Journal of the Korean Institute of Industrial Engineers
Vol. 43, No. 1, pp. 001-011, February 2017.
ISSN 1225-0988 | EISSN 2234-6457
https://doi.org/10.7232/JKIIE.2017.43.1.001

<Original Research Paper>

확률적 자원제약 스케줄링 문제 해결을 위한 가변 이웃탐색 기반 동적 의사결정

임 동 순†

한남대학교 산업경영공학과

Dynamic Decisions using Variable Neighborhood Search for Stochastic Resource-Constrained Project Scheduling Problem

Dong Soon Yim

Department of Industrial and Management Engineering, Hannam University

Stochastic resource-constrained project scheduling problem is an extension of resource-constrained project scheduling problem such that activity duration has stochastic nature. In real situation where activity duration is not known until the activity is finished, open-loop based static policies such as activity-based policy and priority-based policy will not well cope with duration variability. Then, a dynamic policy based on closed-loop decision making will be regarded as an alternative toward achievement of minimal makespan. In this study, a dynamic policy designed to select activities to start at each decision time point is illustrated. The performance of static and dynamic policies based on variable neighborhood search is evaluated under the discrete-event simulation environment. Experiments with J120 sets in PSPLIB and several probability distributions of activity duration show that the dynamic policy is superior to static policies. Even when the variability is high, the dynamic policy provides stable and good solutions.

Keywords: Stochastic Resource-Constrained Project Scheduling Problem, Variable Neighborhood Search, Closed- Loop Decision Making

1. 서 론

프로젝트 수행에 필요한 단위 작업과 요구되는 자원이 주어졌을 때 자원제약 스케줄링 문제(Resource-Constrained Project Scheduling Problem)는 자원 능력을 만족하면서 프로젝트 종료시각을 최소화하는 스케줄의 생성을 목적으로 한다. 확률적 자원제약 스케줄링 문제(Stochastic Resource-Constrained Project Scheduling Problem)는 자원제약 스케줄링 문제에 작업시간의 불확실성이라는 현실적인 상황이 부가된다. 이에 대한 대부분의 연구들은 작업시간에 대한 확률분포를 정의한 후 기대 종료시각의 최소를 가져오는 정책(policy)을 생성하거나 또는 어떠한 상황에서도 스케줄이 기준 스케줄(baseline schedule)에 크게 벗어나지 않게 되는 안전한 정책의 생성을 목적으로 한다. 각 단위작업의 시작시간이 포함된 확정적 스케줄은 확률적 자원제약 스케줄링 문제의 해로 적합하지 않기 때문에 보다 유연한 정책을 해로 생성한다.

정책은 확정적 작업시간을 가진 AON(Activity-On-Node) 네트워크와 자원제약을 입력으로 했을 때 스케줄을 생성하는 함수로 정의될 수 있다(Stork, 2001). 정책은 프로젝트 실행 중 의사결정이 필요한 시점(프로젝트 시작 시점 또는 한 작업이 종료된 시점)에서 어떤 작업을 시작하여야 하는지에 대한 답을 제공한다.

본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었음(UD140022PD).
† 연락저자 : 임동순 교수, 34430 대전광역시 대덕구 오정동 133 한남대학교 산업경영공학과, Tel : 042-629-7399, Fax : 042-629-8004, E-mail : dsyim@hnu.kr
2016년 11월 30일 접수; 2016년 12월 20일 1차 수정본 접수; 2017년 1월 23일 2차 수정본 접수; 2017년 1월 24일 게재 확정.

정책 중 가장 보편적인 것은 단순히 작업의 순서를 나타내는 작업리스트이다. 프로젝트 실행 이전에 작업리스트를 결정하고, 실행 시 이에 따라 작업 시작 순서가 결정되는 개방루프 형태의 의사결정에서 이러한 정책들은 정적인 특성을 갖는다. 확률적 자원제약 스케줄링에 대한 대부분의 연구들은 다양한 알고리즘을 적용하여 이러한 정적인 정책의 도출을 목적으로 하였다. GRASP(Greedy Randomized Adaptive Search Procedure)(Ballestin and Leus, 2009), 우선순위 규칙 기반의 SDGS (Stochastic Dynamic Generation Scheme)(Bruni *et al*., 2011), 유전알고리즘(Ballestin, 2004, 2008), 타부 탐색(Tsai and Gemmil, 1998) 등이 이러한 목적의 알고리즘에 이용되었다.

그러나, 정적인 정책이 프로젝트 종료 시까지 그대로 적용되기 어렵다. 실제적으로 발생된 작업시간이 계획단계에서 정의한 확률 분포를 벗어나는 상황이 발생할 수 있다. 예측이 빗나가면 계획이 수정되어야 하고, 정책이 변경되어야 한다. 설령 정의된 확률분포의 범위에 따르는 값으로 실현되었다고 해도 최초에 작성된 정책에 따라 나머지 작업들에 대한 시작 순서를 유지하는 보수적인 정책은 바람직하지 않다. 현실적인 상황에서는 한 작업이 종료되었을 때 마다 현재 정보와 갱신된 예측정보를 바탕으로 새로운 정책을 생성하고, 이에 따라 시작하여야 할 작업을 결정하는 것이 최적의 프로젝트 달성에 기여할 수 있다. 즉, 폐쇄루프에 기반하여 생성되는 동적인 정책이 개방루프 기반의 정적인 정책에 비해 더 효과적일 수 있다. 확률적 자원제약 스케줄링 문제에서는 개방루프 기반의 정책에 비해 폐쇄루프 기반의 정책에 대해서는 많은 연구가 이루어지지 않았다.

Ferandez *et al*.(1998)은 확률적 자원제약 스케줄링 문제에 다기간 확률적 계획법 개념을 적용하여 폐쇄 루프 기반의 동적인 의사결정을 모색하였다. 최근의 연구로는 Li와 Womer(2015)의 연구가 있다. 동적인 의사결정을 위해 폐쇄루프 근사 동적계획법(closed-loop approximate dynamic programming)을 제안하였다. 프로젝트 실행 중 의사결정이 필요한 시점에서 시작하여야 할 작업을 선택하기 위한 최적화 절차가 수행되었다. 절차는 현 시점에서 시작 가능한 작업(선후행 조건과 자원제약 조건을 만족하는 작업)들을 선정하는 것과 이들을 대상으로 한 평가로 구성된다. 평가는 몬테카를로 시뮬레이션을 이용하였다. 시작 가능한 각 작업에 대해서 시작되었다고 가정한 한 번의 시뮬레이션 실행에서는 남아있는 작업들에 대한 작업시간을 생성하고, 제약 프로그래밍을 적용하여 프로젝트 종료까지의 스케줄을 생성하였다.

본 연구에서는 동적인 의사결정을 위해 시간에 따라 변하는 확률적 자원제약 스케줄링 문제를 다룬다. 의사결정이 필요한 시점마다 새로운 문제를 생성하고, 시작하여야 할 작업을 결정하는 폐쇄루프 기반의 정책 생성을 목적으로 한다. 확률적 자원제약 스케줄링 문제를 풀기 위한 기본적인 방법으로 가변이웃탐색(VNS : Variable Neighborhood Search)(Hansen *et al*., 2010) 알고리즘을 이용한다. 본 논문에서는 동적인 의사결정을 위해 문제에 맞게 수정된 VNS를 적용하였다. 동적인 의사결정 절차를 평가하기 위한 방법은 기존의 정적인 정책을 평가하는 방법과는 상이하여야 한다. 몬테카를로 시뮬레이션 기반의 평가보다는 실제와 유사한 가상 상황을 만든 시뮬레이션 환경에서의 평가를 수행하는 방법론을 제안하였다.

논문의 구성은 다음과 같다. 제 2장에서는 동적인 확률적 자원제약 스케줄링 문제에 대한 정의를 하고, 제 3장에서는 문제를 풀기 위한 가변 이웃탐색 알고리즘을 설명한다. 제 4장에서는 동적인 의사결정의 평가를 위한 이산사건 시뮬레이션 기반의 실험방법을 설명하고, 제 5장에서는 표준 데이터를 이용하여 가변 이웃탐색 알고리즘 기반의 정책에 대해 성능분석을 수행한 결과를 소개한다.

2. 동적인 확률적 자원제약 스케줄링 문제

확률적 자원제약 스케줄링 문제는 다음과 같이 정의된다. AON으로 표현된 자원제약 프로젝트의 네트워크를 $G=(V, E)$라고 하자. 노드 집합 $V=\{0, 1, 2, \cdots, n, n+1\}$는 작업의 집합을 의미하며 작업 0과 $n+1$은 프로젝트의 시작과 종료를 나타내는 가상 작업이다. 간선 집합인 E는 작업 간 선후행 조건을 나타낸다. $K=\{1, \cdots, m\}$는 프로젝트 수행에 필요한 자원의 집합을 나타내고 복원 가능한 자원 $k\in K$는 단위 시간 당 제한된 용량 R_k를 가지고 있다. 작업 j는 실행 시 자원 k를 r_{jk}만큼 필요로 한다. 작업 j의 기간 d_j는 주어진 분포에 따르는 확률변수이다. 이러한 네트워크에서의 확률적 자원제약 스케줄링 문제는 프로젝트 종료 시각의 기대값을 최소화하는 정책을 도출하는 것을 목적으로 한다.

정책 중 가장 보편적인 것은 작업의 순서를 나타내는 작업리스트에 의한 것으로 두 종류가 있다. 정책이 실행될 때 반드시 리스트의 순서에 따라 작업들을 시작하여야만 한다면 작업 기반 정책(activity-based policy)이라 하고, 가능한 그 순서를 지키는 경우에는 우선순위 기반 정책(priority-based policy)이라 한다. 따라서, 작업 기반 정책에 따르는 작업리스트들은 작업 간의 선후행 조건을 만족하도록 구성되어야 한다. 리스트의 k번째 위치에 있는 작업의 선행 작업들은 항상 k번째 이전에 위치한다. 일반적으로 작업리스트를 언급할 때 이러한 선후행 조건을 만족하는 리스트를 의미하며 유효한 리스트라고 불리기도 한다.

반면, 우선순위 기반 정책에 따르는 작업리스트들은 반드시 작업들의 선후행 조건을 만족하지 않아도 된다. 작업 기간이 확정적인 값으로 주어졌을 때 순차적 스케줄 생성(sequential schedule generation scheme)(Kolish and Hartmann, 1999)과 같은 방법에 의해 리스트들로부터 작업들의 시작 순서를 생성할 수 있다. 일반적으로 우선순위 기반 리스트는 몇가지 단점을 가지고 있다고 알려져 있다. 최적 일정을 생성할 수 없을 뿐만 아니라 작업 기간이 감소했음에도 불구하고, 프로젝트 종료시각이 증가하게 되는 이상 현상이 발생한다(Graham, 1966; Ballestin and Leus, 2009; Li and Womer, 2015).

현실적인 상황에서는 프로젝트 실행 중 한 작업이 종료되었

을 때 마다 남아 있는 작업 중 어느 작업들을 시작시켜야 하는지에 대한 의사결정이 필요하다. 이러한 의사결정을 위한 작업리스트 기반의 정책들은 프로젝트 시작 이전 계획단계에서 생성된 확률적 자원제약 스케줄링 문제의 해를 바탕으로 이루어지므로 정적인 특성을 지닌다.

그러나, 현실적으로는 풀어야 할 확률적 자원제약 스케줄링 문제는 시간에 따라 동적으로 변한다. 프로젝트 실행 중 어느 시점에서 특정 작업이 종료되었다고 하자. 현재시각 T에서의 모든 작업들은 3가지로 종료된 작업 A_f, 현재 진행 중인 작업 A_a, 그리고, 앞으로 시작되어야 할 작업 A_s로 나뉜다. AON $G'=(V', E')$는 $G=(V, E)$의 부분 그래프로 $V'=V-A_f=\{0, 1\}\cup A_a\cup A_s$이다. E'은 두 가지 종류의 선으로 구성된다. 첫 번째 종류는 E에 속하는 선 중 V'들끼리 연결된 선 이고, 두 번째 종류는 A_f에서 A_a 또는 A_s를 연결하는 선 $e\in E$를 가상 작업 0에서 시작하는 선으로 변경한 것이다. 단, 두 작업을 연결하는 중복된 선을 허용하지 않는다. A_a에 속하는 작업 j의 기간 d_j'은 앞으로 남은 나머지 기간에 대한 확률변수이고, A_s에 속하는 작업들의 기간 d_j'은 확률변수로 d_j와 동일하다. 문제 정의가 이루어지는 시각을 0으로 재설정하자. A_a에 속하는 작업 j들의 시작시간은 0으로 고정되는 제한 조건을 갖는다. 자원 용량은 현재 진행중인 작업의 자원 사용량을 감안하여야 한다. A_a에 속하는 작업 j의 작업기간인 확률변수 d_j'를 생성한 값을 D_j'라고 하자. R_k 대신에 k번째 자원에 대한 시각 t에서의 용량을 R_{kt}로 재정의하자.

$$R_{kt}=R_k-\sum_{\substack{j\in A_a\\ D_j'>t}} r_{jk}$$

동적인 확률적 자원제약 스케줄링 문제는 AON $G'=(V', E')$에서 재정의 된 자원 용량을 고려하여 프로젝트 종료시각의 기대값을 최소화하기 위해 현 시점에서 시작하여야 할 작업 $j\in A_s$들을 결정하는 것이다.

동적인 정책을 도출하기 위해서는 프로젝트 실행 중 한 작업이 종료되었을 때마다 동적인 확률적 자원제약 스케줄링 문제를 생성하고, 문제에 대한 해를 구한다. 해로부터 현재 시점에서 시작 가능한 작업을 결정한다.

3. 확률적 자원제약 스케줄링 문제에 적용되는 가변 이웃탐색

3.1 가변 이웃탐색

가변 이웃탐색 알고리즘은 부분 탐색과 전역 탐색의 장점을 살리기 위해 개발된 메타 휴리스틱의 일종이다(Hansen *et al*., 2010). 자원제약 스케줄링 문제에서도 가변 이웃탐색 알고리즘이 우수한 해를 생성한다고 보고되었다(Fleszar and Hindi, 2004). 본 연구에서는 다양한 가변 이웃탐색 방법 중 확률적 자원제약 스케줄링 문제에 효과적이라고 판단되는 일반적 가변 이웃탐색(general VNS) 알고리즘을 적용하였다.

Algorithm : General VNS

0 Select the set of neighborhood structures N_k, for $k=1, \cdots, k_{\max}$
Select the set of neighborhood structures N_l, for $l=1, \cdots, l_{\max}$
Find an initial solution A
Choose a stopping condition

1 **Repeat** the following sequence until the stopping condition is met

1.1 $k=1$

1.2 **Repeat** the following steps until $k=k_{\max}$

1.2.1 Shaking : Generate a point A' at random from the k th neighborhood $N_k(A)$ of A

1.2.2 Local search by VND

1.2.2.1 $l=1$

1.2.2.2 **Repeat** the following steps until $l=l_{\max}$

1.2.2.2.1 Exploration of neighborhood : Find the best neighbor A'' of A' in $N_l(A')$;

1.2.2.2.2 Move or not : If $f(A'')<f(A')$ set $A'=A''$ and $l=1$; otherwise $l=l+1$;

1.2.3 Move or not : If this local optimum is better than the incumbent, move there ($A=A''$), and continue the search with $N_l(k=1)$; otherwise, set $k=k+1$;

알고리즘의 단계 0에서 생성된 초기해 $A\{a_1, a_2, \cdots, a_n\}$는 유효한 리스트로 선후행 조건을 만족한다. 단계 1.2.1에서 셰이킹(shaking)으로 구한 현재해 A의 이웃해인 A'을 입력으로 하여 단계 1.2.2에서 부분 최적화에 기초한 가변 이웃하강(VND : Variable Neighborhood Descent)을 적용한다. 단계 1.2.3에서는 가변 이웃하강에 의한 가장 좋은 해 A''가 현재해 A보다 우수하면 이를 현재해로 치환한다. 해 A의 성능 척도인 $f(A)$는 프로젝트 종료시각의 기대값이다. 이러한 절차는 보통 최대 실행시간으로 정의되는 알고리즘 종료 조건을 만족할 때까지 반복된다.

알고리즘에서 작업 기간이 확률변수의 값으로 주어졌을 경우 유효한 해에 대한 일정과 종료시각을 구하기 위해서는 작업들이 가능한 빠르게 시작되도록 하는 전진 스케줄링(forward scheduling) 방법을 적용한다.

현재해 A에 대한 셰이킹은 리스트에서 임의의 두 작업을 선택하여 서로 위치를 바꾸는 교환 연산만을 수행토록 하였다 ($k_{\max}=1$). 셰이킹 결과로 생성된 해는 유효한 해가 아닐 수 있다. 유효하지 않은 해는 다음에 설명될 리스트 스케줄링에 의해 유효한 해로 수정된다.

교환연산의 횟수에 따라 셰이킹의 강도가 다르게 된다. 교환횟수가 적으면 현재해에서 가까운 이웃해를 생성하지만 교환횟수가 크면 클수록 현재해에서 멀리 떨어진 해를 생성하여 셰이킹의 강도가 크게 되는 효과를 가져온다.

가변 이웃하강에서의 부분최적화 방법으로 반복적 전/후진 향상(Repetitive Forward/backward improvement)(Li and Willis, 1992)과 교환 연산(Yim, 2011)을 고려하였다. 세부적인 알고리즘은 다음과 같다.

Algorithm : RFBI(A')

0 Set initial solution A' to current solution A''
1 Repeat
1.1 Generate neighbor $N(A'')$ by applying FBI(A'')
1.2 If $f(N(A'')) < f(A')$ set $A'' = N(A'')$; otherwise return A''

Algorithm : FBI(A'')

0 Apply forward scheduling to A''
1 Generate $N(A'')$ by sorting activities in the scheduling in decreasing order of finish time
2 Apply backward scheduling to $N(A'')$
3 Generate $N(A'')$ by sorting activities in the schedule in increasing order of start time
4 Return $N(A'')$

Algorithm : ExBestAccept(A')

0 Set initial solution A' to current solution $A'' = \{a_1, a_2, \cdots, a_n\}$
Set flag = true
1 Repeat while flag = true
1.1 Set flag = false
Set temporary solution $A^* = A''$
1.2 For each (a_i, a_j), $i < j$
1.2.1 Generate neighbor $N(A'')$ by exchanging (a_i, a_j) in A''
1.2.2 If $N(A'')$ is valid and $f(N(A'')) < f(A^*)$, set $A^* = N(A'')$ and flag = true
1.3 If (flag = true) set $A'' = A''$
2 Return A''

Algorithm : ExFirstAccept(A')

0 Set initial solution A' to current solution $A'' = \{a_1, a_2, \cdots, a_n\}$
Set flag = true
1 Repeat while flag = true
1.1 Set flag = false
1.2 For each (a_i, a_j), $i < j$
1.2.1 Generate neighbor $N(A'')$ by exchanging (a_i, a_j) in A''
1.2.2 If $N(A'')$ is valid and $f(N(A'')) < f(A'')$, set $A'' = N(A'')$ and flag = true and break
2 Return A''

알고리즘 RFBI는 반복적 전/후진 향상 방법으로 전진, 후진 스케줄링을 더 이상의 향상이 없을 때까지 반복한다. 스케줄링을 위해서는 확정적인 작업 시간이 주어져야 한다. 본 연구에서는 각 작업의 소요기간 기대치 $E[d_j]$를 이용하여 스케줄링이 이루어지도록 하였다.

교환은 셰이킹에서 사용된 방법과 유사하게 리스트의 작업쌍에 대해 서로 위치를 바꾸는 연산이다. 부분 최적화를 적용하는데 있어 우선 채택(first-accept)과 최선 채택(best-accept)[1]의 두 가지 전략이 존재한다. 우선 채택에서는 이웃해를 탐색하다가 가장 먼저 향상을 가져 오는 해를 가장 좋은 이웃해로 간주하고 이를 새로운 초기해로 하여 탐색을 반복한다. 최선 채택에서는 모든 이웃해를 탐색한 후 이 중 가장 큰 향상을 가져오는 해를 새로운 초기해로 한다. 두 알고리즘 ExBestAccept와 ExFirstAccept은 각각 두 전략을 구현한 것이다.

3.2 가변 이웃탐색 알고리즘 해의 이용

제 5장의 실험 및 분석에서 동적인 의사결정에 이용된 정책은 작업 기반, 우선순위 기반, 그리고, 동적 정책이다. 가변 이웃탐색 알고리즘의 해를 이용한 이들 정책들을 설명하기 위해 <Figure 1>의 네트워크를 예로 든다. 네트워크에서 작업 0과 10은 가상 작업이다. 이들을 제외한 각 작업에 명시된 숫자는 소요기간의 기대치/자원 요구량을 의미한다. 각 작업 기간의 분포는 주어졌다고 가정한다. 자원은 한 종류로 용량은 2이다.

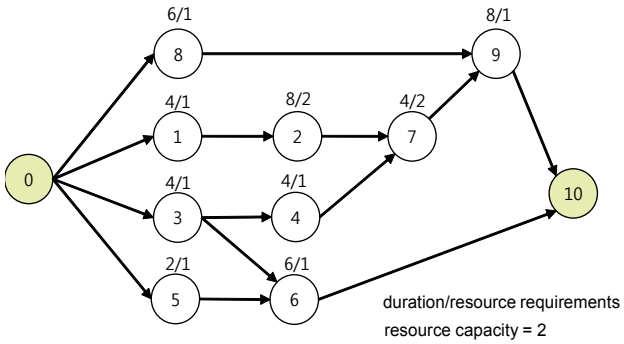


Figure 1. An Example AON

가변 이웃탐색 알고리즘에서의 해는 작업리스트로 선후행 조건이 만족되는 유효한 해이다. 초기해 A를 생성하기 위해 우선 랜덤의 해를 생성한다. 초기 랜덤의 해 A_0를 {9, 8, 7, 6, 5, 4, 3, 2, 1}이라고 하자. 가상 작업들은 리스트에서 제외된다. 이 해는 선후행 조건을 만족하지 않아 리스트의 순서에 따라 작업을 시작할 수 없다. 예를 들어 가장 처음 순서의 9번 작업을 시각 0에서 시작할 수 없다. 7, 8번 작업이 끝나야 9번 작업

이 시작될 수 있기 때문이다. 이 해를 유효한 작업리스트 A으로 바꾸기 위해 다음과 같이 순차적 스케줄 생성 방법을 수행한다. 초기에 A는 비어있는 리스트이다.

첫 번째 단계에서 A_0에서 시작 가능한 작업을 탐색하여 이를 A에 추가한다. 탐색순서는 리스트의 순서에 따른다. 작업 8이 시작 가능하므로 A_0서 8을 제거하고, A에 추가하면 $A_0 = \{9, 7, 6, 5, 4, 3, 2, 1\}$, $A = \{8\}$이다. 두 번째 단계에서는 리스트 A_0의 순서에 따라 작업 5가 시작 가능하여 이를 제거한 후 A에 추가하면 $A_0 = \{9, 7, 6, 4, 3, 2, 1\}$, $A = \{8, 5\}$가 된다. 이 같은 절차를 $A_0 = \Phi$가 될 때까지 반복하면 작업리스트 $A = \{8, 5, 3, 4, 6, 1, 2, 7, 9\}$를 생성한다.

프로젝트 시작 시점인 t = 0에서 시작할 작업을 결정하기 위해 네트워크와 자원 용량을 입력으로 하는 확률적 자원제약 스케줄링 문제에 가변 이웃탐색을 적용한다. 초기해 A를 입력으로 가변 이웃탐색 알고리즘에서 생성한 유효한 해를 {1, 3, 2, 4, 5, 8, 6, 7, 9}라고 하자. 이 리스트가 정적 정책 하에서의 작업리스트가 된다.

시각 0에서는 동적 또는 정적 정책 모두 동일한 작업리스트를 이용하므로 리스트의 순서에 따라 작업 1과 3을 시작시킨다. 두 작업 중 먼저 끝나는 작업 1의 실제 종료 시각이 3이라고 하자. 이 시각(t = 3)에서 새로 시작할 작업을 결정하여야 한다. 정적인 작업 기준 정책 하에서는 수량 2의 자원을 필요로 하는 작업 2가 다음 순서이지만 자원 초과로 시작시킬 수 없어 작업 3이 종료될 때까지 대기하여야 한다. 반면 우선순위 기반 정책 하에서는 작업순서에 따라 탐색하여 시작 가능한 작업 5를 선정하고 시각 3에서 시작시킨다.

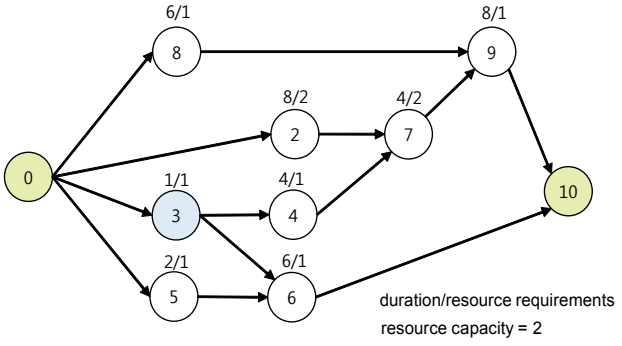


Figure 2. Dynamically Created AON at t = 3

동적인 확률적 자원제약 스케줄링 문제에서는 $A_f = \{1\}$, $A_a = \{3\}$, $A_s = \{2, 4, 5, 6, 7, 8, 9\}$로 A_f가 제거된 네트워크는 <Figure 2>와 같다. 현재 시각은 0으로 재설정된다. 현재 진행중인 작업 3은 시각 0에서 시작되는 것으로 고정시키고, 작업시간 기대치를 1로 감소시킨다. 이러한 문제에 대해 가변 이웃탐색 알고리즘을 실행시킨다. 초기해는 이전 단계에서의 해에서 A_f와 A_a를 제거한 {2, 4, 5, 8, 6, 7, 9}로 한다. 가변 이웃탐색 알고리즘으로 구한 해를 {4, 2, 5, 8, 6, 7, 9}이라고 하자. 현재 시각에서 시작할 작업을 선택하기 위한 동적 정책은 우선순위 기반 정책에 기초한다. 첫번째 순서인 작업 4는 작업 3이 끝나지 않아 시각에서 시작시킬 수 없다. 반면 다음 순서인 작업 2는 자원제약으로 인해 현재시점에서 시작시킬 수 있다. 따라서, 동적 정책 하에서는 그 다음 순서인 작업 5를 시작시킨다.

동적인 확률적 자원제약 스케줄링 문제를 생성하기 위해서는 현재 진행 중인 작업에 대한 작업시간 기대치와 확률분포를 결정하여야 한다. 본 연구에서는 원래의 작업시간 기대치에서 작업 시작 후 현재 시각까지 진행된 시간을 뺀 값을 새로운 기대치로 하였다. 만약 이 값이 0보다 작다면 최소 기간인 1로 설정하였다. 확률분포는 기대치, 분산 등의 파라미터를 제외하고, 원래 정의된 확률분포 함수에 따르는 것으로 하였다. 남아있는 용량의 설정은 확률분포로부터 생성된 작업시간에 따라 결정된다. 만약, 작업 3의 생성된 작업시간이 1이라면 $R_0 = 1$이고 나머지 기간에 대해 $R_t = 2, t > 0$가 된다. 이 같이 남아 있는 용량의 결정은 해에 대한 성능척도인 프로젝트 종료시각의 기대치를 구할 때 모든 작업시간에 대한 인스턴스가 생성된 후 결정된다.

4. 이산사건 시뮬레이션을 이용한 동적 정책 성능 평가

정책들의 성능 분석을 위해서는 실제와 유사한 가상적인 상황에서 어느 정도의 효과가 있는지를 평가하는 절차가 필요하다. 정적인 정책을 대상으로 하는 경우의 성능 평가는 몬테카를로 시뮬레이션을 이용할 수 있다. 한 번의 실행에서 확률 변수인 작업 시간에 대한 인스턴스를 포함하는 시나리오를 생성하고, 시나리오 하에서 특정 정책에 의한 스케줄을 생성한다. 이러한 절차를 반복 실행하여 프로젝트 종료시각에 대한 기대치를 구한다. 몬테카를로 시뮬레이션 방식의 이러한 실험은 동적 정책의 평가에 적합하지 않다. 최대한 실제적인 상황과 유사한 환경 하에서 실험을 하여야 한다. 이를 위해서는 작업의 시작과 종료 사건을 발생시킬 수 있는 이산사건 시뮬레이션을 고려할 수 있다. Li *et al.*(2012)은 확률적 자원제약 스케줄링 문제에 이산사건 시뮬레이션을 이용하였지만 실제적인 상황에 대한 모의보다는 정적인 정책에 대한 평가를 목적으로 하고 있다. 그 외 대부분의 시뮬레이션 최적화 기반 연구들은 정적인 정책의 평가를 위한 몬테카를로 시뮬레이션을 적용하고 있다. 동적인 정책의 성능 분석을 수행한 Li and Womer (2015)의 연구에서는 이에 대한 언급은 없지만 이산사건 시뮬레이션과 유사한 방법을 사용하였을 것으로 추측된다.

동적 정책을 평가하기 위한 시뮬레이션은 스케줄링과 이산사건 시뮬레이션 엔진으로 구성되어 서로 메시지를 주고받음으로써 수행된다(<Figure 3> 참조). 스케줄링 모듈은 시뮬레이션 엔진으로부터 특정 작업이 종료되었다는 메시지를 받는 즉시 현 시점에서 시작하여야 할 작업을 결정하여 이를 시뮬레이션 엔진에게 통보한다. 랜덤샘플링을 통해 각 작업시간에 대한 시나리오를 생성한 시뮬레이션 엔진은 현재 시각에서 시

작되는 작업에 대한 메시지를 받아 그 작업이 종료될 미래사건을 계획한다. 또한, 미래사건 중 가장 빨리 발생될 사건을 선정하여 그 사건의 발생시각으로 시뮬레이션 시각이 진전되고, 종료되는 작업에 대한 메시지를 스케줄링 모듈에게 보내게 된다. 따라서, 시뮬레이션 엔진은 다음 사건 시간 진전(next-event time advance) 방식의 이산사건 시뮬레이션에 해당한다.

이산사건 시뮬레이션을 이용한 실험의 성능척도로는 시뮬레이션 엔진에서 생성한 작업 시간에 대해 이들을 완벽하게 예측하였다고 가정했을 때의 최적 종료시각과 평가 대상이 되는 정책 하에서의 종료시각 간 차이에 대한 기대치가 의미 있게 된다.

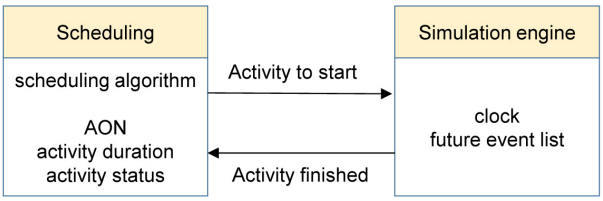


Figure 3. Scheduling Module and Discrete-Event Simulation

5. 실험 및 분석

5.1 실험 설정

가변 이웃탐색 기반 정책의 성능 분석을 위해 표준 문제인 PSPLIB(Kolish and Sprecher, 1996)에서 가장 복잡성이 큰 J120 문제들을 고려하였다. J120 문제는 120개의 작업을 포함하는 문제로 60개의 카테고리(X1~X60)로 구성되어 있고, 각 카테고리에 10개씩의 문제들이 있어 총 600개의 문제가 있다. 문제에 포함된 작업의 소요기간은 고정된 값으로 이를 기대치 $E[d]$로 간주하였다. 소요기간에 대한 확률분포로는 대부분 연구에서 사용된 5가지를 사용하였다(<Table 1> 참조).

가변 이웃탐색 알고리즘을 포함한 실험 환경은 자바 프로그래밍 언어로 구현되어 2.6 GHz의 Xeon CPU와 32GB RAM의 PC에서 실행하였다.

예비 실험결과로 도출된 결과에 따라 가변 이웃탐색 알고리즘의 파라미터를 설정하였다.

- 셰이킹에서 교환 연산의 횟수를 10으로 설정하였다.
- 우선 채택을 가변 이웃하강에서의 전략으로 적용하였다. 실험 결과 우선 채택이 최선 채택에 비해 우수한 성능을 보였다. 제한된 실행시간이 주어졌을 때 최선 채택이 제한된 지역 내의 이웃해들 탐색에 치중하는 반면, 우선 채택은 보다 넓은 범위의 다양한 해들을 탐색할 수 있는 장점을 가지는 것으로 보인다.
- 동적 정책을 위한 가변 이웃 탐색은 시뮬레이션 엔진으로부터 작업이 끝났다는 메시지를 받을 때 마다 실행된다. 120개 작업을 대상으로 하는 한 번의 시뮬레이션 실행에서 최대 120번의 가변 이웃탐색 실행이 발생한다. 시간에 따라 가변 이웃탐색에서 대상으로 하는 확률적 자원제약 스케줄링 문제의 작업 수는 최소 1씩 감소한다. 최초에는 120개 작업을 대상으로 하였지만 프로젝트가 종료되기 전 단계에서는 최소 하나의 작업만을 대상으로 한다. 시간에 따른 확률적 자원제약 스케줄링 문제 복잡성의 감소로 가변 이웃탐색의 종료 사건인 실행 시간 역시 감소토록 하였다. 남아 있는 작업 수가 n일 때(최초 실행시간-1초)/(총 작업 수-n+1)+1초를 실행시간으로 하였다.

5.2 자원제약 스케줄링 문제에서의 가변 이웃탐색 알고리즘 성능

본 연구에서 사용한 가변 이웃탐색의 성능을 평가하기 위하여 작업시간이 고정된 상수인 경우의 자원제약 스케줄링 문제에서 기존 연구 결과와 비교하였다. 비교 대상은 3가지로 유전 알고리즘, 제한적 이웃탐색 기반의 가변 이웃탐색 결과(Fleszar and Hindi, 2004), 그리고, 지금까지 알려진 가장 좋은 해이다. 유전 알고리즘은 가급적 Tseng and Chen(2006)의 연구에서 사용된 것과 동일하도록 구현하였다. 제한적 이웃탐색 기반의 가변 이웃탐색 알고리즘 성능은 논문(Fleszar and Hindi, 2004)에서 언급된 결과를 사용하였다. 지금까지 알려진 가장 좋은 해는 PSPLIB의 자료를 참조하였다.

유전 알고리즘은 가변 이웃탐색에서와 마찬가지로 유효한 작업리스트로 해를 표현하였다. 유효한 초기해를 만들기 위해 LFT(Latest Finish Time) 규칙을 적용한 후회 기반 편향 랜덤샘플링(regret based biased random sampling) (Kolish, 1996)을 이용하였다. 해에 대한 스케줄은 전진 스케줄링에 의해 구한다. 적자선택은 룰렛 휠 방식을 사용하고, 교배연산자로는 두 위치 전진과 두 위치 후진, 돌연변이 연산자로는 리스트에서 임의의 두 작업 위치를 바꾸는 교환 연산을 사용한다. 교배 확률과 돌연변이 확률은 각각 0.75와 0.05로 하였고, 개체수는 100으로 하였다.

Table 1. Probability Distribution of Activity Duration

	Type	Support	variance	Symmetric?
U1	Uniform	$(E[d]-\sqrt{E[d]},\ E[d]+\sqrt{E[d]})$	$E[d]/3$	Yes
U2	Uniform	$(0,\ 2E[d])$	$E[d]^2/3$	Yes
EXP	Exponential	$\mu = E[d]$	$E[d]^2$	No
B1	Beta	$(E[d]/2,\ 2E[d])$	$E[d]/3$	No
B2	Beta	$(E[d]/2,\ 2E[d])$	$E[d]^2/3$	No

자원제약 스케줄링 문제에 적용된 Fleszar and Hindi(2004)의 가변 이웃탐색은 다음과 같은 주된 면에서 본 연구에서의 가변 이웃탐색과 구별된다.

- 셰이킹에서는 작업리스트에서 랜덤의 작업을 선택하여 이를 다른 위치로 이동시키는 시프트 연산을 사용하였다.
- 가변 이웃하강에서 부분최적화를 적용하는 전략으로는 제한된 이웃해 탐색을 사용하였고 연산으로는 순차적인 시프트를 사용하였다. 제한된 이웃해 탐색에서는 작업리스트의 현재 위치에서 시작하여 오른쪽으로 7개의 작업에 대해 가능한 모든 다른 위치로의 시프트 대안을 고려한다. 이 중 가장 좋은 해를 현재해로 치환하고, 작업리스트를 환형이라고 가정하여 다음 7개 작업에 대해 이 절차를 반복한다. 만약, 7개 작업의 시프트가 현재해 보다 좋지 않은 경우가 연속 10번 나오면 알고리즘을 정지하도록 하였다.

X1부터 X60까지의 각 카테고리에 속한 10개 문제에 대해 가변 이웃탐색과 유전 알고리즘을 적용하여 프로젝트 종료시각을 구하였다. 두 알고리즘 모두 한 문제에 대해 2분 동안 실행하였다. <Table 2>는 각 문제에 대해 구한 주공정(critical-path)을 기준으로 한 편차율을 나타낸다. 지금까지 알려진 가장 좋은 해의 평균 편차율은 29.18%이다. 제한적 이웃탐색 기반의 가변 이웃 탐색(RVNS)는 33.10%로 알고리즘 중 가장 좋은 성능을 보인다. 본 연구에서의 가변 이웃탐색(VNS)은 33.27%로 비슷한 수준의 성능을 보인다. 유전 알고리즘(GA)은 44.05%로 가장 좋지 않은 성능을 보인다. RVNS에서는 알고리즘 종료사건이 최대 실행시간이 아니므로 문제에 따라 실행시간이 상이하다. 알고리즘 실행 중 생성된 스케줄의 수는 실행시간에 비례한다. RVNS에서 생성된 스케줄 수의 평균은 VNS가 다른 알고리즘에 비해 커 VNS가 RVNS보다 평균적으로 많은 실행시간을 갖는다. 그러나, RVNS에서 생성된 최대 스케줄 수는 VNS의 두배 이상이 된다. 비록 Pentium III 급 PC에서 최대 20분을 넘지 않았다고 논문(Fleszar and Hindi, 2004)에서 보고하였지만 복잡성이 작은 문제에는 실행시간이 작게 되나 복잡성이 커짐에 따라 실행시간이 증가했을 것으로 추측된다. 복잡성이 작은 문제의 수가 많아 평균 스케줄 수는 상대적으로 작아진다. 그러나, VNS에서는 동일한 2분의 실행시간을 할당하였으므로 모든 문제에서 비슷한 스케줄 수가 생성된다. 결과적으로 비슷한 편차율임에도 평균 스케줄 수는 RVNS에 비해 VNS가 1.7배 많다.

Table 2. Solution Deviation from Critical-Path and Number of Schedules Generated

Algorithm	Dev (%)		Schedules generated	
	Ave	Std	Ave	Max
VNS	33.27	44.58	3,266,192	4,890,642
GA*	44.05	50.80	1,067,893	1,258,600
RVNS**	33.10		1,874,641	10,778,083
Best-known solutions	29.18	40.55		

*Based on GA algorithm in Tseng and Chen(2006).
**Based on result in Fleszar and Hindi(2004).

5.3 확률분포를 알고 있는 경우 해의 평가

확률적 자원제약 스케줄링 문제를 풀기 위한 알고리즘에서는 해에 대한 평가를 하기 위해 몬테카를로 시뮬레이션을 이용한다. 해가 작업리스트로 표현되는 경우 한 번의 시뮬레이션 실행에서는 정의된 확률 분포로부터 각 작업 시간을 결정하는 시나리오를 생성하고, 특정 리스트 스케줄링에 의해 성능 척도인 프로젝트 종료시각을 구한다. 이러한 실행을 반복하여 얻어진 샘플들을 대상으로 기대 종료시각을 추정한다.

기대 종료시각을 보다 정확히 추정하기 위해서는 많은 횟수의 반복을 실행하여야 한다. 확률적 자원제약 일정문제에서는 1회의 실행에 소요되는 시간이 대체로 크지 않아 이러한 반복 실행은 큰 부담이 되지 않을 수 있다. 그러나, 가변 이웃탐색 등과 같은 부분 최적화에 기반한 알고리즘에서는 고려되는 이웃해의 수가 많아지고 이에 따라 실행 수가 증가하여 우수한 해를 생성하기 위해 적지 않은 실행 시간을 필요로 한다.

확률적 자원제약 스케줄링 문제를 대상으로 하는 대부분의 알고리즘에서는 해에 대한 정확한 기대 종료시각을 구하기보다는 두 해 중 어느 해가 더 좋은지를 판단하는 것이 중요하다. 예를 들어, 초기해에 대한 이웃해를 구하여 이 둘을 비교하여 해의 향상을 꾀하는 부분 최적화 알고리즘에서는 이웃해가 더 좋으면 이를 초기해로 하여 알고리즘을 반복 수행한다. 가변 이웃탐색 알고리즘도 이에 속한다. 초기해를 입력으로 부분 최적해를 구할 때 새로운 해와 이전의 해를 비교하여 어느 해가 더 좋은지 평가하여야 한다. 두 해에 대한 비교에서 작업 시간을 기대값 $E[d]$로 고정시켜 구한 평가 결과가 작업시간을 확률변수로 인식하여 몬테카를로 시뮬레이션을 통해 구한 평가결과와 크게 차이가 없다면 많은 횟수의 시뮬레이션 실행을 할 필요가 없다. 단지 한 번의 실행으로 해에 대한 비교 평가를 하면 된다.

<Figure 4>는 X1_1에 대해 5가지 확률분포를 적용한 문제에서 가변 이웃탐색 알고리즘에 의한 결과로 셰이킹을 통해 생성된 각 초기해 100개와 가변 이웃하강을 적용한 100개 부분최적해를 나타낸다. 알고리즘에서는 해의 평가를 위해 1회의 실행으로 $E[d]$를 이용하였다. 각 해에 대한 x값은 알고리즘에서 생성된 해에 작업시간 기대값을 적용한 프로젝트 종료시각을 나타내고, y 값은 동일한 해에 대해 5가지 확률분포의 샘플링으로 구한 종료시각으로 100개 샘플의 평균값을 나타낸다.

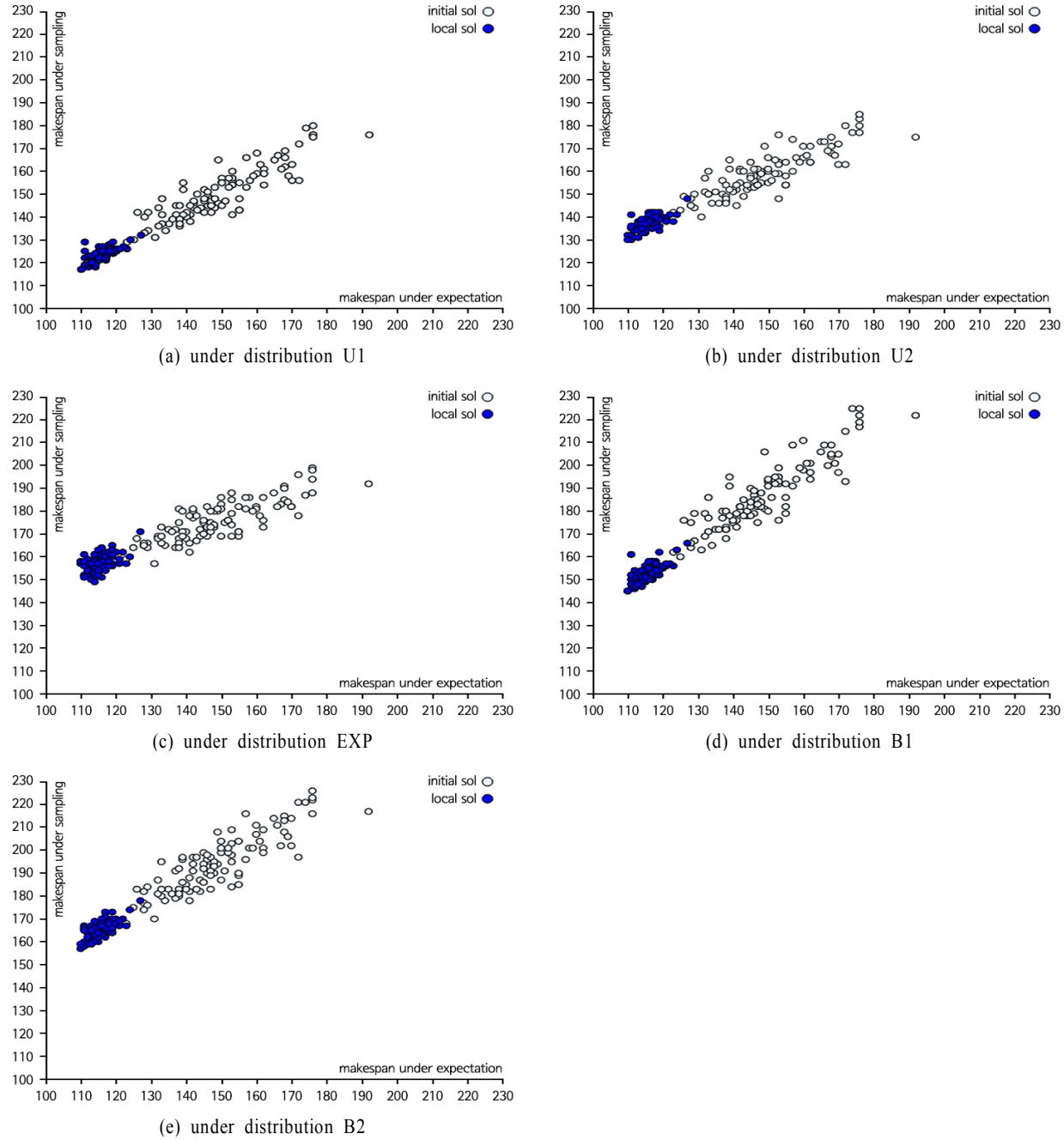
그림들은 작업시간을 상수 $E[d]$로 하여 구한 종료시각과 100회 실행으로 기대 종료시각을 추정한 값 사이에 차이가 있음을 나타낸다. 두 값에 차이가 없다면 $y = x$의 관계가 성립하여야 하지만 그런 관계는 보이지 않는다. 활동의 기간에 대한 5가지 모든 확률분포에서 x값 보다는 y값이 크므로 작업시간을 상수로 간주한 경우의 종료시각은 기대 종료시각보다 적게 계산되는 경향이 있음을 알 수 있다.

그러나, 두 값 간에 선형관계는 존재하여 x값이 큰 해는 대체로 y값도 크게 되는 경향을 보인다. 선형 관계가 있다는 것은 두 해를 비교할 때 작업시간을 상수로 고정시켜 구한 평가 결과와 100회 반복 실행하여 구한 평가결과간에 큰 차이가 없다는 것이다.

더욱이 초기해와 부분 최적해 간에 x, y값 모두에서 확연한 차이를 보여 작업시간을 상수 $E[d]$로 하는 경우의 가변 이웃탐색 알고리즘이 좋은 해를 생성할 수 있음을 의미한다. X1_1 외 다른 문제에 대해서 실험한 결과도 위와 유사한 결과가 도출되었다.

위 결과는 적은 샘플에 의한 몬테카를로 시뮬레이션으로 잘못된 기대 종료시각을 구하기보다는 작업시간을 고정된 상수 $E[d]$로 하여 해를 평가하는 것이 더 좋은 해를 생성할 수 있음을 시사한다. 그러나, 작업시간을 상수로 하여 구해진 해는 최적해가 아닐 수 있음에 유의하여야 한다. 보다 우수한 해를 구하기 위해서는 많은 횟수의 반복 실행이 필요할 수 밖에 없다. 결국 해의 질과 계산시간 부담 간에 트레이드 오프가 존재한다.

이러한 트레이드 오프를 해결하는 방법 중의 하나는 가변 이웃탐색 알고리즘의 셰이킹과 가변 이웃하강 연산에서는 $E[d]$에 의해 프로젝트 종료시각을 구하고, 셰이킹과 가변 이웃하강에 의해 구해진 부분 최적해에만 기대 종료시각을 추정토록 하는 것이다. <Figure 4>에서 보듯이 $E[d]$에 의한 가변 이웃탐색 알고리즘은 초기해에 비해 우수한 부분 최적해를 생성하는 것은 분명하다. 단지 부분 최적해 간의 정확한 비교가 필요할 뿐이다. 이를 위해 부분 최적해에 대해 보다 정확한 기대 종료시각을 추정하도록 한다.



(a) under distribution U1

(b) under distribution U2

(c) under distribution EXP

(d) under distribution B1

(e) under distribution B2

Figure 4. Relation between Makespan with Expected Duration and Sampled Makespan

구체적으로 가변 이웃탐색 알고리즘의 단계 1.2.3에서 부분 최적해 A''에 대해서는 100번의 반복에 의한 평균 종료시각 $f(A'')$을 구하고, 이를 현재해 A의 평균 종료시각 $f(A)$와 비교한다. 그 외 단계에서의 종료시각들인 $f(A)$, $f(A')$, $f(A'')$은 $E[d]$에 의한 1회의 실행으로 구한다.

5.4 가변 이웃탐색 기반 정책 성능 분석

본 논문에서 제안한 가변 이웃탐색 기반의 정책들에 대한 성능 분석을 제 4장에서 설명된 이산사건 시뮬레이션 기법을 이용하여 수행하였다. 실행시간의 부담으로 인해 J120의 60개 카테고리 중 6개를 선택하고, 선택된 카테고리의 10개 문제 중 한 문제를 선택하였다. 이렇게 선택된 6개 문제에 대해 5가지 작업시간 확률분포를 적용하여 총 30개 문제를 대상으로 하였다. 각 문제에 대해 이산사건 시뮬레이션 25회 실행으로 동적 정책, 우선순위 기반 정책, 작업 기반 정책들의 성능을 비교하였다. 3가지 정책 모두 최대 1분의 실행시간이 주어진 가변 이웃탐색의 해를 기반으로 수행되었다. 우선순위 기반 정책과 작업 기반 정책은 프로젝트 시작 전 계획단계에서 가변 이웃탐색 알고리즘으로 구한 해를 이용한다. 동적 정책은 동적으로 발생하는 확률적 자원제약 스케줄링 문제에 대한 가변 이웃탐색의 해를 바탕으로 한다. 성능척도로는 작업시간을 완벽히 정확하게 예측하였다는 가정하에서 구한 가변 이웃탐색 알고리즘의 해를 기준으로 각 정책의 실행 결과 구현된 프로젝트 종료시각의 편차율로 하였다.

<Table 3>는 각 문제에 대해 25회 시뮬레이션 실행한 결과의 평균과 표준편차를 나타낸다. <Figure 5>는 작업시간에 대한 각 확률분포에서 평균 편차율을 도식화한 것이다. <Figure 5>

Table 3. Policy Deviation from VNS Solution with Perfect Information

Problem	Dynamic policy		Priority-based policy		Activity-based policy		$H_0 : p_1 < p_2$	$H_0 : p_2 < p_3$
	ave(p_1)	Std	ave(p_2)	std	ave(p_3)	std	$P(t < t_0)$	$P(t < t_0)$
X10_3(U1)	3.08	2.13	8.19	3.17	16.58	5.09	1	1
X10_3(U2)	7.17	4.76	10.01	5.87	49.93	16.71	0.96	1
X10_3(EXP)	12.58	8.55	12.96	8.94	91.51	33.25	0.56	1
X10_3(B1)	3.24	2.56	8.66	2.80	13.70	4.36	1	1
X10_3(B2)	4.48	3.61	10.66	4.87	35.58	10.52	1	1
X20_3(U1)	5.55	2.30	8.52	2.78	15.09	2.58	1	1
X20_3(U2)	7.69	4.90	10.07	8.26	50.33	15.67	0.88	1
X20_3(EXP)	7.33	8.23	8.92	8.38	75.58	25.68	0.75	1
X20_3(B1)	4.88	2.41	7.25	2.84	15.98	5.57	1	1
X20_3(B2)	6.80	3.05	9.78	4.44	34.65	7.87	0.99	1
X30_3(U1)	2.14	2.02	5.15	3.97	11.77	5.43	1	1
X30_3(U2)	3.43	4.82	6.99	6.01	44.03	12.75	0.99	1
X30_3(EXP)	4.53	5.42	6.13	4.84	69.84	24.07	0.86	1
X30_3(B1)	1.62	1.58	4.90	2.66	11.53	4.86	1	1
X30_3(B2)	2.53	2.74	5.77	4.98	28.87	10.26	1	1
X40_3(U1)	3.93	1.98	6.49	2.64	14.98	5.58	1	1
X40_3(U2)	4.57	4.92	6.68	4.21	44.59	13.98	0.94	1
X40_3(EXP)	3.60	4.95	5.34	5.97	67.14	27.86	0.86	1
X40_3(B1)	4.51	2.25	6.16	2.18	14.19	4.10	0.99	1
X40_3(B2)	5.23	2.67	8.07	2.77	31.81	7.73	1	1
X50_3(U1)	2.18	1.79	5.89	2.85	12.34	4.46	1	1
X50_3(U2)	3.56	3.05	5.35	4.21	39.08	13.91	0.95	1
X50_3(EXP)	2.32	2.88	4.24	3.69	67.52	22.37	0.97	1
X50_3(B1)	2.43	1.95	7.14	3.85	12.09	6.67	1	1
X50_3(B2)	3.50	2.42	8.14	3.89	27.61	8.38	1	1
X60_3(U1)	3.98	1.73	6.68	2.26	13.30	3.64	1	1
X60_3(U2)	10.74	4.14	17.24	7.39	69.12	15.77	1	1
X60_3(EXP)	7.21	5.66	8.77	5.63	66.68	20.84	0.83	1
X60_3(B1)	4.25	2.21	6.99	2.51	13.07	3.25	1	1
X60_3(B2)	7.12	1.82	8.47	2.71	28.32	5.41	0.98	1

에 따르면 평균 편차율은 동적 정책, 우선순위 기반 정책, 작업 기반 정책 순으로 크다. 특히, 작업 기반 정책은 분산이 가장 큰 지수분포 문제에서 평균 73.05%의 편차율을 보여 6.26%, 7.73%의 평균 편차율를 보인 다른 두 정책에 비해 매우 열등한 성능을 보인다. 비교적 작은 분산의 U1 문제에서도 동적 정책에 비해 4배의 큰 편차율을 보인다. 동적 정책은 모든 분포에서 최소 3.48%, 최대 6.25%의 평균 편차율로 다른 정책들에 비해 매우 우수하고 안정적인 결과를 보인다. 우선순위 기반 정책은 최소 6.82%, 최대 9.39%로 비교적 안정적인 값을 보이지만 동적 정책에 비해 큰 편차율을 가진다.

<Table 3>의 마지막 두 열은 동적 정책과 우선순위 기반 정책의 차이, 우선순위 기반 정책과 작업 기반 정책의 편차율 차이에 대한 t 통계량을 확률로 변환한 값이다. U1, B1, B2 분포에서는 동적 정책이 우선순위 기반 정책에 비해 최소 98% 확률로 평균 편차율이 작다. U2에서는 최소 88%의 확률로 작고, EXP 분포에서는 최소 56%, 최대 97%의 확률로 작다. 따라서, 대체로 동적 정책이 우선순위 기반 정책에 비해 우수한 정책임을 나타낸다. 작업기반 정책은 다른 두 정책에 비해 100% 확률로 평균 편차율이 작아 가장 열등한 정책임을 나타낸다.

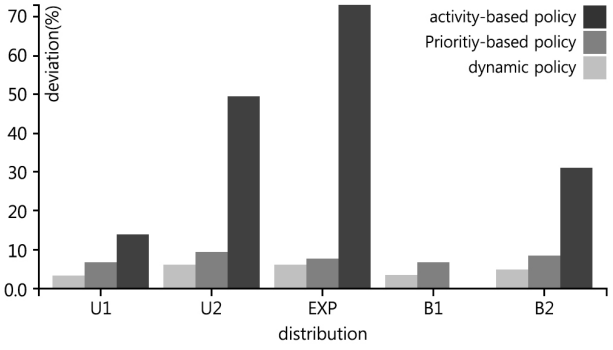


Figure 5. Policy Deviation Under Probability Distribution of Activity Duration

6. 결 론

본 논문에서는 동적인 의사결정이 요구되는 확률적 자원제약 스케줄링 문제를 다루었다. 동적인 의사결정에 이용될 수 있는 3가지 정책들인 작업 기반, 우선순위 기반, 동적 정책들을 비교 분석하였다. 각 정책에서 요구되는 우수한 해를 효과적으로 구하기 위해 확률적 자원제약 스케줄링 문제에 적합하도록 고안된 가변 이웃탐색 방법을 사용하였다. 가변 이웃 탐색에 포함된 부분 최적화 방법에서는 기대 종료시각을 정확히 구하기보다는 해들 간의 우열을 비교하기 위해 한 해에 대해 기대 작업시간을 이용하여 1회의 평가를 실시하였다. 그러나, 구해진 부분 최적해들에 대해서는 100회의 평가를 실시하여 성능 척도인 기대 종료시각을 보다 정확히 계산하도록 하였다. 3가지 정책들의 성능 분석을 위해서는 이산사건 시뮬레이션 기반의 실험을 수행하였다. 폐쇄루프 기반의 동적 정책은 의사결정이 필요한 시점마다 새로운 확률적 자원제약 스케줄링 문제를 생성하고, 이에 대한 해를 구하는 반면, 작업기반과 우선순위 기반의 정책은 프로젝트 초기에 생성한 해를 프로젝트 종료까지 이용하는 정적인 정책에 속한다. PSPLIB의 J120 문제를 대상으로 5가지 작업시간에 대한 확률분포로 실험한 결과는 동적인 정책이 정적인 정책에 비해 매우 우수하고 안정적인 성능을 보인다. 특히, 분산이 큰 확률분포에서는 성능의 차이가 더 커지는 경향을 보인다. 불확실성이 클수록 정적 정책의 성능은 더 떨어지나, 동적 정책은 비교적 안정적인 좋은 성능을 보인다. 현재 상황을 고려한 유연적인 정책이 보수적인 정책에 비해 불확실성에 더 잘 대처할 수 있음을 반영한다. 확률적 자원제약 문제의 해결을 위해서는 작업시간의 불확실성에 대한 정확한 추정이 필요하다. 동적인 정책 하에서는 새로운 정보가 입력되었을 때마다 기존의 추정이 수정되어야 하고, 이러한 수정은 프로젝트의 실행 중 지속적으로 이루어져야 한다. 향후 작업시간의 잘못된 추정에 대응할 수 있는 안정된 스케줄 생성에 대한 연구가 이루어져야 할 것으로 판단된다.

참고문헌

Ballestin, F. and Leus, R. (2009), Resource-Constrained Project Scheduling for Timely Project Completion with Stochastic Activity Durations, *Production and Operations Management*, **18**(4), 459-474.

Braysy, O. and Gendreau, M. (2005), Vehicle Routing Problem with Time Windows, Part I : Route Construction and Local Search Algorithms, *Transportation Science*, **39**(1), 104-118.

Bruni, M. E., Beraldi, P., Guerriero, F., and Pinto, E. (2011), A Heuristic Approach for Resource Constrained Project Scheduling with Uncertain Activity Durations, *Computers and Operations Research*, **38**(9), 1305-1318.

Fernandez, A. A., Armacost, R. L., and Pet-Edwards, J. J. (1998), Understanding Simulation Solutions to Resource Constrrained Project Scheduling Problems with Stochastic Task Duration, *Engineering Management Journal*, **10**(4), 5-13.

Fleszar, K. and Hindi, K. S. (2004), Solving the Resource-Constrained Project Scheduling Problem by a Variable Neighborhood Search, *European Journal of Operations Research*, **155**(2), 402-413.

Graham, R. L. (1966), Bounds for Certain Multiprocessing Anomalies, *Bell System Technical Journal*, **45**(9), 1563-1581.

Hansen, P., Mladenovic, N., and Perez, J. A. M. (2010), Variable Neighborhood Search : Methods and Applications, *Annals of Operations Research*, **175**(1), 367-407.

Kolisch, R. and Hartman, S. (2006), Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling : An Update, *European Journal of Operational Research*, **174**(1), 23-37.

Kolisch, R. and Sprecher, A. (1996), PSPLIB-A project scheduling library, *European Journal of Operational Research*, **96**(1), 205-216.

Li, H. and Womer, N. K. (2015), Solving Stochastic Resource-Constrained Project Scheduling Problems by Closed-Loop Approximate Dynamic Programming, *European Journal of Operational Research*, **246**(1), 20-33.

Li, K. Y. and Willis, R. J. (1992), An Iterative Scheduling Technique for Resource-Constrained Project Scheduling, *European Journal of Operational Research*, **56**(3), 370-379.

Li, S., Jia, Y., and Wang, J. (2012), A Discrete-Event Simulation Approach with Multiple-Comparison Procedure for Stochastic Resource-Constrained Project Scheduling, *International Journal of Advanced Manufacturing Technology*, **63**(1), 65-76.

Stork, F. (2001), Stochastic Resource-Constrained Project Scheduling, Ph. D. thesis, Technische Universitat Berlin.

Tsai, Y.-W. and Gemmil, D. D. (1998), Using Tabu Search to Schedule Activities of Stochastic Resource-Constrained Projects, *European Journal of Operational Research*, **111**(1), 129-141.

Tseng, L.-Y. and Chen, S.-C. (2006), A Hybrid Metaheuristic for the Resource-Constrained Project Scheduling Problem, *European Journal of Operational Research*, **175**(2), 707-721.

Valls, V., Ballestin, F., and Quintanilla, S. (2004), A Population-Based Approach to the Resource-Constrained Project Scheduling Problem, *Annals of Operations Research*, **131**(1), 305-324.

Valls, V., Ballestin, F., and Quintanilla, S. (2008), A Hybrid Genetic Algorithm for the Resource-Constrained Project Scheduling Problem, *European Journal of Operational Research*, **185**(2), 495-508.

Yim, D. S. (2011), Performance Analysis of Local Optimization Algorithms in Resource-Constrained Project Scheduling Problem, *Journal of the Korean Institute of Industrial Engineers*, **37**(4), 408-414.