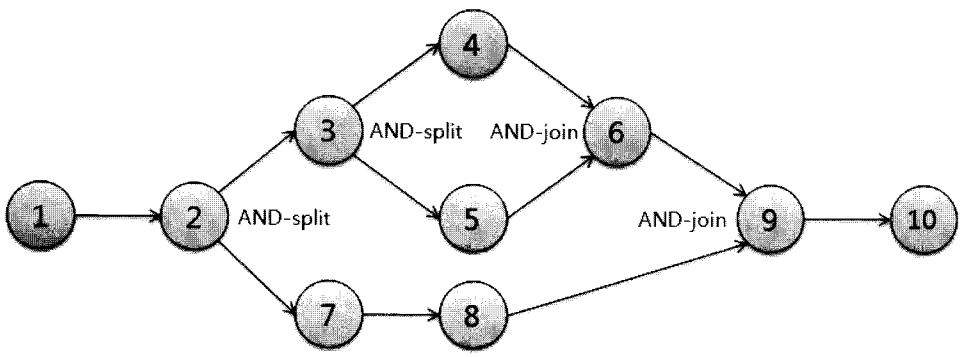
	1	2	3	4	5	6	7	8	9	10
l_j	0	9	6	0	0	-3	0	0	-2	0

(3) 활동이 순차구조일 경우의 자원할당

현재 자원을 할당해야 하는 활동 j의 l_j값을 참조하여 j가 순차구조에 속한 활동(즉, $l_j = 0$)일 경우에는 다음의 식 (1)에 의해 자원 r^*를 선택하여 할당하고 C_{jr}을 갱신한다.

$$C_{jr^*} = \min_{r \in \{1,\cdots,R\}} \begin{Bmatrix} \max\{C_{(j-1)r},\ C_{\max}\} \\ +\alpha_{jr} \cdot d_j + \beta \cdot s \end{Bmatrix} \quad (1)$$

식 (1)은 윤상흠과 신용승[4]의 연구에서 실시간 자원할당을 위해 제안된 것으로 리스트 상의 직전 활동인 $j-1$까지 처리된 최종 완료시간이 자원 r에서 발생된 경우($C_{(j-1),r}$)와 다른 자원에서 발생된 경우($C_{\max}$)를 구분하여 활동 j에 어떤 자원을 할당하느냐에 따라 발생되는 예상 완료시간을 각각 계산하고 그 값이 최소가 되는 자원 r^*를 선택



〈그림 3〉 순차구조와 AND구조로 재구성된 워크플로우의 예

한다는 의미이다.

(4) 활동이 AND-split일 경우의 처리

어떤 활동이 AND-split일 경우, 이에 대응하는 AND-join 활동을 찾아 AND 구조를 가지는 부분 문제의 범위 λ'을 정의하고 λ'에 대해 다음 (5)번부터의 유전자 알고리즘을 이용한 탐색을 수행한다. 예를 들어 <그림 3>의 경우 현재 활동 2를 처리하는 시점이라면 $l_2 = 9$이므로 활동 2와 활동 9사이의 λ' = (3, 4, 5, 6, 7, 8)이 하나의 부분 리스트가 될 수 있다.

(5) 초기 모집단의 생성

λ'을 임의의 순서로 변형하여 *POPsize* 만큼의 개체를 생성한다. 이때 활동의 선후관계 $P_{j_1 j_2}$를 고려하여 적합한 개체만을 생성한다.

(6) 자원할당과 모집단의 적합도 평가

개체해의 적합도는 그 해의 최종 완료시간을 계산하여 평가한다. 그 결과 적합도가 가장 우수한 개체해를 λ_{best}에 저장한다. 이는 적합도가 가장 뛰어난 개체가 돌연변이 할 가능성이 있기 때문이다. 해당 개체해의 활동 리스트 순서대로 활동 j에 대한 최적의 할당자원 r^*를 위의 식 (1)을 활용하여 찾는다.

(7) 선택

본 연구에서는 순위 선택 방법(ranking select method)을 사용하여 교배의 후보 해를 선택하며 개체들의 적합도 값을 기준으로 순위 선택이 이루어진다. 먼저 모집단의 적합도 값을 기준으로 순위를 부여한다. 그리고 주어진 선택확률에 따라 교배를 위한 후보 해를 가려낸다. 선별확률은 1부터 *POPsize*까지 둘 수 있는데 선택확률 이상의 순위에 해당하는 개체가 선택된다.

(8) 교배

교배를 위해 먼저 선택된 개체해들 중에서 임의로 한 쌍을 골라 부모 개체로 정한다. 본 연구에서는 알고리즘의 계산량을 줄이기 위해 단순유전자 알고리즘을 사용하였으며 Hartmann[7]에서 'Resource-Constrained Project Scheduling'문제에 대해 비교적 우수한 결과를 보이는 것으로 검증된 Two-point Crossover 방법을 사용하였다.

(9) 자손 개체의 적합도 평가

교배를 통해 생성된 자손의 해가 우선순위 관계 $P_{j_1 j_2}$를 만족하는지 평가하여 만족하면 자손 개체의 적합도를 계산한다. 만약 자손 개체의 적합도가 부모 개체보다 뛰어나면 모집단에서 부모 개체를 자손 개체로 대체하고, 그렇지 않다면 자손 개체를 제거한다. 그리고 현재까지 가장 뛰어난 적합도를 가진 개체 λ_{best}와도 비교하여 새로 생성된 자손 개체가 더 뛰어날 경우 λ_{best}를 갱신한다.

(10) 돌연변이

돌연변이는 부분 최적화(local optimum)를 방지하기 위한 도구이다. 모집단에서 각각의 개체에 대해 [0, 1] 범위의 난수를 발생한 다음 $p_{mutation}$보다 작은 값을 가지는 개체들에 대해 돌연변이를 적용한 후 각 활동의 선후관계가 적합하다면 모집단에서 해당 개체를 갱신한다. 본 연구에서는 Hartmann [7]에서 사용한 돌연변이 방법을 채택하였다. 즉, $p_{mutation}$을 통해 선별된 개체에서 i번째와 $i+1$번째 활동의 위치를 서로 바꿔서 새로운 개체를 형성하는 방식을 사용한다.

(11) 모집단의 적합도 평가

돌연변이 한 개체들에 대해 적합도를 평가한 후 λ_{best}와 비교하여 그 적합도가 높으면 λ_{best}를 갱신한다.

(12) 종료 조건과의 비교

부분 문제는 미리 정해 둔 GEN(generation의 단계 수) 혹은 시간을 기준으로 종료 여부를 판단한다. 종료 조건에 해당하지 않으면 (7)번의 '선택' 단계부터 반복 실행한다. 종료 조건을 만족하면 현재 저장된 최적의 활동 리스트(λ_{best})를 부분 문제의 해로 등록하고 현재 시점에서의 C_{jr}을 갱신한다.

(13) 전체 알고리즘의 종료

워크플로우의 모든 부분 문제에 대한 해가 확정되고 워크플로우의 모든 활동에 대한 활동순서와 자원할당이 정해지면 마지막 $C_{\max}$ 값이 최종 워크

플로우 완료시간이 되고 남은 활동이 있다면 그 활동의 성격에 따라 (3)번 또는 (4)번 단계 부터 반복 실행한다. 유전자 알고리즘은 임의로 생성한 개체들의 영향을 크게 받는다. 즉, 생성된 개체에 따라 부분 최적화가 될 수 있는데 이를 해결하기 위해 본 연구는 전체 알고리즘을 1,000회 반복하여 가장 뛰어난 완료시간과 해당하는 활동순서 및 자원할당의 해를 찾은 다음 알고리즘을 종료한다.

3.2 나머지 OR구조의 자원할당[단계 2]

OR 구조는 OR-split 활동의 처리 결과에 따라 다양한 경로가 존재하며 그 경로에 따라 업무처리 순서와 자원할당이 달라질 수 있다. 따라서 본 연구에서는 [단계 1]에서 고려한 임계경로 위주의 해 이외에 다른 경로로 실제 인스턴스가 발생할 경우를 위한 해를 준비한다. 이때 [단계 1]에서 결정된 워크플로우 전체 업무처리순서는 유지하면서 새로 대체된 OR 경로 내의 활동들에 대해서만 새로운 자원 할당을 하게 된다.

이를 위해 알고리즘의 [단계 2]는 다음과 같은 원리를 사용한다. 첫째, [단계 1]에서 도출한 임계경로 위주의 업무처리순서와 자원할당 결과를 입력받는다. 둘째, OR구조상의 다른 경로상의 활동들로 기존의 임계경로의 활동들을 대체한다. 이 때 새로운 활동의 처리순서는 임계경로의 것과 동일하게 유지한다. 셋째, 대체한 활동들에 대한 자원할당을 유전자 알고리즘을 통해 탐색한다. 이러한 원리를 바탕으로 나머지 OR구조의 자원할당에 대한 알고리즘은 다음과 같다.

〈임계경로 이외의 OR경로에 대한 알고리즘 [단계 2]〉

(1) [단계 1]의 업무처리순서 및 자원할당 해를 불러오기

(2) 대체 활동의 데이터 불러오기

임계경로 이외의 OR 경로에 속하는 활동들에 관한 데이터를 가져온다. 이때 임계경로를 포함한 전체 활동의 리스트(λ)와 대체 활동 이외의 활동에 관한 자원할당은 유지하고 대체된 활동들에 대한 할당 자원은 초기화 해둔다.

(3) 부분 문제 구성

대체 활동들 즉, 할당된 자원이 없는 활동들을 포함하는 부분 문제를 구성한다. 할당 자원이 없는 활동들은 대체 활동 표를 참조하여 λ'으로 둔다.

(4) 초기 모집단 생성

λ'의 리스트에 해당하는 각각의 활동을 수행할 수 있는 자원들을 임의로 선택해서 모집단의 크기만큼 개체(individual)를 생성한다(λ_{rsc}). 실행가능 자원은 $\alpha_{jr|\,j\in\lambda'}$의 값이 0이 아닌 자원들을 후보로 하여 임의 선택한다.

(5) 모집단의 적합도 평가, (6) 선택, (7) 교배, (8) 자손 개체의 적합도 평가, (9) 돌연변이, (10) 모집단의 적합도 평가, (11) 종료 조건과의 비교는 [단계 1]과 동일하다.

4. 알고리즘 성능분석

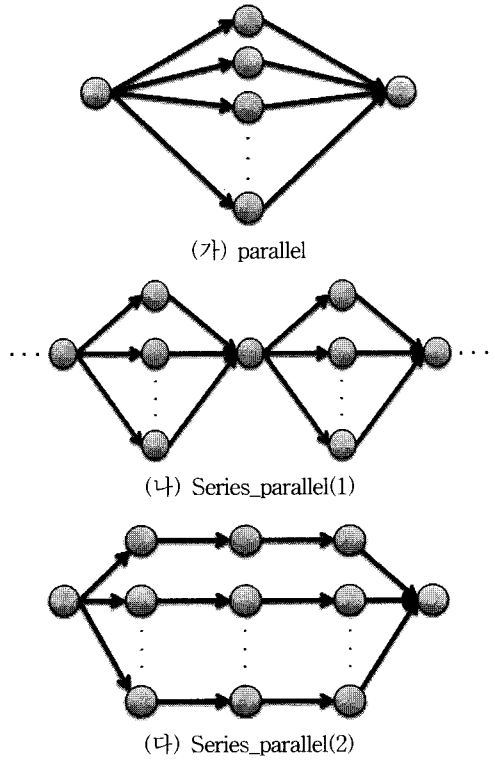
본 장에서는 제안한 알고리즘에 대한 성능 분석을 수행하였으며 실험에 사용된 토폴로지는 앞서 기술한 바와 같이 OR구조와 LOOP구조를 제거하고 AND와 순차구조만을 포함하는 새로운 토폴로지로 재구성된 것을 사용하였다. 실험은 <그림 4>와 같이 병렬성(parallelism)을 강조한 Parallel구조, 이러한 병렬구조가 반복되는 Series_parallel(1)구조, 병렬구조 내에 순차구조가 혼재하는 Series_parallel(2)구조의 3가지 토폴로지에 대해 진행하였고(단계 1), OR 분기에 해당하는 대체 활동에 대한 자원할당 실험(단계 2)을 Series_parallel(2)의 환경에서 진행하였다.

프로그램은 크게 3부분으로 구성되는데 구조 타입에 따라 문제를 생성하는 부분, [단계 1]의 해를 구하는 부분, 그리고 [단계 2]의 해를 구하는 부분으로 나누었다. [단계 1]과 [단계 2]는 각각 유전자 알고리즘, 실시간 자원할당 알고리즘, 랜덤 해를 구하는 프로그램으로 나누어 전체 7개의 개별 프

로그램으로 본 알고리즘의 성능을 분석하였다.

또한 성능은 활동 수(num_activity)가 60, 90, 120, 150개, 전체 자원수(num_resource)가 5, 10, 15인 경우에 진행되었다. 유전자 알고리즘의 성능을 평가하기 위해 모집단의 크기는 25, 50, 100 세 가지에 걸쳐 변화시켰으며 알고리즘의 종료 조건인 세대의 수(GEN)는 25, 50, 100 세 가지에 대해 실험을 진행하였다. 그리고 선택의 확률(p_select)을 0.5, 0.8, 돌연변이 확률(p_mutation)을 0.01, 0.05로 변화시키며 알고리즘의 성능을 비교하였다.

위의 각각의 조합에 대해 20개씩의 문제샘플을 생성, 총 77,760개의 샘플을 생성하였고 각각의 샘플에 대해 얻어진 유전자 알고리즘의 해(GA)를 기존의 윤상흠과 신용승[4]의 실시간 자원할당 알고리즘의 해(RT) 및 랜덤 해(RND)와 비교하였다. 랜덤해의 결과는 총 100번의 반복수행을 통해 얻은 완료시간들의 평균값을 적용하였다. 실험에 사용된 고유업무시간은[11, 50] 사이에서 임의로 선택하였으며, 업무수행도는[1.0, 1.5] 사이에서 임의로 선택하였다. 자원제약 상황은 각각의 활동에 대해 40% 이내의 자원만이 이용 가능하도록 설정하였다.



〈그림 4〉 선후관계 구조에 따른 프로세스 토폴로지의 종류

실험의 결과는 <표 3>에 요약되어 있다. <표 3>은 본 연구의 알고리즘과, 실시간 자원할당 알고리즘, 그리고 랜덤 해의 완료시간의 평균 및 실시간 자원할당 알고리즘과의 편차와 랜덤해의 평균과의 편차를 평균한 백분위 값을 나타낸다. 즉 유전자 알고리즘에 의한 완료시간을 V_{GA}, 실시간 자원할당 알고리즘의 완료시간을 V_{RT}, 그리고 랜덤해의 완료시간 평균을 V_{RND}라고 표시할 때, 실시간 자원할당 알고리즘과의 편차(Dev.(RT))는 $100\times(V_{RT}-V_{GA})/V_{RT}$, 랜덤해의 평균과의 편차(Dev.(RND))는 $100\times(V_{RND}-V_{GA})/V_{RND}$로 표현한다. <표 3>의 결과를 보면 유전자 알고리즘이 실시간 알고리즘과 랜덤 해보다 대체적으로 좋은 성능을 보임을 알 수 있다. 특히 Series_parallel(1)에서는 성능 차이가 더 크게 나타내고 있다.

다음으로 유전자 알고리즘의 여러 가지 모수의

〈표 3〉 성능 비교결과

프로세스 토폴로지	GA	RT	RND	Dev.(RT)	Dev.(RND)
Parallel	550.80	598.07	809.50	8.55	34.66
Series_parallel(1)	832.16	957.46	1320.86	12.88	37.46
Series_parallel(2)	551.77	584.00	803.51	6.13	33.73
전 체	644.91	713.18	977.96	9.18	35.28

변화에 따른 성능 변화를 살펴본다. 유전자 알고리즘의 성능에 영향을 미치는 모수에는 모집단의 크기, 세대의 수, 선택 확률, 그리고 돌연변이 확률 등이 있다. 이들 모수들을 개념적 유사성에 따라 두 가지 조합, 즉 (모집단의 크기, 세대의 수)와 (선택 확률, 돌연변이 확률)로 나누어 실험을 진행하였다. 각각의 경우의 실험결과가 <표 4>와 <표 5>에 요약되어 있다. <표 4>에서 보는 바와 같이 세대수와 모집단의 크기가 커질수록 유전자 알고리즘과 다른 해와의 성능차이가 다소 커졌으나 크게 변화되지는 않았다. 확률 모수의 경우에는 <표 5>에서 나타난 바와 같이 선택 확률과 돌연변이 확률의 값이 클수록 상대적 성능은 다소 향상됨을 알 수 있으며, 특히 선택 확률이 0.5, 돌연변이 확률이 0.05일 때 뛰어남을 확인할 수 있다.

다음은 [단계 2]에 대한 성능평가를 실시하였다. [단계 2]는 OR 구조를 가지는 토폴로지에 적용되므로 적절한 성능평가를 위해 Series_Parallel(2)에 대해 실험하였다.

실험은 임의의 난수를 발생시켜 그 수만큼 OR_split 노드가 Series_Parallel(2) 구조 내에 존재하도록 토폴로지를 구성하였다. 또한 고려해야 하는 OR구조 내의 경로상의 활동 수를 임계경로의 활동 수와 같거나 작도록 설정하였다. 실험의 결과는 <표 6>에 요약되어 있다. 기존 RT 알고리즘과의 비교에서 선택 가능한 전체 자원의 수가 적을수록 성능차이가 커진다는 것을 알 수 있었다. 또한 활동 수가 증가할수록 그 성능차가 커짐을 알 수 있었다. 자원의 수가 적어진다는 것은 자원 제약이 심하다는 것으로 해석될 수 있으며 위의 결과는 자원 제약이 증가할수록 본 연구에서 제시한 알고리즘의 효과가 크다는 것을 보여준다 할 수 있다. 반면 활동수가 작고 자원이 충분한 경우(예를 들어 60~15인 경우) 유전자 알고리즘과 실시간 자원할

〈표 4〉 모집단의 크기와 자손 수의 변화에 따른 실험결과

POPsize	GEN	CPU time	GA	RT	RND	Dev. (RT)	Dev. (RND)
25	25	6.89	647.79	713.75	978.57	8.85	35.05
	50	10.74	647.58	714.06	978.87	8.92	35.12
	100	18.41	644.33	711.34	975.12	9.07	35.17
50	25	10.59	645.57	713.31	978.00	9.11	35.24
	50	15.90	646.17	714.52	980.34	9.17	35.31
	100	26.40	643.19	711.89	976.55	9.28	35.34
100	25	19.14	643.96	713.44	977.12	9.33	35.29
	50	28.43	642.70	712.66	977.64	9.42	35.48
	100	46.98	642.88	713.65	979.39	9.51	35.54

〈표 5〉 선택과 돌연변이 확률의 변화에 따른 실험결과

p_select	p_mutation	GA	RT	RND	Dev.(RT)	Dev.(RND)
0.5	0.01	645.69	714.21	978.99	9.17	35.26
	0.05	644.34	713.23	977.95	9.29	35.33
0.8	0.01	645.22	712.67	976.98	9.09	35.21
	0.05	644.38	712.61	977.90	9.19	35.33

〈표 6〉 부 임계경로 이외의 경로에 대한 실험결과

활동수	자원수	CPU time	GA	RT	RND	Dev.(RT)	Dev.(RND)
60	5	2.70	398.89	489.59	633.94	18.25	36.21
	10	2.72	264.73	284.50	427.51	6.71	37.07
	15	2.72	219.80	218.52	354.01	-0.84	36.79
90	5	3.46	559.47	687.79	876.35	18.31	35.35
	10	3.50	349.57	380.48	562.62	7.88	36.93
	15	3.52	279.11	280.21	449.58	0.17	36.88
120	5	4.19	720.71	888.55	1112.56	18.54	34.46
	10	4.25	432.68	478.09	695.82	9.21	36.93
	15	4.27	337.10	343.08	545.68	1.50	37.29
150	5	4.97	880.01	1087.46	1338.57	18.67	33.58
	10	5.07	513.83	571.95	818.40	9.86	36.41
	15	5.09	392.73	404.91	636.55	2.74	37.45
평 균		3.87	445.72	509.59	704.30	6.13	9.25

당 알고리즘은 성능 면에서 큰 차이가 없다. 하지만 <표 6>은 임계경로를 제외한 기타 경로에 대한 결과로 앞에서 구한 임계경로에 대한 해를 함께 고려할 경우에는 좀 더 효과적인 결과를 얻을 수 있다. CPU time은 본 알고리즘의 컴퓨터상의 소요시간으로, 동일한 문제에 대해 알고리즘을 1,000회 반복했을 때의 소요시간이다. 1,000회의 반복에 걸리는 시간은 2~6초로 워크플로우 내 전체 활동 수가 증가할수록 계산 시간도 증가함을 알 수 있다.

5. 결론 및 향후 연구방향

본 연구는 자원제약 상황에서 워크플로우 업무처리를 위한 업무처리순서 결정과 자원할당이라는 두 가지 의사결정을 통합하여 해결할 수 있는 알고리즘을 개발하였다. 이를 위해 본 논문은 2단계 유전자 알고리즘을 제안하였으며 그 성능실험을 다양한 토폴로지와 모수에 대해 실시하였다.

제안된 알고리즘의 [단계 1]에서 업무처리순서는 유전자 알고리즘을 통해 결정하지만 각 활동에 대한 자원할당은 기존의 선행 연구에서 제안한 자원할당적합도 함수를 활용하였다. 반면 [단계 2]에서는 자원할당에서도 유전자 알고리즘을 통해 결정하였다. 따라서 [단계 1]에서의 자원할당 문제에서도 유전자 알고리즘을 적용한다면 더 뛰어난 해를 구할 수 있을 것이라 기대되지만 자원수가 커질 경우 전체 계산시간에 부담이 될 수 있다.

Alcaraz and Maroto[6]에 따르면 자원제약 프로젝트 스케줄링 문제(resource-constrained project scheduling problem, RCPSP)를 최적화 방법을 활용하여 최적 해를 구할 수 있는 한계 활동 수가 30개라고 한다. 그 이상의 활동에 관해서는 임계경로와의 편차를 통해 간접적으로 그 성능을 평가하고 있다. 마찬가지로 본 연구와 같은 자원제약 워크플로우 문제(resource-constrained workflow problem)의 해에 대한 절대적 성능을 평가할 수 있는 방법도 마련되어 있지 않다. 그러므로 자원제약 워크플로우 문제의 성능을 평가하기 위해서는 작은

사이즈의 문제에 대한 최적해를 도출할 수 있는 최적화 알고리즘의 개발이 새로운 연구과제가 될 수 있다.

참 고 문 헌

[1] 손진현, 김명호, "확장된 구조적 워크플로우 스키마에서 워크플로우 임계 경로의 결정", 「정보과학회논문지 : 데이타베이스」, 제29권, 제2호(2002), pp.138-147.

[2] 손진현, 오석균, 이윤준, 김명호, "고성능 분산 워크플로우를 위한 선형 계획법 기반의 워크플로우 작업 할당 방법", 「정보과학회논문지 : 데이타베이스」, 제27권, 제3호(2000), pp.549-557.

[3] 손진현, 장덕호, 김명호, "워크플로우 임계 경로에 관한 분석", 「정보과학회논문지 : 데이타베이스」, 제28권, 제4호(2001), pp.677-678.

[4] 윤상흠, 신용승, "워크플로우 완료시간 최소화를 위한 실시간 자원할당 알고리즘", 「산업경영시스템학회지」, 제29권, 제1호(2006), pp.1-8.

[5] 최경훈, 손진현, 김명호, "워크플로우 작업의 효율적인 배치를 위한 다단계 워크플로우 그래프 분할 기법", 「정보과학회논문지 : 데이타베이스」, 제30권, 제3호(2003), pp.310-319.

[6] Alcaraz, J., C. Maroto, "A Robust Genetic Algorithm for Resource Allocation in Project Scheduling," *Annals of Operations Research*, Vol.102(2001), pp.83-109.

[7] Hartmann, S., "A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling," *Naval Research Logistics*, Vol.45 (1998), pp.733-750.

[8] Hollingsworth, D., "The Workflow Reference Model," *Workflow Management Coalition*, Doc. Num. TC00-1003, Issue 1.1(1995), pp. 1-42.

[9] Plesums, C., "Introduction to Workflow, "*The Workflow Handbook 2003* edited by L. Fischer, Future Strategies Inc., Lighthouse, FL(2003), pp.19-38.

[10] Zhong, J., B. Song, "Verification of Resource Constraints for Concurrent Workflows," SYNASC'05(2005), pp.353-360.