經營科學
第23卷 第2號
2006年 11月

유전자 알고리즘을 이용한 매트릭스조직의 소프트웨어 개발 스케쥴링*

양미나** · †이건호**

Scheduling of Matrix Organization for Software Development using Genetic Algorithm*

Mi Na Yang** · Gun Ho Lee**

■ Abstract ■

Efficient scheduling for software development is a major concern for software engineers. Industries simultaneously try to perform a variety of projects with the limited resources on schedule. A way to overcome the limitation of resources is sharing of the resources through the projects. This study discusses the matrix organization for software development.

A scheduling for matrix organization is a special case of project management problem. The ultimate goal of scheduling problem in this study is to minimize the overall duration of the multiple projects. A genetic algorithm is presented to solve the scheculing problem of the matrix organization and is substantiated with numerical results.

Keyword : Matrix Organization, Genetic Algorithm, Project Scheduling

1. 서 론

동시대의 기업들은 기술개발, 경영혁신, 신제품 개발 등의 다양한 활동에 자원과 노력을 집중하고 있다. 이러한 활동들은 시작 시점과 완료 시점이 명확한 프로젝트의 형태이며 이러한 프로젝트의 일정관리는 기업의 핵심적인 성공요인이 되고 있다. 규모가 작은 프로젝트일 경우 관리자가 가지고

논문접수일 : 2005년 08월 1′일 논문게재확정일 : 2006년 10월 09일
* 본 연구는 숭실대학교 지원으로 수행되었음.
** 숭실대학교 산업정보시스템공학과
† 교신저자

있는 고유한 경험에 의해서 프로젝트의 계획이 어느 정도 가능하다. 그러나 프로젝트의 규모가 대형화되고 수행해야 할 프로젝트가 다양 할수록 관리자의 경험에만 의존한 일정계획 수립에는 한계가 있다.

소프트웨어를 개발하는 많은 기업들은 다수의 프로젝트를 동시에 수행하면서 주어진 자원의 제한으로 혹은 효율적인 활용을 위해 자원을 서로 공유하는 매트릭스 개발 조직을 가지고 있다. 이러한 매트릭스 조직은 각 프로젝트에 필요한 자원을 개발 단계별 기능의 팀에서 차출하여 작업을 수행하고, 작업을 완료한 자원을 다른 프로젝트에 투입할 수 있도록 자원 활용도를 높이는 조직구조이다.

프로젝트의 일정계획 및 통제를 위한 기존의 관리기법인 PERT(Program Evaluation and Review Technique)와 CPM(Critical Path Method)은 다양한 현장의 문제를 해결하는 데는 많은 제한이 있다. PERT나 CPM 등은 자원이 충분하다는 가정에서 문제를 풀기 때문에 비교적 간단한 절차로 문제를 해결할 수 있다. 또 자원에 대한 제약이 아주 작은 상황에는 보다 간단한 네트워크 모형만으로 문제를 해결하는데 충분하다. 그러나 자원의 공급에 제한이 있다면 일정을 결정하는데 어려움이 따르며, 이를 해결하는 데는 복잡한 절차가 필요하게 된다. 프로젝트의 기능을 수행하는 데 필요한 자원이 한정되어 있다면, 잠재적으로 병목현상을 일으킬 수 있는 작업이 발생할 수 있고, 효과적으로 일정계획을 수립하기가 어려울 것이다.

프로젝트 일정계획 문제는 전통적인 PERT/CPM 및 퍼지 기법을 응용한 연구[2], 퍼지 PERT를 사용하여 시스템 개발의 모호함을 극복하려는 시도[13], 소프트웨어 개발을 위한 퍼지 프로젝트 스케쥴링 시스템[10] 등 많은 연구가 있다. 복수의 프로젝트를 자원의 제약 아래 수행하기 위한 일정계획 방법에 대한 연구는 휴리스틱 알고리즘을 이용한 방법[1], Constraint Programming을 이용한 방법[5]이 있다. 인공지능 기법을 이용한 방법으로는 Tabu Search를 적용한 일정계획 연구[3] 및 제약 만족 기법(Constraint Satisfaction Technique)을 적용한 Job Shop 일정계획 방법[4]이 있다.

유전자 알고리즘을 이용한 연구로는 우선순위와 스케쥴링을 결합한 방법[19], 퍼지 최적 모델 방법[14], 카테고리 프로젝트 일정계획에 적용한 방법[18], 자원제약 하의 다중 프로젝트에 적용한 방법[12], 다중모드 자원에 적용한 방법[17], PERT 네트워크에 시간비용을 적용한 방법[9], 자원 레벨링(Resource Levelling)을 고려한 방법[15] 등의 연구가 있다.

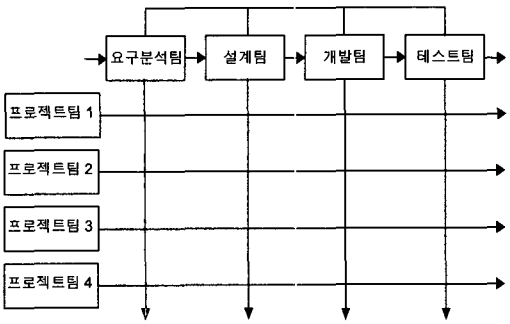
프로젝트 일정계획을 위한 연구는 아주 오랫동안 지속되고 있으나 매트릭스 조직을 위한 소프트웨어 개발 프로젝트 스케쥴링 문제에 대한 연구는 이루어지지 않았다. 따라서 본 연구에서는 유전자 알고리즘을 이용하여 매트릭스 조직에 적합한 프로젝트 스케쥴링을 방법을 제시하고자 한다. 소프트웨어 개발 스케쥴링 문제는 두 개 또는 그 이상의 소프트웨어 개발 프로젝트를 동시에 수행할 경우 각 프로젝트 네트워크에서 작업의 선행관계와 주어진 자원을 활용하여 각 프로젝트의 수행기간을 최소로 하는 것이다.

2. 매트릭스 조직의 스케쥴링

소프트웨어 life cycle 중 실질적인 개발과정은 사용자의 요구사항을 분석하여 도출하는 요구분석단계, 요구분석에 기초하여 소프트웨어가 어떻게 작동될 것인가를 설계하는 단계, 설계에 따라 코딩하는 개발단계와 요구되는 품질요소의 수준에 맞게 작동이 되는가를 시험하는 테스트 단계로 이어진다. 본 논문에 적용한 소프트웨어 개발 매트릭스 조직은 <그림 1>과 같다. 종축에는 기능적 팀으로 요구분석팀, 설계팀, 개발팀, 테스트팀이 있고, 횡축에는 각 프로젝트팀이 있다. 개발인력은 기능팀인 동시에 프로젝트팀에 속하여 한 프로젝트의 기능을 수행한 후 다른 프로젝트에 다시 투입될 수 있다. 특정 프로젝트 팀의 개발 인력은 소속팀 기능 및 다른 기능팀의 기능을 수행할 수

있다.

본 연구에서는 n개의 프로젝트가 존재하고 그 프로젝트 내에는 폭포수 모형의 개발과정 즉, 요구분석, 설계, 개발, 테스트의 작업을 포함한다.



〈그림 1〉 소프트웨어 개발 매트릭스 조직

2.1 개발팀의 구성

<표 1>은 각 인력의 소속팀과 지원 가능한 업무를 나타내고 있다. 개발인력 R1은 요구분석팀 소속으로 타 기능팀을 지원 하지 않으며 D1은 개발팀과 테스트팀의 지원이 가능하다는 뜻이다. 개발인력에 따라 복수의 지원이 업무가 있을 수 있는 반면 불가능한 작업도 있을 수 있으며 작업의 능률에서도 관련분야의 경험과 친숙도(familarity) 그리고 개인적인 능력(competence)에 따라서 소속팀 및 지원팀에서 각각 작업능률이 차이가 있다고 가정한다. 또한 동일 소속팀에서 동일한 작업에도 개발자에 따라 처리시간은 각각 차이가 있다고 가정한다.

프로젝트별 개발인력의 구성은 <표 2>와 같이 프로젝트의 각 기능에 해당하는 소속팀과 지원팀의 개발인력이 할당될 수 있다. 각 기능에 배치된 자원은 해당 기능팀의 자원과 다른 기능팀의 자원으로 구성된다. 프로젝트 1의 경우는 요구분석단계에서 소속팀원 3명이 수행하고 설계단계에서는 소속팀원 2명과 지원하는 1명이 수행하게 된다. 이와 같이 구현과 테스트 단계까지 7명의 소속팀원과 4명의 지원으로 수행하게 된다.

〈표 1〉 팀별 자원의 구성

개발인력	소속팀	지원 가능 기능			
		요구분석	설계	개발	테스트
R1	요구분석				
R2	요구분석		○	○	○
R3	요구분석		○		○
R4	요구분석		○	○	○
D1	설 계			○	○
D2	설 계			○	○
D3	설 계	○		○	○
D4	설 계	○		○	○
I1	개 발	○	○		○
I2	개 발				
I3	개 발	○			
I4	개 발	○			○
T1	테스트		○		
T2	테스트	○			
T3	테스트	○		○	
T4	테스트		○		

〈표 2〉 프로젝트의 자원 배치

프로젝트 번호	요구분석	설계	구현	테스트	개발인력 수
1	3	2(1)	1(2)	1(1)	7(4)
2	(2)	(3)	2(1)	1(1)	3(7)
3	2	1(2)	2(2)	1(2)	6(6)
4	1(1)	1(1)	2(1)	1(1)	5(4)
개발인력수	6(3)	4(7)	7(6)	4(5)	21(21)

주) (*) : 타 기능팀으로부터 지원받은 개발인력 수.

2.2 프로젝트별 기능의 구성

각 프로젝트의 기능은 <표 3>과 같이 요구분석, 설계, 코딩, 테스트를 의미하고 각 기능의 처리시간(processing time)은 기능을 처리하는데 소요되는 작업의 량(man-month)으로부터 얻을 수 있다. 본 연구에서는 프로젝트의 기능별 작업 예상 처리시간은(expected processing time)은 과거의 경험자료를 바탕으로 가중치를 고려하여 얻을 수 있는 것으로 가정한다. 각 자원별 관련분야의 경험과 친

숙도(familarity) 그리고 개인적 능력(competence)에 따라 작업능률 등을 <표 1>, <표 4>와 같이 고려하고 있다.

어떤 프로젝트에서 각 기능의 총 처리시간의 합은 그 프로젝트의 처리시간이 된다. 따라서 본 연구의 스케줄링은 각 기능에 개발인력을 적절히 배치하여 전체 프로젝트의 처리시간과 대기시간 및 유휴시간 등을 고려한 전체 프로젝트의 완료기간을 최소화하는 것을 목적으로 한다. 소프트웨어의 개발인력 간의 의사소통과 상호협조가 생산성에 크게 영향을 미치므로 프로젝트별 기능에 대한 개발인력의 상한선은 사전에 주어지는 것으로 가정한다.

〈표 3〉 프로젝트별 기능의 구성

프로젝트	기 능	처리시간
프로젝트 1	요구분석	53
	설 계	66
	개 발	68
	테스트	60
프로젝트 2	요구분석	88
	설 계	68
	개 발	68
	테스트	54
프로젝트 3	요구분석	76
	설 계	54
	개 발	64
	테스트	57
프로젝트 4	요구분석	63
	설 계	57
	개 발	56
	테스트	53

2.3 스케줄링 문제

매트릭스 조직의 소프트웨어 개발 스케줄링에서 궁극적으로 찾아야 할 해는 전체 프로젝트가 납기지연 없이 수행되고 개발기간을 최소로 하기 위하여 어느 프로젝트에 어떤 개발인력을 배치하는가이다.

전체 프로젝트 중 수행기간이 가장 긴 프로젝트의 개발기간, t_{cp}는 다음과 같다.

$$t_{cp} = \max\{t_i, i = 1, \cdots, n\}$$

t_i = 프로젝트 i의 수행시간

매트릭스 조직의 스케줄링에서 전체 프로젝트의 개발기간을 단축하기 위해서 t_{cp}에 해당하는 프로젝트의 수행기간을 어떻게 효과적으로 단축하느냐가 중요한 관심사항이다. 본 연구에서는 유전연산의 반복을 통해 t_{cp}의 프로젝트에 개발자원을 우선적으로 배치하여 각 프로젝트는 각 납기일(due date) 이내에 프로젝트가 완료하도록 프로젝트들 간의 종료시점의 차이를 최소화하여 전체 프로젝트의 개발기간을 단축하고자 한다.

각 기능을 수행할 수 있는 팀에는 개발인력을 포함하고 각 프로젝트에 적절히 배치하여 프로젝트를 수행한다. 소프트웨어 개발 특성 상 주요 자원은 개발인력이므로 본 연구에서는 개발인력 위주의 일정계획을 수립하고자 한다.

3. 유전자 알고리즘 설계

생성된 개체군은 하나의 염색체로 표현하고 적합도의 평가를 통해 가용한 해가 된다.

3.1 염색체(chromosome) 정의

프로젝트의 기능과 개발인력을 개체군으로 구성하여 염색체를 표현하면 <그림 2>와 같다. 염색체는 각 프로젝트와 프로젝트를 구성하고 있는 기능팀들, 즉 요구분석, 설계, 개발, 테스트의 팀들이 개발순서대로 위치하도록 한다. 또한 각 기능팀 소속 개발자원을 나타내고 있다 <그림 2> 프로젝트의 각 기능 팀에 타 기능 팀의 자원을 활용하여 전체 프로젝트의 자원 활용도를 높이도록 구성하였다. 각 기능에 배치된 개발인력은 해당 기능팀의 개발인력과 다른 기능팀의 개발인력을 포함한다. 예를 들어, 프로젝트 1의 설계기능 자원 중 설계팀 자원 D1과 D2외에 R4이 요구분석팀의 자원으로 설계업

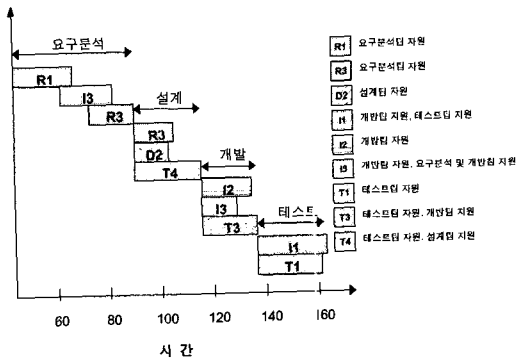
무를 지원하고, 개발기능 팀에서 개발팀 자원 I1외에 설계팀의 D3, D4 자원이 지원하고 있는 염색체를 나타내고 있다.

프로젝트	프로젝트 1				프로젝트 2				프로젝트 3			
기능	요구분석	설계	개발	테스트	요구분석	설계	개발	테스트	요구분석	설계	개발	테스트
자원	R1 R2 R3	D1 D2 R4	I1 D3 D4	T1 I4	I3 T2 T3	R2 R3 T4	I2 I3 T4	T2 D1	R1 R2 R4	R3 D2 T4	I1 R2 R4	D3 D4

〈그림 2〉 염색체 정의

3.2 초기해(initial population) 생성

유전자 정보를 바탕으로 프로젝트별 기능에 개발인력을 적절히 배치하여 가용한 해를 생성한다. 생성 방법은 기능을 수행할 수 있는 기능팀 개발인력과 해당 기능팀을 지원할 수 있는 개발인력을 무작위로 선택하여 개발인력의 처리시간 합이 기능의 처리시간을 만족하도록 하였다. 이미 다른 프로젝트에 배치된 개발인력은 작업 완료시간을 처리시간 계산 시 반영하게 된다. 작업시간이 계산된 자원을 프로젝트별 각 기능에 배치하고 프로젝트 기한일(due date)을 체크한다



〈그림 3〉 초기해 염색체의 Gantt차트 예

<그림 3>은 전체 프로젝트 Gantt차트의 일부로서, 각 기능팀의 자원들이 동시에 작업을 수행할 수 있다. 하지만 자원이 이미 다른 프로젝트 혹은 다른 기능의 작업에 할당되어 있다면 할당받은 작업을 완성한 후에 투입이 가능하다. 이런 경우 해당 작업은 대기상태에 있게 된다. <그림 3>의 자원 P3과 I3은 수행 작업을 종료한 수에 다른 작업에 투입되고 있음을 나타내고 있다.

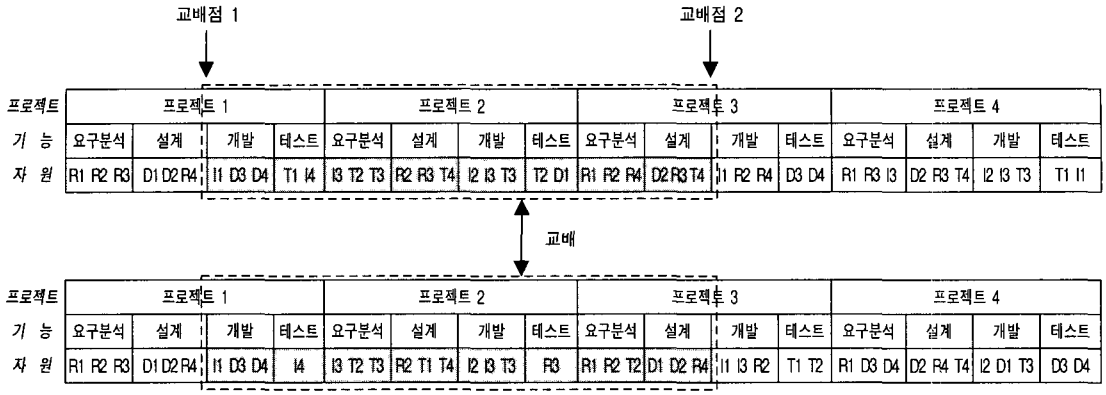
3.3 교배(crossover) 연산

본 논문에서는 두 개의 교배점을 선택하여 교배점 사이 프로젝트의 자원을 교환하는 부분일치 교배(partially matched crossover) 방식을 적용하였다. 균등분포를 이루는 [1, $4n$-1]의 범위에서 교배점을 무작위로 선택하였다(n=프로젝트 수). 앞의 교배점과 뒤의 교배점 사이가 교배구간으로 실제로 교배가 이루어지는 부분이다. 교배 전 염색체는 <그림 4>와 같다.

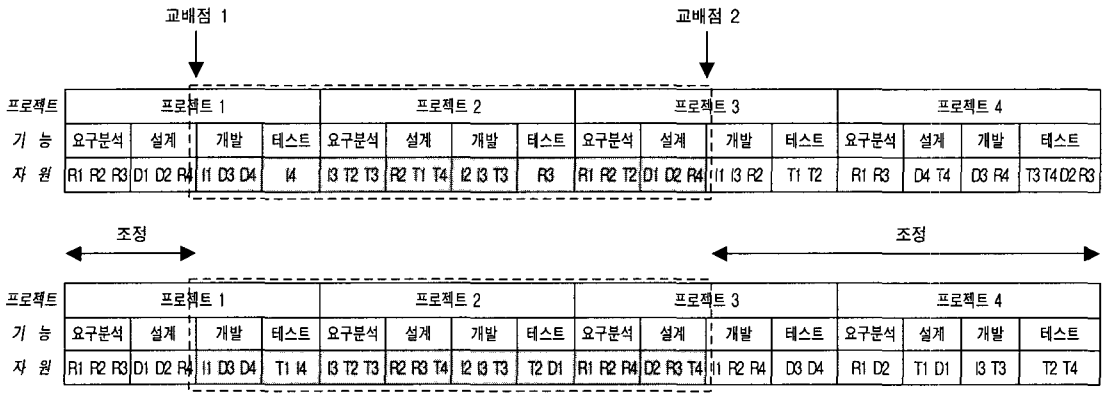
두 개체군에서 프로젝트 개발인력의 일부분을 프로젝트의 기능을 기준으로 교배점을 생성하여 개체군을 교환한다. 이때 교배 후에 각 프로젝트의 개발인력이 중복 배치 될 수 있는데, 교배구간의 개발인력과 교배구간 밖의 개발인력의 중복은 전체 프로젝트의 처리시간이 증가하거나 일부 개발인력의 집중 배치로 유휴자원이 많아지고 적절한 인력 활용이 어려워진다. 따라서 교배 후에는 교배구간 밖 프로젝트의 인력을 교배구간 내 프로젝트의 인력과 중복되지 않도록 중복여부를 체크하고, 개발인력의 중복을 재조정하는 과정이 필요하다. 교배연산 과정을 표현하면 다음과 같다.

BEGIN
  프로젝트 교배점을 무작위로 선택
  프로젝트 내의 기능 교배점을 무작위 선택
  교배구간 내의 개체군 교환(swapping)
  교배구간 내의 개체군 처리시간 계산
  교배구간 밖의 개체군 조정
END

교배 후 염색체를 표현하면 <그림 5>와 같다. 교배구간 밖의 개발인력을 교배구간 내의 개발인력과 중복을 피하여 조정 배치하게 된다. 상위 개체군의 교배구간 밖 프로젝트 3을 예로 들면 개발



〈그림 4〉 교배 전 개체군



〈그림 5〉 교배 후 개체군

기능의 R4 인력은 교배 후에 동일 프로젝트의 요구분석 기능에 배치되어 인력이 집중될 가능성이 있다. 따라서 R4 인력을 I3 인력으로 조정 배치하였다. 교배구간 밖의 다른 프로젝트와 기능들도 마찬가지 방법으로 중복을 피하여 조정하는 과정을 거치게 된다.

3.4 돌연변이(mutation)

본 연구의 유전자 알고리즘 돌연변이는 매트릭스 조직 스케줄링의 특성에 따라 처리시간이 짧은 프로젝트의 개발인력을 처리시간이 가장 긴 CP (Critical Path) 프로젝트의 기능을 지원하도록 하여 전체 프로젝트의 처리시간을 단축시키는 방법을 적용한다. CP 프로젝트의 기능 중 처리시간이 다른 프로젝트들의 평균 처리시간 보다 큰 기능을 찾아내어 빠른 프로젝트에서 추출한 자원을 배치하였다. 돌연변이 후에 CP 프로젝트를 재검색 하여 다시 돌연변이를 적용하고, 만일 CP 프로젝트의 처리시간이 돌연변이 전 CP 프로젝트의 처리시간 보다 길어지면 돌연변이 연산을 중지하였다. 이러한 연산 과정을 의사코드로 표현하기 위하여 다음을 정의 한다.

t_b : 돌연변이 전 CP 프로젝트 종료시간
t_a : 돌연변이 후 CP 프로젝트 종료시간
$f \leftarrow \min\{t_i ; i=1,\cdots,n\}$
여기서 n는 프로젝트 수
r : 개발인력
p_{ij} : 프로젝트 i의 기능 j의 처리시간

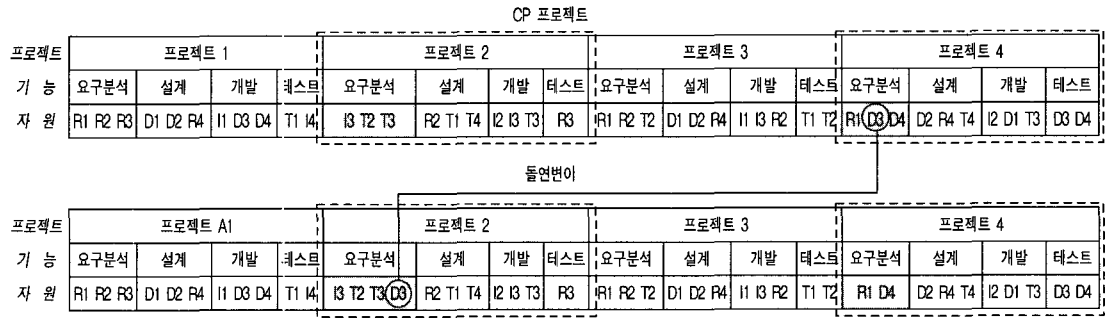
BEGIN
  $t_b \leftarrow 0$
  $t_a \leftarrow 0$
  COMPUTE t_i, for $i=1,\cdots,n$
  COMPUTE p_{ij} for $i=1,\cdots,n$; $j=1,\cdots,m$
  DO
    FIND CP 에 해당하는 프로젝트, $k \leftarrow$ CP
    FIND f
    $t_b \leftarrow t_k$
    $t_a \leftarrow \min\{p_{kj}\,;\, j=1,\cdots,m\}$
    $r \leftarrow f$에서 f의 참여율 낮은 자원
    DELETE r (f에서 r을 제거)
    INSERT r (체에 r 삽입)
    COMPUTE t_i, for $i=1,\cdots,n$
    $t_a \leftarrow t_k$
  WHILE ($t_b < t_a$)
END

돌연변이 연산의 전 후를 비교하여 표현하면 <그림 6>과 같다. 빠른 처리시간 프로젝트의 기능 인력을 추출하여 CP 프로젝트의 기능에 추가하였다. 연산 후에는 CP 프로젝트가 바뀔 수 있으므로 처리시간을 재계산하여 새로운 CP 프로젝트를 찾아 연산을 반복하게 된다.

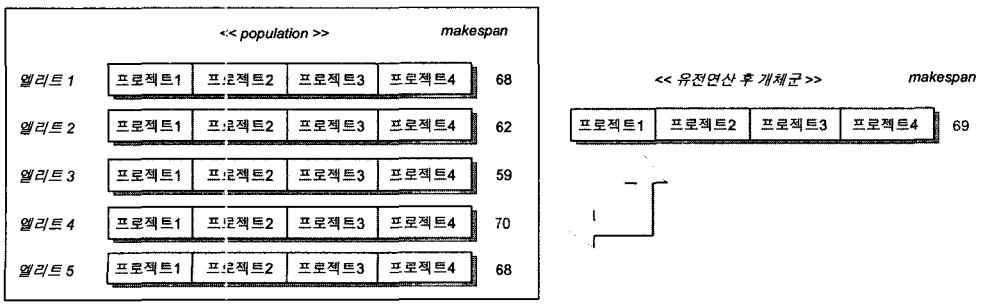
3.5 적합도 평가(fitness check)

유전연산 적용 후 생성되는 개체군은 프로젝트 납기일을 준수하여야 한다. 유전연산 후 개체군의 처리시간을 비교하여 처리시간이 짧을수록 적합도가 높은 것으로 판단하고, 적합도가 높은 엘리트 개체군을 보존하여 유전연산 수행시 생성되는 개체군과 비교하였다. 생성된 개체군의 적합도가 엘리트 집단의 적합도 보다 높은 경우 엘리트 집단의 개체군과 교환하여 최적의 엘리트 개체군을 보존하였다.

엘리트 집단의 열성 엘리트와 새로운 개체군과의 교환을 표현하면 <그림 7>과 같다. 개체군의 처리시간을 비교하여 교환한다.



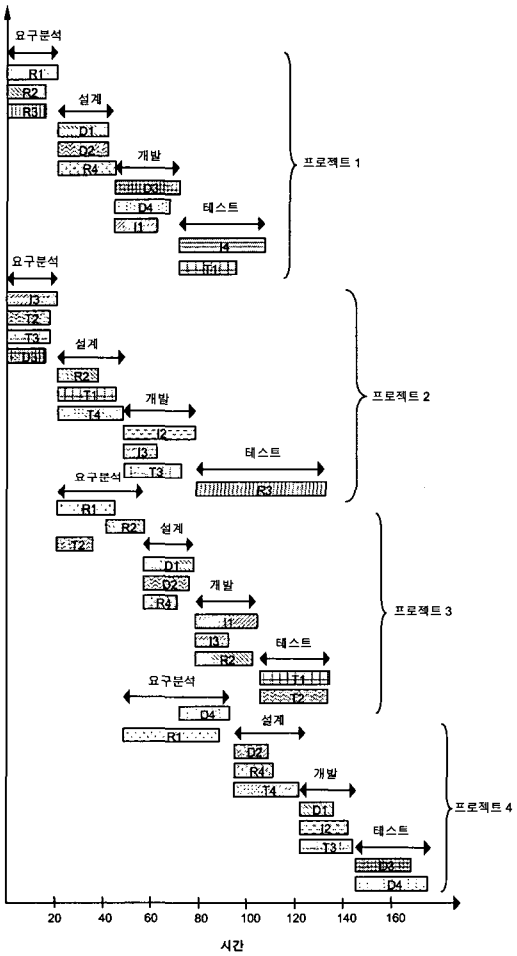
〈그림 6〉 돌연변이 연산



〈그림 7〉 적합도 평가 후 엘리트 보존

유전연산 후 개체군의 처리시간이 69로 기존 개체군의 엘리트 4의 처리시간 70 보다 우수하다. 따라서 population에서 엘리트 4는 제거하고 처리시간 69에 해당되는 개체군을 추가하게 된다.

유전연산의 종료규칙(termination rule)은 교배연산과 돌연변이 연산을 설정된 반복수 내에서 수행하며 해를 찾도록 하였다.



〈그림 8〉 각 프로젝트별 스케줄링의 예

각 프로젝트별 임의의 스케줄링의 결과는 <그림 8>에서와 같이 간트차트로 나타낼 수 있다. 각 개별인력의 지원가능 작업이 제한되면 스케줄링의 문제는 보다 단순화되어 쉽게 해를 찾을 수 있겠지만, 개발인력의 활동도가 떨어져 유휴시간(idle time)도 많아지게 된다. 반면 지원가능 작업이 많으면 많을수록 스케줄링의 문제는 어렵겠지만 유휴시간 없이 타 프로젝트의 다른 기능팀에 즉시 편성되어 개발업무를 담당 할 수 있게 된다. 따라서 개발인력의 다 기능화는 개발의 생산성 및 인력의 활용도에 기여 할 수 있을 것이다.

4. 분석과 평가

알고리즘의 구현과 테스트는 Microsoft Windows 2000의 O/S와 Intel Celeron CPU 2.4 GHz/256M RAM을 사용하고 Java(V. jdk 1.4) 언어를 이용하였다. 유전알고리즘을 검증하기 위하여 30 가지의 다양한 문제유형을 컴퓨터 프로그램을 이용하여 생성하였다. 프로젝트의 크기는 100개 이내로 제한하고, 각 프로젝트별로 개발인력의 구성은 프로젝트 수에 비례하게 하였다. 프로젝트의 기능별 요구되는 작업시간은 균등분포의 구간 [50, 70]에서 무작위 추출하여 생성하였고, 또한 개발자별 혹은 개발자의 업무별 작업능률의 비율을 제시하기 위하여 소속팀 개발인력의 처리시간은 균등분포에 의한 구간 [5, 7]에서 생성하였다. 또한 지원팀 개발인력의 처리시간은 소속팀의 개발인력 보다 능률이 떨어지는 것으로 가정하고, 균등분포 구간 [6, 10]에서 생성하였다. 팀별 개발인력의 생성 데이터를 보면 <표 4>와 같다. 팀별 개발인력의 수는 해당 프로젝트의 수에서 기능팀 수의 배수 내의 균등분포에서 무작위 생성하였다. 문제유형별 프로젝트 수 및 팀별 인원 구성을 보면 <표 5>와 같다. 엘리트 개체군의 해는 best solution과 개체군과의 편차의 산술평균으로 평가하였다.

$$\text{편차} = \sqrt{(best\ solution - \text{개체군처리시간})^2}$$

$$best\ solution = \min_{i \in p}\{t_i, i = 1, 2, \cdots n\}$$

t_i : 엘리트 i의 처리시간

p : 유전연산 종료 후 개체군의 엘리트 집합

n : 개체군의 크기(population size)

〈표 4〉 개발인력의 주업무와 지원업무의 능률

개발인력	소속팀		프로젝트팀1		프로젝트팀2		프로젝트팀3	
	소속팀	시간	지원팀	시간	지원팀	시간	지원팀	시간
R1	요구분석	6	설계	8	개발	7	테스트	
R2	요구분석	7						
R3	요구분석	6	개발	7				
R4	요구분석	5	개발	6	테스트	6		
R5	요구분석	5	설계	6	테스트	6		
R6	요구분석	6						
R7	요구분석	6	개발	7	설계	7		
D1	설계	7						
D2	설계	4	테스트	8	요구분석	6		
D3	설계	6						
D4	설계	7	테스트	8	요구분석	8		
D5	설계	5						
D6	설계	7	요구분석	9	테스트	7	개발	6
D7	설계	5	테스트	8	요구분석	8		
D8	설계	5	개발	6	요구분석	7		
I1	개발	6	요구분석	7	테스트	7	설계	8
I2	개발	6	설계	7				
I3	개발	7	설계	8	요구분석	10	테스트	9
I4	개발	6	테스트	7	요구분석	8		
I5	개발	7	테스트	6				
T1	테스트	5	개발	6	설계	6	요구분석	6
T2	테스트	6	요구분석	7	설계	8		
T3	테스트	5	요구분석	6	설계	6		
T4	테스트	6	설계	7	요구분석	10	개발	7
T5	테스트	5	설계	7	요구분석	7		
T6	테스트	6	개발	7	설계	9		
T7	테스트	5	설계	6				
T8	테스트	6	설계	8				

〈표 5〉 문제유형 최우수와의 편차

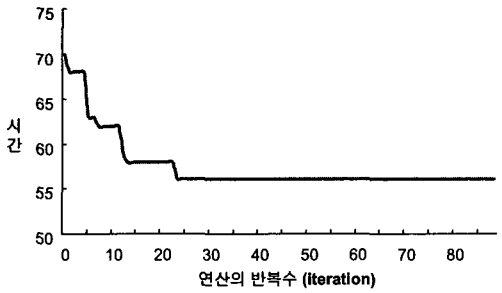
문제유형	프로젝트수	팀별 자원수				자원수	최우수해	평균편차
		요구분석	설계	개발	테스트			
1	8	10	13	8	10	41	153	16.37
2	15	21	24	17	17	79	115	16.67
3	57	65	84	91	57	297	130	16.49
4	30	50	45	50	43	188	98	19.65
5	41	65	70	61	52	248	101	19.57
6	82	157	85	99	100	441	99	16.40
7	94	164	163	148	164	639	108	19.24
8	20	26	35	22	38	121	95	17.92
9	94	153	138	119	180	590	113	16.52
10	9	11	13	16	14	54	66	31.09
11	44	85	84	64	65	298	100	22.07
12	14	23	21	20	17	81	93	20.17
13	23	32	35	29	37	133	120	15.29
14	25	29	26	25	45	125	106	27.51
15	57	113	80	96	38	357	114	24.08
16	35	64	48	60	37	209	92	23.32
17	83	125	149	146	89	509	103	23.35
18	52	84	77	81	83	324	112	26.17
19	28	34	55	39	43	171	91	29.86
20	95	166	139	129	157	591	111	21.63
21	14	21	23	14	18	76	96	25.88
22	92	94	147	93	144	478	132	19.49
23	94	125	144	183	183	635	103	21.45
24	9	12	16	9	16	53	75	20.49
25	38	63	43	49	66	220	105	18.92
26	91	105	156	168	158	587	93	18.49
27	18	26	23	18	35	102	104	29.64
28	68	120	85	84	127	416	108	23.47
29	88	127	148	143	90	508	116	17.94
30	24	32	47	33	42	154	87	28.77

4.1 연구 결과분석

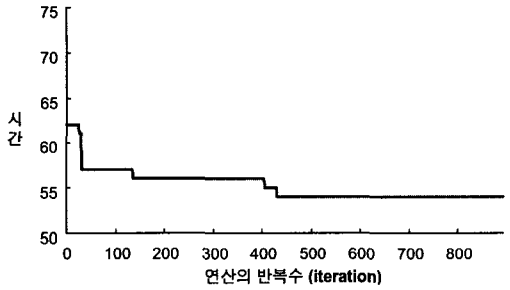
문제유형을 선택하여 유전연산의 반복수를 조절해 가며 연산 결과를 분석하였다. <그림 9>와 같이 반복수가 증가 할수록 처리시간이 단축되고 있다. 본 연구에서는 반복수가 어느 시점에 이르면 더 이상의 시간단축은 일어나지 않았고, 알고리즘 종결 이전의 최소 처리시간의 해를 가장 우수한 해(best solution)로 하였다.

같은 문제유형에서 유전연산의 반복수를 증가시켜 1000회를 수행한 결과 엘리트 개체군은 유전연산의 반복수에 비례하여 생성됨을 확인할 수 있었다. 이것을 그래프로 표현하면 <그림 9>와 같다. 유전연산의 반복수에 따른 최소 수행시간은 큰 차이

를 보이지 않았지만, 반복회수가 많은 경우 우수한 형질의 엘리트 개체군이 더 넓게 분포하므로 좋은 해를 선택할 확률이 높아질 수 있음을 알 수 있다.



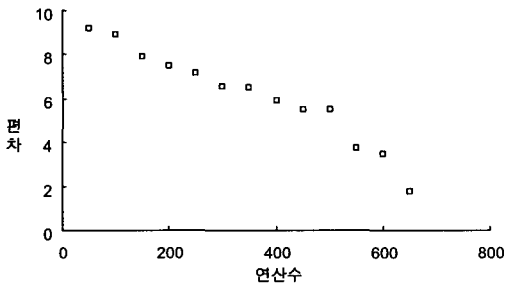
〈그림 9〉 유전연산 100회 반복 후 처리시간



〈그림 10〉 유전연산 1000회 반복 후 처리시간

유전연산의 반복수를 달리하여 생성되는 개체군과 최적해와의 편차를 <표 6>에서 비교하였다. 유전연산을 많이 반복 할수록 최적해와 가용한 개체군과의 편차가 줄어드는 것을 확인할 수 있다.

유전연산 반복수에 따른 최적해와 개체군과의 편차는 <그림 11>과 같이 감소함을 알 수 있다.

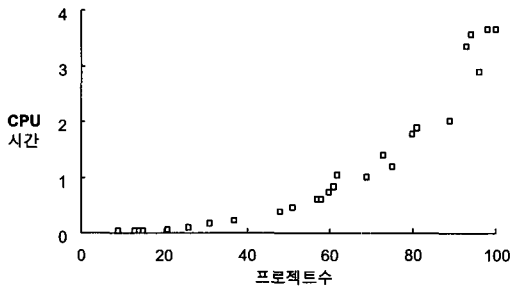


〈그림 11〉 최소 makespan과 개체군 편차

〈표 6〉 연산 반복에 따른 최우수 해와의 편차

유전연산 반복수	최우수해 (best solution)	객체군 평균	편 차
50	87	94	9.17
100	87	100	8.89
150	87	95	7.87
200	87	94	7.48
250	87	95	7.14
300	87	94	6.56
350	87	94	6.48
400	86	93	5.92
450	87	92	5.48
500	88	95	5.29
550	87	92	3.74
600	87	90	3.46
650	87	88	1.73

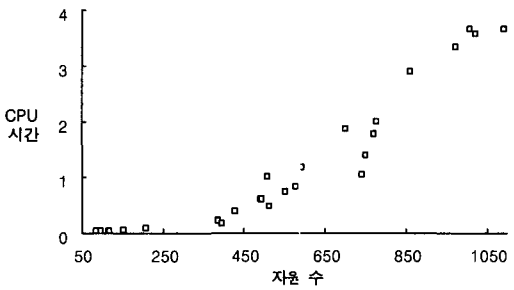
문제 유형별 프로젝트 수에 따른 유전연산 수행시간을 <그림 12>에 나타내고 있다. 프로젝트수가 많을 수 록 최우수 해를 찾는 유전연산의 수행시간이 증가하고, 프로젝트 수가 많아 문제가 복잡해지면 유전연산의 수행시간이 급격하게 늘어남을 나타내고 있다.



〈그림 12〉 프로젝트 수와 유전연산 수행시간

문제 유형별 개발인력의 수에 따른 유전연산 수행시간을 <그림 13>에서 나타내고 있다. 프로젝트의 크기가 커서 개발인력이 많아 질 수록 best solution을 찾는 유전연산의 수행시간이 증가하고 있다. 또한 프로젝트 수가 많아 문제가 복잡해지고, 개발인력의 투여수가 많아지면 유전연산의 수행시

간이 급격하게 늘어남을 확인할 수 있다.



〈그림 13〉 자원 수와 수행시간

5. 결 론

최근 소프트웨어 개발 환경은 급속도로 변화하고 있다. 프로젝트의 문제 영역도 점점 복잡해져가고 규모도 확대되는 추세이다. 따라서 개발인력을 적절히 배치하여 프로젝트의 납기일을 준수하며 좋은 품질의 소프트웨어를 개발하는 일은 중요하면서도 쉽지 않은 일이 되었다.

유전자 알고리즘의 연산을 반복 할수록 총 처리시간이 짧은 엘리트(elite) 개체군이 생성되어 우수한 형질의 개체군의 선택 확률이 높아짐을 알 수 있었다. 따라서 프로젝트의 스케줄링 시 유휴 개발인력을 최소화하여 인력의 적절한 배치가 용이해지고, 이를 통해 프로젝트의 처리시간을 단축하는 일정계획이 가능해졌다. 또한 유전연산의 반복수를 증가시킬수록 최우수 해(best solution)와 가용한 개체군과의 편차가 감소하여 좋은 해의 분포가 넓어짐을 확인하였다.

매트릭스 조직에서 스케줄링의 결과로 얻는 해의 질은 우수한 알고리즘을 이용하는 것도 중요하지만, 개발자원의 다 기능화를 통한 업무의 할당의 융통성이 부여 될 때 유휴시간 및 대기시간을 최소화하여 보다 더 우수한 해를 도출 할 수 있을 것이고 이는 곧 전사적인 생산성을 용이하게 향상 시킬 수 있는 방안이 될 수 있음을 보여주고 있다.

본 논문은 전통적인 폭포수 모형에 알고리즘을 적용한 연구로서 프로젝트의 개발 프로세스가 반복되는 객체지향 개발 환경에 적용하기에는 한계가 존재한다. 객체지향 개발을 위한 매트릭스 조직의 스케줄링을 위해서는 프로세스의 반복 처리 방법에 대한 연구가 있어야 하고, 개발인력을 프로젝트의 반복 단계별로 배치하는 방법에 대한 점진적인 연구가 필요할 것이다.

참 고 문 헌

[1] 김정자, 공명달, "자원제약하의 복수 프로젝트 일정계획을 위한 휴리스틱 알고리즘", 「대한산업공학회지」, 제13권, 제1호(1987), pp.110-119.

[2] 여한구, 이종태, "PERT/CPM에서의 프로젝트 완료시간 예측과 주경로 파악 및 통제를 위한 퍼지 기법의 응용", 「산업기술논문집」, 제13권(1999), pp.149-160.

[3] 윤종준, 이화기, "자원제약하의 동적 다중 프로젝트 일정계획에 Tabu Search 적용", 「산업경영시스템학회지」, 제22권, 제52호(1999), pp.297-309.

[4] 윤종준, 이한기, "Tabu Search와 Constraint Satisfaction Technique를 이용한 Job Shop 일정계획", 「산업경영시스템학회지」, 제25권, 제71호(2002), pp.92-101.

[5] 이화기, 정제원, "Constraint Programming 을 이용한 자원제약 동적 다중프로젝트 일정계획", 「산업공학」, 제12권, 제3호(1999), pp.362-373.

[6] 장여일, 서정윤, "프로젝트 관리도구로서의 CCPM의 실효성에 관한 연구", 「인제농촌」, 제14권, 제2호(1998).

[7] 전영준, 김동연, 김진일, "유전자 알고리즘을 이용한 효과적인 퍼지 분류 방법에 관한 연구", 「동의대학교 산업기술연구지」, 제15권(2001), pp.213-219.

[8] 최원준, "CC(Critical Chain) Project 관리", 울산대학교 산업공학과, 2002.

[9] Azaron, A., C. Perkgoz, and M. Sakawa, "A genetic algorithm approch for the time-cost trade-off in PERT networks," *Applied Mathematics and Computation*, 2004.

[10] Hapke, M., A. Jaszkiewicz, and R. Slowinski, "Fuzzy project scheduling system for software development," *Fuzzy Sets and Systems*, Vol.67(1994), pp.101-117.

[11] Holland, J.H., "Adaptation in Natural and Artificial Systems," *University of Michigan Press*, (Second edition : MIT press, 1992), 1975.

[12] Kim, K.W., Y.S. Yun, J.M. Yoon, M. Gen, and G. Yamazaki, "Hybrid genetic algorithm whth adaptive abilities for resource-constrained multiple project scheduling," *Computers in Industry*, Vol.56(2005), pp. 143-160.

[13] Kim, S.-R., "Fuzzy PERT Applications for System Development Scheduling," 「북악정보기술논집」, Vol.9(2003), pp.1-14.

[14] Leu, S.-S., A.-T. Chen, and C.-H. Yang, "A GA-based fuzzy optimal model for construction time-cost trade-off," *International Journal of Project Management*, Vol.19 (2001), pp.47-58.

[15] Liu, S.-X., "Genetic Algorithm for Resource Levelling Problem in Multi-mode Project Scheduling," *Control and Decision*, Vol.16, No.1(2001).

[16] Mintzberg, H., *Power of In and Around Organizations*, Englewood Cliffs, N.J., Prentice-Hall Inc., 1983.

[17] Mori, M. and C.C. Tseng, "A genetic algorithm for muti-mode resource constrained project scheduling problem," *European Journal of Operational Resrarch*, Vol.100(1997), pp.134-141.

[18] Ozdamar, L., "A Genetic Algorithm Approach to a General Category Project Scheduling Problem," *IEEE Trans. on Syst., Man, and Cybern.*, Vol.29(1999).

[19] Qiu, M., "Prioritising and scheduling road projects by genetic algorithm", *Mathematics and Computers in Simulation*, Vol. 43(1997), pp.569-574.