매트릭스조직의 소프트웨어 개발 스케쥴링
Software Development Scheduling for Matrix Organization

양미나*, 이건호**
숭실대학교 산업정보시스템공학과, *minaminay@yahoo.co.kr, **ghlee@ssu.ac.kr

Abstract

Efficient scheduling for software development is a major concern for software engineers. A industry simultaneously performs a variety of projects with the limited resources without overdue. 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 reduce the overall duration of the multiple projects. A genetic algorithm is presented to solve the scheduling problem of the matrix organization and is substantiated with numerical results.

1. 서론

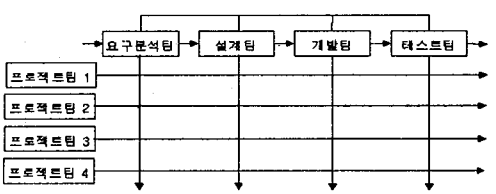
소프트웨어를 개발하는 많은 기업들은 다수의 프로젝트를 동시에 수행하면서 주어진 자원의 제한으로 혹은 효율적인 활용을 위해 자원을 서로 공유하는 매트릭스 개발 조직을 가지고 있다.

프로젝트 일정계획 문제는 전통적인 PERT/CPM 및 퍼지기법을 응용한 연구[1], 퍼지 PERT를 사용하여 시스템 개발의 모호함을 극복하려는 시도[2], 소프트웨어 개발을 위한 퍼지 프로젝트 스케쥴링 시스템[3] 등 많은 연구가 있었다. 복수의 프로젝트를 자원의 제약 아래 수행하기 위한 일정계획 방법에 대한 연구는 휴리스틱 알고리즘을 이용한 방법[4], Constraint Programming을 이용한 방법[5] 등이 있다. 인공지능 기법을 이용한 방법으로는 Tabu Search를 적용한 일정계획 연구[6] 및 제약 만족 기법(Constraint Satisfaction Technique)을 적용한 Job Shop 일정계획 방법[7]이 있다. 유전자 알고리즘을 이용한 연구로는 우선순위와 스케쥴링을 결합한 방법[8], 퍼지 최적 모델 방법[9], 카테고리 프로젝트 일정계획에 적용한 방법[10], 자원제약 하의 다중 프로젝트에 적용한 방법[11], 다중모드 자원에 적용한 방법[12], PERT 네트워크에 time-cost를 적용한 방법[13], Resource Levelling을 고려한 방법[14] 등의 연구가 있다.

본 연구에서는 유전자 알고리즘을 이용하여 매트릭스 조직에 적합한 프로젝트 스케쥴링 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

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



[그림 1]. 소프트웨어 개발 매트릭스 조직

- 개발팀의 구성

<표1>은 각 인력의 소속팀과 지원 가능한 업무를 나타내고 있다. 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.

<표1> 팀별 자원의 구성

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

프로젝트별 개발인력의 구성은 <표2>와 같이 프로젝트의 각 기능에 해당하는 소속팀과 지원팀의 개발인력이 할당될 수 있다. 각 기능에 배치된 자원은 해당 기능팀의 자원과 다른 기능팀의 자원으로 구성된다.

<표2> 프로젝트의 자원 배치

프로젝트 번호	요구분석	설계	구현	테스트	개발인력수
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)

(*) : 타 기능팀으로부터 지원받은 개발인력 수

- 프로젝트별 기능의 구성

각 프로젝트의 기능은 <표3>와 같이 요구분석, 설계, 코딩, 테스트를 의미하고 각 기능의 처리시간(processing time)은 기능을 처리하는데 소요되는 작업의 량(Man-month)을 의미한다.

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.

<표3> 프로젝트별 기능의 구성

프로젝트	기 능	처리시간
프로젝트 1	요구분석	65
	설 계	68
	개 발	70
	테스트	60
프로젝트 2	요구분석	54
	설 계	52
	개 발	69
	테스트	52
프로젝트 3	요구분석	67
	설 계	61
	개 발	57
	테스트	56
프로젝트 4	요구분석	54
	설 계	60
	개 발	52
	테스트	55

- 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.

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

t_i = 프로젝트 i 의 수행시간

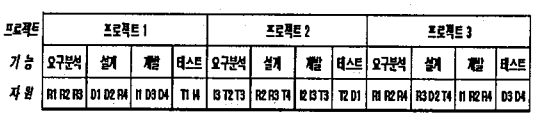
전체 프로젝트의 개발기간을 단축하기 위해서 t_{cp} 에 해당하는 프로젝트의 수행기간을 어떻게 효과적으로 단축하느냐가 중요한 관심사항이다. 본 연구에서는 유전연산의 반복을 통해 t_{cp} 의 프로젝트에 개발자원을 우선적으로 배치하여 각 프로젝트는 각 납기일(due date) 이내에 프로젝트가 완료하도록 프로젝트들 간의 종료시점의 차이를 최소화하여 전체 프로젝트의 개발기간을 단축하고자 한다.

4. 유전자 알고리즘 설계

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

- 염색체(chromosome) 정의

프로젝트의 기능과 개발인력을 개체군으로 구성하여 염색체를 표현하면 [그림 2]와 같다. 각 프로젝트는 요구분석, 설계, 개발, 테스트의 과정이 순서대로 위치하고, 각 기능에는 개발인력이 배치되어 구성된다. 각 기능에 배치된 개발인력은 해당 기능팀의 개발인력과 다른 기능팀의 개발인력을 포함한다.



[그림 2]. 염색체 정의

- 초기해(initial population) 생성

유전자정보를 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 계산시 반영하게 된다. 작업시간이 계산된 자원을 프로젝트별 각 기능에 배치하고 프로젝트 기한일(due date)을 체크한다.

- 교배(crossover) 연산

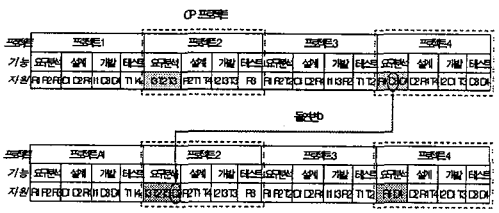
본 논문에서는 두 개의 교점을 선택하여 교배점 사이 프로젝트의 자원을 교환하는 부분일치 교배(partially matched crossover) 방식을 적용하였다. 균등분포를 이루는 [1, 4n-1]의 범위에서 교배점을 무작위로 선택하였다(n=프로젝트 수). 앞의 교배점과 뒤의 교배점 사이가 교배구간으로 실제로 교배가 이루어지는 부분이다.

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.

- 돌연변이(mutation)

본 연구의 유전자 알고리즘 돌연변이는 처리시간이 짧은 프로젝트의 개발인력을 처리시간이 가장 긴 CP(Critical Path) 프로젝트의 기능을 지원하도록 하여 전체 프로젝트의 처리시간을 단축시키는 방법을 적용한다. 돌연변이 후에 CP 프로젝트를 재검색 하여 다시 돌연변이를 적용하고, 만일 CP 프로젝트의 처리시간이 돌연변이 전 CP 프로젝트의 처리시간 보다 길어지면 돌연변이 연산을 중지하였다. 연산 과정을 의사코드로 표현하기 위하여 다음을 정의 한다.

돌연변이 연산의 전 후를 비교하여 표현하면 [그림 3]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.



[그림 3]. 돌연변이 연산

- 적합도 평가(fitness check)

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.

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

6. 분석과 평가

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 수는 해당 프로젝트의 수에서 기능팀의 수의 배수 내의 균등분포에서 무작위 생성하였다. 문제유형별 프로젝트 수 및 팀별 인원구성을 보면 <표 5>와 같다. 엘리트 개체군의 해는 best solution 과 개체군과의 편차의 산술평균으로 평가 하였다.

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

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

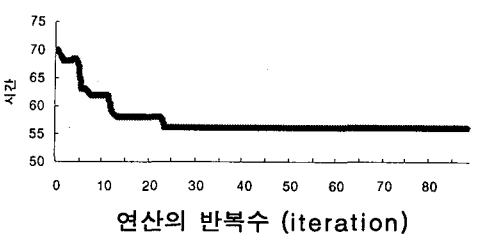
t_i : 엘리트 i의 처리시간

p : 유전연산 종료 후 개체군의 엘리트 집합

n : 개체군의 크기 (population size)

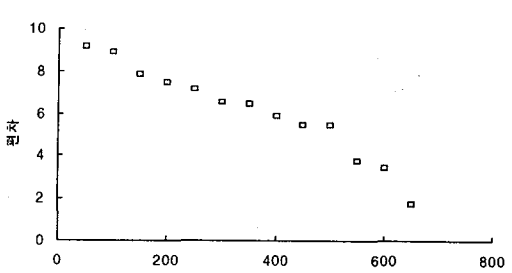
- 연구 결과분석

문제유형을 선택하여 유전연산의 반복수를 조절해 가며 연산한 결과 [그림4]와 같이 유전연산의 반복수가 증가 할수록 처리시간이 단축되었고, 반복수가 어느 시점에 이르면 더 이상의 시간단축은 일어나지 않았다. 알고리즘 종결 이전의 최소 처리시간의 해를 가장 우수한 해(best solution)로 하였다. 엘리트 개체군은 유전연산의 반복수에 비례하여 생성됨을 확인할 수 있었다. 유전연산의 반복수에 따른 최소 수행시간은 큰 차이를 보이지 않았지만, 반복회수가 많은 경우 우수한 형질의 엘리트 개체군이 더 넓게 분포하므로 좋은 해를 선택할 확률이 높아질 수 있음을 알 수 있다.



[그림 4]. 유전연산 반복 후 처리시간

유전연산의 반복수를 달리하여 생성되는 개체군과 최적해와의 편차를 <표4>에서 비교하였다. 유전연산을 많이 반복할수록 최적해와 가용한 개체군과의 편차가 줄어드는 것을 확인할 수 있다. 유전연산 반복수에 따른 최적해와 개체군과의 편차는 [그림 5]와 같이 감소함을 알 수 있다.



[그림 5]. 최소 makespan과 개체군 편차

<표4> 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

7. 결론

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 도출할 수 있을 것이다.

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

참고문헌

[1] 여한구, 이종태, "PERT/CPM에서의 프로젝트 완료시간 예측과 주경로 파악 및 통제를 위한 퍼지 기법의 응용", 산업기술논문집, Vol.13 No.- [1999], 149-160

[2] Kim, Seung-Ryeol, "Fuzzy PERT Applications for System Development Scheduling", 북악정보기술논집, Vol.9 No.- [2003] 1-14

[3] Maciej Hapke, Andrzej Jaszkiewicz, Roman Slowinski, "Fuzzy project scheduling system for software development", Fuzzy Sets and Systems 67 (1994) 101-117

[4] 김정자, 공명달, "자원제약하의 복수 프로젝트 일정계획을 위한 휴리스틱 알고리즘", 대한산업공학회지, Vol.13 No.1 [1987] 110-119

[5] 이화기, 정제원, "Constraint Programming을 이용한 자원제약 동적 다중프로젝트 일정계획", 산업공학, Vol.12 No.3 [1999] 362-373

[6] 윤종준, 이화기, "자원제약하의 동적 다중 프로젝트 일정계획에 Tabu Search 적용" 산업경영시스템학회지, Vol.22 No.52 [1999] 297-309

[7] 윤종준 이한기, "Tabu Search와 Constraint Satisfaction Technique를 이용한 Job Shop 일정계획", 산업경영시스템학회지 Vol.25 No.71 [2002] 92-101

[8] Min Qiu, "Prioritising and scheduling road projects by genetic algorithm", Mathematics and Computers in Simulation 43 (1997) 569-574

[9] Sou-Sen Leu, An-Ting Chen, Chung-Huei Yang, "A GA-based fuzzy optimal model for construction time-cost trade-off", Intemational Journal of Project Management 19 (2001) 47-58

[10] Linet Ozdamar, "A Genetic Algorithm Approach to a General Category Project Scheduling Problem", IEE Trans. on Syst., Man, and Cybern., vol. 29 February 1999

[11] KwanWoo Kim, YoungSu Yun, JungMo Yoon, Mitsuo Gen, Genji Yamazaki, "Hybrid genetic algorithm whth adaptive abilities for resource-constrained multiple project scheduling", Computers in Industry 56 (2005) 143-160

[12] Masao Mori, Ching Chih Tseng, "A genetic algorithm for muti-mode resource constrained project scheduling problem", European Joural of Operational Resrarch 100 (1997) 134-141

[13] Amir Azaron, Cahit Perkgoz, Masatoshi Sakawa, "A genetic algorithm approch for the time-cost trade-off in PERT networks", Applied Mathematics and Computation (2004)

[14] Liu, S.-x., "Genetic Algorithm for Resource Levelling Problem in Multi-mode Project Scheduling", CONTROL AND DECISION Vol.16 No.1 [2001]

[15] Mintzberg, Henry, "Power of In and Around Organizations", Englewood Cliffs, N.J., Prentice-Hall Inc., 1983

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

[17] 장여일, 서정윤, "프로젝트 관리도구로서의 CCPM의 실효성에 관한 연구", 인제농촌, Vol 14, No.2 (1998.12.31)

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

[19] 전영준, 김동연, 김진일, "유전자 알고리즘을 이용한 효과적인 퍼지 분류 방법에 관한 연구", 동의대학교 산업기술연구지 제 15권, pp.213-219, 2001. 2