Journal of the Korean Institute of Industrial Engineers
Vol. 37, No. 4, pp. 408-414, December 2011.

자원제약 프로젝트 스케쥴링 문제에 적용 가능한 부분 최적화 방법들의 성능 분석

임 동 순†

한남대학교 산업경영공학과

Performance Analysis of Local Optimization Algorithms in Resource-Constrained Project Scheduling Problem

Dong Soon Yim

Department of Industrial and Management Engineering, Hannam University, Daejeon, 309-791, Korea

The objective of this paper is to define local optimization algorithms (LOA) to solve Resource-Constrained Project Scheduling Problem (RCPSP) and analyze the performance of these algorithms. By representing solutions with activity list, three primitive LOAs, i.e. forward and backward improvement-based, exchange-based, and relocation-based LOAs are defined. Also, combined LOAs integrating two primitive LOAs are developed. From the experiments with standard test set J120 generated using ProGen, the FBI-based LOA demonstrates to be an efficient algorithm. Moreover, algorithms combined with FBI-based LOA and other LOA generate good solutions in general. Among the considered algorithms, the combined algorithm of FBI-based and exchange-based shows best performance in terms of solution quality and computation time.

Keywords: Resource-Constrained Project Scheduling, Local Optimization Algorithm, Heuristic

1. 서 론

자원제약의 프로젝트 스케쥴링 문제(RCPSP)는 다음과 같이 정의된다. 한 프로젝트는 n개의 활동과 2개의 가상활동을 포함하는 $n+2$개의 활동으로 구성되고, 활동의 수행에 특정의 자원이 요구된다. $J=\{0, 1, \cdots, n, n+1\}$는 활동의 집합을 의미하고, $K=\{1, \cdots, k\}$는 프로젝트 수행에 필요한 자원의 집합을 나타낸다. 활동 0과 $n+1$은 프로젝트의 시작과 종료를 의미하는 가상 활동이다. 활동들 간에, 그리고, 활동의 수행에 다음과 같은 제약사항이 존재한다.

- 활동 간에 선후행 조건이 있어 활동 j는 모든 선행 활동인 P_j들이 종료되어야 시작될 수 있다.
- 활동 j는 작업 시간 d_j에 걸쳐 매시간 마다 자원 $k \in K$에 대해 r_{jk}개의 자원이 필요하다
- 자원 k는 제한된 용량 R_k를 가지고 있다.

RCPSP는 이러한 제약 조건하에서 프로젝트의 총기간(make-span)을 최소화하는 스케쥴을 작성하는 문제이다(Brucker *et al*., 1999). 이 문제는 NP-hard 에 속하여 활동의 수가 증가할수록 해공간의 크기가 기하급수적으로 증가한다(Blazewicz *et al*., 1998). 따라서 많은 수의 활동을 포함하는 문제에서는 주어진 시간 내에 최적해를 구하기가 불가능하게 되어 휴리스틱 방법을 이용하여 근사 최적해를 구하는 것이 더 의미가 있게 된다(Kolisch and Hartman, 1999, 2006; Tomos and Lova, 2003).

휴리스틱 방법에 속하는 부분 최적화 알고리즘(Local Optimization Algorithm : LOA)은 현재해(current solution)에 대한 이웃해(neighborhood solution)를 탐색하는 이웃 탐색(neighborhood search)에 속한다(Ahuja *et al*., 2000). 이 알고리즘을 적용하기 위

이 논문은 2011년도 한남대학교 학술연구조성비 지원에 의하여 연구되었음.
† 연락저자 : 임동순 교수, 306-791 대전광역시 대덕구 오정동 133 한남대학교 산업경영공학과, Tel : 042-629-7399, Fax : 042-629-8004, E-mail : dsyim@hnu.kr
2011년 8월 22일 접수; 2011년 11월 10일 게재 확정.

하여는 현재해 x에 대한 이웃해 $N(x)$의 정의가 중요하다. 정의된 $N(x)$에 속하는 한 해인 x'의 비용이 현재해 x의 비용보다 낮다면 x'으로 x를 대체한다. 이 같은 대체 과정은 모든 이웃해의 비용이 현재해의 비용보다 크거나 같을 때까지 반복된다. 따라서 알고리즘이 종료될 때의 x는 고려되는 문제의 부분 최적해가 된다. 외판원 문제에서의 2-OPT나 클러스터링 문제에서의 K-Means 방법이 대표적인 부분 최적화 방법에 속한다.

많은 반복 횟수로 인한 시간 복잡성에도 불구하고, LOA는 해의 질을 향상 시키는 우수한 방법으로 인식 되어왔다. 특히, 메타 휴리스틱 방법과 LOA를 결합한 하이브리드 메타 휴리스틱(hybrid meta heuristic) 방법은 여러 문제에 대해 좋은 해를 생성한다고 알려져 있다. RCPSP 문제에서도 메타 휴리스틱의 해를 향상시키기 위하여 휴리스틱 방법을 결합한 하이브리드 메타 휴리스틱 방법들이 제안되었다. 유전자 알고리즘 또는 개미 알고리즘과 같은 메타 휴리스틱에 FBI(Forward Backward Improvement)(Tseng and Chen, 2006), double justification(Valls *et al*., 2005, 2008), 또는 HIA(Homogeneous Interval Algorithm)(Chen *et al*., 2010; Valls *et al*., 2004)를 결합한 방법들이 제안되었다.

그러나 FBI 나 double justification 는 부분 최적해를 구하는 방법이 아닌 단순한 향상적 휴리스틱에 속한다. HIA 방법에서는 Forward HIA와 Backward HIA를 더 이상의 향상이 없을 때까지 반복하여 부분 최적화에 속하나 많은 이웃해를 탐색하기보다는 하나의 이웃해를 탐색해 나가는 방법에 속한다. 저자들이 문헌조사로 파악하기로는 둘 이상의 이웃해를 탐색해 나가는 LOA를 메타 휴리스틱과 결합한 연구는 없는 듯하다. 이는 RCPSP 문제에 적용될 수 있는 LOA에 대하여 심도 있는 연구가 수행되지 않았던 것에 기인하는 것으로 파악된다. 이러한 필요성에 의하여 본 논문에서는 RCPSP 문제에 적용될 수 있는 LOA들을 정의하고, 이들의 성능을 분석한 결과를 논의한다. 본 논문의 제 2장에서는 RCPSP 문제의 해를 표현하는 방법인 활동리스트에 대해 소개하고, 제 3장에서는 활동 리스트 표현방법에 적용될 수 있는 부분 최적화 방법들을 설명한다. 제 4장에서는 PSPLIB의 문제들을 대상으로 한 실험을 통해 부분최적화 방법들의 성능을 분석한 결과를 논의한다.

2. 활동 리스트 표현 방법

RCPSP 문제의 해는 각 활동의 시작 시간을 나타내어야 한다. 그러나, 해가 각 활동의 시작 시간을 직접적으로 표현할 경우 다양한 LOA와 메타 휴리스틱 등의 적용이 어렵다. 본 연구에서는 활동들을 순서화된 리스트로 표현한 활동리스트(Activity list)(Kolisch and Hartman, 1999; Tomos and Lova, 2003)로 해를 표현하는 방법을 이용한다. 활동리스트는 해를 직접적으로 나타내고 있지 않으므로 별도의 디코딩(decoding) 방법을 이용하여 각 활동의 시작시간이 포함된 스케줄로 해석하도록 한다.

활동리스트로 표현된 해 $S=\{a_1, a_2, \cdots, a_n\}$는 1부터 n까지 번호를 매긴 n개 활동에 대한 순서화를 의미한다. 예를 들어, $S=\{2, 3, 4, 1\}$은 4개의 활동들에 대해 활동 2, 활동 3, 활동 4, 활동 1의 순서를 의미한다. 활동 리스트를 해로 디코딩하는 방법은 SGS(Schedule Generation Scheme)에 의한다. SGS는 거의 모든 RCPSP에 적용되는 휴리스틱의 가장 기본적인 핵심 역할을 한다. SGS는 현재까지 작성된 부분 스케줄에 삽입될 활동을 선택하고, 그 활동을 더하여 스케줄을 완성해 가는 방법으로 구성되어 있다. 이 같은 활동 선택과 부분 스케줄로의 활동 추가는 최종 스케줄을 완성할 때까지 반복된다. 본 연구에서는 두 가지 SGS인 순차적 SGS와 병렬 SGS 중 순차적 SGS를 이용하고, 부분 스케줄에 삽입될 활동의 선택은 활동 리스트의 순서 그 자체에 따르도록 한다. 삽입되는 활동의 시작 시각은 선후행 조건과 자원의 제약을 고려하여 가능한 가장 빠른 시점으로 하는 전진 스케줄링(forward scheduling)을 기본적인 디코딩 방법으로 한다(Kolisch and Hartman, 1999).

부분 스케줄에 삽입될 활동의 선택이 활동리스트의 순서에 따르기 때문에 활동리스트의 활동들에 대한 순서화가 잘못 이루어져 있다면 설명된 디코딩 방법에 의해 유효한 스케줄이 생성되지 못한다. 본 연구에서의 활동리스트는 선후행 활동 관계에 따라야 하며 다음 조건에 의해 리스트의 유효화를 쉽게 조사할 수 있다.

● 유효한 활동리스트 조건

활동리스트 $S=\{a_1, a_2, \cdots, a_n\}$에서 활동 $a_j (j=1, \cdots, n)$의 모든 선행 활동들은 $a_1, \cdots, a_{j-1}$ 중에 있어야 한다.

따라서 활동 a_j에 정의된 선행 활동이 m개 있다면 이들 선행활동이 $a_1, \cdots, a_{j-1}$중에 있는지를 조사하기 위해 $(j-1)\times m$번의 탐색횟수가 요구된다. 모든 활동의 평균 선행 활동수가 m이라면 한 리스트의 유효성을 조사하기 위해 평균 $m\times(n-1)\times(n-2)/2$번의 탐색횟수가 요구되어 $O(mn^2)$의 복잡성을 갖는다.

활동리스트에 적용되는 LOA들을 설명하기 전에 향상적 휴리스틱 중 매우 우수한 결과를 가져온다고 알려져 있는 FBI 방법을 우선 소개하기로 한다. 알고리즘은 다음과 같이 단순하다.

● 알고리즘 : FBI

단계 0 : 초기해 S_0를 생성한다.
단계 1 : S_0에 대해 전진 스케줄링을 수행한다.
단계 2 : 스케줄에서 활동들을 종료시점의 감소순으로 정렬하여 활동리스트 S_1을 생성한다.
단계 3 : S_1에 대해 후진 스케줄링(backward scheduling)을 수행한다.
단계 4 : 스케줄에서 활동들을 시작 시점의 증가순으로 정렬하여 활동리스트 S_2를 생성한다.
단계 5 : S_2에 대해 디코딩(전진 스케줄링)을 수행한다.

FBI는 주어진 활동리스트에 대해 전진 스케쥴링으로 해를 구하고, 다시 스케쥴의 활동들을 종료 시점 감소순으로 정렬하여 새로운 활동리스트를 생성한다. 이 활동리스트에 대해 각 활동의 종료시점이 가능한 가장 늦게 되고, 자원 제약이 만족되도록 후진 스케쥴링을 수행한다. 다시 이 스케쥴에서 활동의 시작 시점의 증가순으로 정렬한 새로운 활동리스트를 생성하고, 이 리스트에 대해 전진 스케쥴링을 재차 수행한다. 이 같은 절차는 스케줄의 활동들을 좌우로 흔드는 효과를 내어 활동들 간의 틈(여유시간)을 줄이는 역할을 한다. 시각적으로 볼 때 활동들이 빽빽하게 위치하도록 스케쥴을 조정하여 총기간을 감소시킨다.

3. 부분 최적화 알고리즘

활동리스트는 순열 표현으로 외판원 문제에서 해를 직접적으로 표현한 것과 동일하므로 외판원 문제에서의 LOA(즉, 이웃해를 생성하는 메커니즘)를 그대로 RCPSP에 적용할 수 있다. 그러나 설명한 바와 같이 활동리스트로 표현된 해는 모두 유효한 해가 아니므로 가급적 유효한 해를 이웃해로 생성하는 방법이 필요하다. 또한, 복잡성이 큰 방법보다는 단순한 방법으로 이웃해를 용이하게 생성할 수 있는 방법이 요구된다. 이러한 요구조건을 감안하여 본 연구에서는 다음의 3가지 기본 알고리즘을 고려하였다.

1) 교환(Exchange)

교환에서 주어진 활동리스트의 이웃해는 각 쌍의 두 활동에 대한 위치를 바꾸는 것이다.

● 알고리즘 : Ex

단계 0 : 초기해를 생성하여 이를 현재해 S로 한다.

단계 1 : 현재해 $S=\{a_1, a_2, \cdots, a_n\}$에서 모든 (i, j), $i<j$에 대해 a_i와 a_j의 위치를 바꾸어 새로운 해 S_{ij}를 구하고, 유효한 해라면 디코딩하여 총기간을 구한다.
S_{ij} 중 가장 적은 총기간을 갖는 S'의 총기간이 S의 총기간보다 작으면 $S=S'$으로 변경하고 단계 1을 반복한다. 아니면 정지한다.

2) 재위치(Relocate)

재위치에서 주어진 활동리스트의 이웃해는 각 활동에 대해 현 위치에서 다른 위치로 이동시킨 것이다.

● 알고리즘 : Re

단계 0 : 초기해 를 생성하여 이를 현재해 S로 한다.

단계 1 : 현재해 $S=\{a_1, a_2, \cdots, a_n\}$에서 모든 (i, j), $i \neq j$에 대해 a_i를 a_j 다음의 위치로 이동시켜 새로운 해 S_{ij}를 구하고, 유효한 해라면 디코딩하여 총기간을 구한다.
S_{ij} 중 가장 적은 총기간을 갖는 S'의 총기간이 S의 총기간보다 작으면 $S=S'$으로 변경하고 단계 1을 반복한다. 아니면 정지한다.

3) 반복적 FBI(FBIn)

FBI는 주어진 활동리스트에 대해 2번의 전진과 1번의 후진 스케쥴링을 수행한다. FBI를 초기해에 대한 하나의 이웃해를 생성하는 방법이라고 간주하여 이를 반복 수행하는 LOA 방법을 고려할 수 있다.

● 알고리즘 : FBIn

단계 0 : 초기해를 생성하여 이를 현재해 S로 한다.

단계 2 : 현재해 S에 대해 FBI를 적용하여 이웃해 S'을 구한다.

단계 3 : S'의 총기간이 S의 총기간 보다 작으면 $S=S'$로 변경하고, 단계 2로 간다. 아니면 정지한다.

알고리즘 Ex는 모든 두 활동에 대해 위치를 서로 교환한다. 따라서, 한번의 반복에 $n(n-1)/2$개의 이웃해를 조사한다. 물론, 모든 이웃해가 유효한 해는 아니다. 이 중 유효한 해에 대해 전진 스케쥴링을 수행한다. 알고리즘 Re는 한 활동을 현 위치에서 다른 위치로 재위치 시키는 방법이므로 한번의 반복에 n개 활동들을 고려하고, 각 활동은 $n-1$개의 이동 위치를 고려하므로 $n(n-1)$개의 이웃해를 조사한다. 반면 FBIn은 한 번의 반복에 하나의 이웃해 만을 고려한다. 따라서 반복 횟수에 따라 다르겠지만 일반적으로 FBIn이 가장 빠르고, Ex, Re 순으로 실행시간이 길어지는 시간 복잡성을 가진다. 이는 다음 장에 설명할 실험 분석에서도 동일한 결과를 가져온다.

Ex와 Re는 하나의 해에 대한 모든 이웃해를 탐색한 후 이 중 가장 좋은 이웃해를 현재해로 치환하기 때문에 한 번의 반복에 적지 않은 시간이 소요된다. 본 논문에서는 모든 이웃해를 탐색하는 특성으로 이러한 LOA들을 각각 Explicit 교환 과 Explicit 재위치라고 명명한다. 실행시간을 줄이기 위한 방안으로는 한번의 반복에 모든 이웃해를 탐색하기보다 탐색 도중의 한 이웃해가 현재해 보다 좋으면 그 이웃해를 현재해로 하고, 다시 현재해에 대한 이웃해 탐색을 수행하는 방법이 있다. 이 경우 이웃해 탐색 순서의 결정이 알고리즘의 성능에 중요한 역할을 한다. 본 연구에서는 Ex의 경우 현재해 $S=\{a_1, a_2, \cdots, a_n\}$에 대해 $i=1, 2, \cdots, n-1$의 반복 안에 $j=i+1, i+2, \cdots, n$의 반복을 실행하도록 하여 (a_i, a_j)의 교환에 의한 이웃해를 탐색하도록 하였다. Re는 $i=1, 2, \cdots, n$의 반복 안에 $j=1, 2, \cdots, n$의 반복을 실행하도록 하여 a_i를 a_j다음의 위치로 변경한 이웃해를 탐색하도록 하였다. 본 논문에서는 전체 이웃해 보다는 이웃해 일부를 탐색하는 특성으로 이러한 알고리즘들을 각각 Partial 교환과 Partial 재위치로 명명하기로 한다.

위에서 설명한 3가지 LOA들을 결합하여 새로운 LOA를 구성할 수 있다. 두 알고리즘의 결합으로 인해 실행시간이 길어지는 단점이 있지만 보다 많은 해공간을 탐색하여 우수한 해

를 생성할 가능성이 커진다.

1) FBIn 과 교환의 결합

한 번의 반복은 FBIn을 수행한 후의 해에 Ex를 수행하는 절차로 구성된다.

• 알고리즘 : FBIn+Ex

단계 0 : 초기해를 생성하여 이를 현재해 S로 한다.
단계 1 : S에 FBIn을 적용하여 새로운 해 S' 을 생성한다.
단계 2: S' 에 Ex를 적용하여 새로운 해 S'' 을 생성한다.
단계 3 : $S = S' = S''$ 이면 정지하고, 아니면 $S = S''$ 으로 변경하여 단계 1 로 간다.

2) FBIn 과 재위치 결합

한번의 반복은 FBIn을 수행한 후의 해에 Re를 수행하는 절차로 구성된다.

• 알고리즘 : FBIn+Re

단계 0 : 초기해를 생성하여 이를 현재해 S 로 한다.
단계 1 : S에 FBIn을 적용하여 새로운 해 S' 을 생성한다.
단계 2 : S' 에 Re를 적용하여 새로운 해 S'' 을 생성한다.
단계 3 : $S = S' = S''$ 이면 정지하고, 아니면 $S = S''$ 로 변경하여 단계 1 로 간다.

3) 교환 과 재위치 결합

한 번의 반복은 Ex를 수행한 후의 해에 Re를 수행하는 절차로 구성된다.

• 알고리즘 : Ex+Re

단계 0 : 초기해를 생성하여 이를 현재해 S로 한다.
단계 1 : S에 Ex를 적용하여 새로운 해 S' 을 생성한다.
단계 2 : S' 에 Re를 적용하여 새로운 해 S'' 을 생성한다.
단계 3 : $S = S' = S''$ 이면 정지하고, 아니면 $S = S''$ 로 변경하여 단계 1로 간다.

4. 실험 및 분석

본 연구에서 제안된 부분 최적화 방법의 성능을 분석하기 위해 Kolish and Sprecher(1996)에 의해 고안된 표준문제인 PSPLIB의 J120 문제들을 고려하였다. J120 문제는 120개의 활동을 포함하는 문제로 60개의 카테고리(X1~X60)로 구성되어 있고, 각 카테고리에 10개씩의 인스턴스들이 있어 총 600개의 문제로 구성되어 있다. 각 알고리즘을 각 문제 인스턴스에 적용하는데 있어 100개의 초기해를 랜덤으로 생성하였다. 랜덤으로 생성했으므로 유효하지 않은 해가 발생할 수 있다. 유효하지 않은 해는 별도의 알고리즘을 이용하여 유효한 해로 변형시켰다. 임의의 해 S_0를 입력으로 하여 유효한 해 S를 생성하는 방법은 다음과 같다.

• 알고리즘 : 유효한 활동리스트 생성

입력 : 랜덤으로 생성된 초기해 $S_0 = \{a_1, a_2, \cdots, a_n\}$
단계 0 : $S = \varnothing$, $i = 1$,
단계 1 : $|S| = n$이면 정지한다.
단계 2 : $a_i \in S$ 라면 $i = i+1$로 하고, 단계 2를 반복한다.
단계 3 : a_i의 모든 선행 활동들이 S 에 있는지 조사한다.
선행활동이 있다면 $S = S + a_i$, $i = 1$로 하고, 단계 1로 간다.
선행활동이 없다면 $i = i+1$로 하고, 단계 2로 간다.

실험 대상이 된 알고리즘은 모두 12가지로 <Table 1>과 같다. FBI는 부분 최적화 알고리즘에 속하지는 않지만 알고리즘들의 성능비교를 위하여 포함되었다. 알고리즘들은 JAVA 프로그래밍 언어로 구현되어 Intel 2.33 GHz CPU와 2 GB의 메모리를 갖춘 PC에서 실행되었다.

Table 1. Local optimization algorithms

no	algorithm	description
1	FBI	FBI
2	FBIn	Repetitive FBI
3	Ex	Explicit Exchange
4	Re	Explicit Relocate
5	FBIn+Ex	Combination of FBIn and explicit Exchange
6	FBIn+Re	Combination of FBIn and explicit Relocate
7	Ex+Re	Combination of explicit Exchange and explicit Relocate
8	Ex'	Partial Exchange
9	Re'	Partial Relocate
10	FBIn+Ex'	Combination of FBIn and partial Exchange
11	FBIn++Re'	Combination of FBIn and partial Relocate
12	Ex'+Re'	Combination of partial Exchange and partial Relocate

X1_1 문제에 대한 실험 결과는 <Table 2>와 같다. 100개 초기해에 대해 각 알고리즘을 적용한 결과로 생성된 100개의 해 중 가장 좋은 해의 총기간인 최소값과 총기간의 향상율(초기해에 비해 향상된 율) 평균과 표준편차, 그리고 100번의 실행에 소요된 실행시간을 나타내었다.

LOA 중 FBIn은 적은 실행시간으로 많은 향상을 가져와 가장 효율이 큰 알고리즘임을 나타낸다. 초기해에 비해 20.09% 정도의 총기간 단축을 가져왔으며 소요시간은 0.094초로 적은 시간이 소요되었다. 가장 큰 향상을 가져온 알고리즘은 23.41%의 FBIn+Re이고, 그 뒤를 FBIn+Re', FBIn+Ex', FBIn+Ex가 따

르고 있다. FBIn을 기준으로 보면, Ex, Re, Ex+Re, Ex', Re', Ex'+Re'들은 저조한 향상율과 많은 실행시간으로 성능이 떨어진다고 볼 수 있다. FBIn에 다른 부분 최적화 알고리즘을 결합한 FBIn+Ex, FBIn+Re, FBIn+Ex', FBIn+Re'은 이들에 비해 해의 질이 좋아 FBIn의 성능이 매우 효과적임을 알 수 있다.

일반적으로 Explicit 탐색에 기반한 Ex, Re들이 Partial 탐색에 기반한 방법보다 많은 실행시간이 소요된 것은 자명한 결과이다. Re'의 경우 101초가 걸렸지만, Re의 경우 152초로 더 많은 시간이 소요되었다. 그러나 많은 실행시간에 비해 Explicit 방법들은 Partial 방법 보다 그다지 큰 향상을 가져오지는 않았다. 또한, Ex'에 비해 Re'가 7배 이상의 시간이 소요되어 Ex'가 보다 효율적임을 나타낸다. 결론적으로 FBIn+Ex'가 실행시간이나 해의 질에서 상대적으로 만족할 만한 결과를 가져왔다고 볼 수 있다. 100개의 해에 대한 실행시간이 35초로 크지 않고, 최소 총기간도 108로 대체로 만족할 만하였다.

Table 2. Experiment result in X1_1 instance

algorithm	best	improvement ratio		CPU time(sec)
		ave(%)	std	
FBI	117	18.43	7.10	0.031
FBIn	114	20.09	7.27	0.094
Ex	119	13.36	7.05	24.328
Re	119	13.97	7.27	151.719
FBIn+Ex	107	23.15	6.94	52.063
FBIn+Re	110	23.41	7.02	196.218
Ex+Re	119	14.94	7.24	141.032
Ex'	117	12.94	7.40	14.234
Re'	119	13.63	7.71	101.094
FBIn+Ex'	108	23.18	7.01	34.594
FBIn+Re'	110	23.29	6.92	124.812
Ex'+Re'	117	14.29	7.63	118.547

이 같은 결과는 X1_1 문제에서 뿐 만 아니라 다른 문제에서도 유사하다. X21에서 X40까지 20개 문제 카테고리에서 각 카테고리에 속한 10개 문제 모두를 대상으로 실험한 결과인 <Figure 1>, <Figure 2> 역시 비슷한 결론을 도출할 수 있다. <Figure 1>에서 x축은 문제 카테고리를 나타내어 각 카테고리에서 10개 문제에 대한 평균 향상율을 그래프로 나타낸 것이다. 그림은 FBIn에 다른 부분 최적화 알고리즘을 결합한 FBIn+Ex, FBIn+Re, FBIn+Ex', FBIn+Re'의 성능이 비슷하고, 이들 알고리듬들이 다른 알고리즘들에 비해 우수한 해를 생성함을 나타낸다. FBI와 FBIn은 X36 문제에서 저조한 향상율을 보여 다른 LOA에 비해 안정성이 떨어지는 것을 알 수 있다. FBIn을 제외한 나머지 LOA들의 효율을 나타내는 <Figure 2>에 의하면 Re 또는 Re'을 포함하는 알고리즘에 비해 Ex 또는 Ex'을 포함하는 알고리즘들이 더 효율적임을 알 수 있다. 또한, Explicit 교환인 Ex'보다는 partial 교환인 Ex가 더 효율적이어 상대적으로 적은 시간 내에 우수한 해를 생성함을 나타낸다.

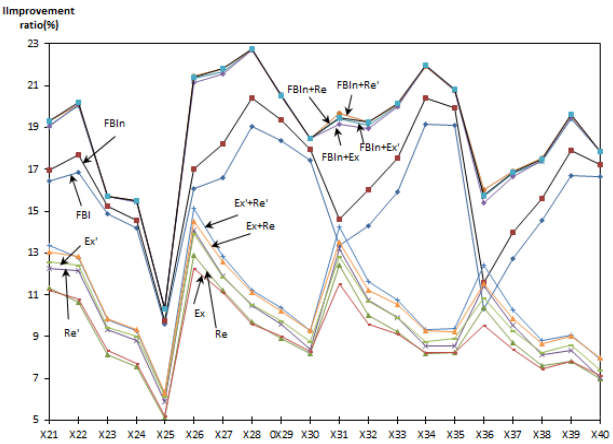

Figure 1. Average improvement ratio of LOAs in X21-X40 problems

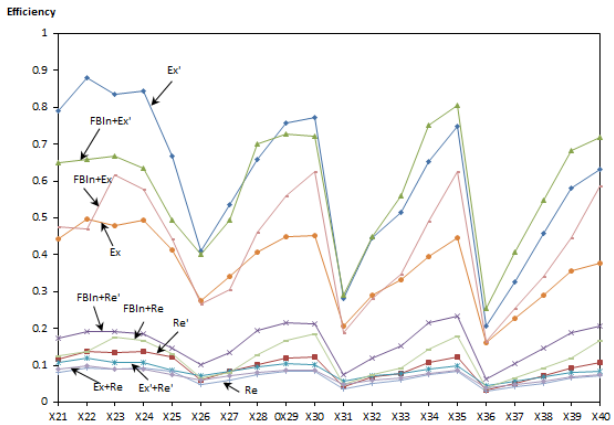

Figure 2. Average Efficiency of LOAs in X21-X40 problems

● **초기해와 부분 최적화 해의 관계**

일반적으로 초기해가 우수하여야 LOA를 적용한 해의 질이 좋을 것으로 예상된다. 그것이 사실이라면 좋은 해를 생성하기 위하여는 소수의 우수한 초기해를 선별하고, 이들에 대해 LOA를 적용하는 것이 모든 초기해에 적용하는 것보다 더 효과적이라는 논리를 제공할 것이다.

<Figure 3>은 X1_1 문제에서 100개의 초기해에 대해 FBIn과 partial LOA들을 적용한 결과인 부분 최적해의 관계를 나타낸다. 그림에서 선형 회귀선은 모두 양의 기울기를 가져 초기해가 좋을수록 부분 최적해가 좋다는 결과를 나타낸다. 회귀선의 기울기가 0이라는 귀무가설을 검정하기 위한 <Table 3>의 ANOVA 표에 의하면 FBIn의 F 값이 3,5751로 1%의 유의수준에서 기울기가 0이라는 가설이 채택되나, 나머지 partial LOA에서는 F 값이 F(1,98 ; 0.01) = 6.9343보다 커 기울기가 0이 아니라는 결론이 채택된다.

그러나 선형회귀선의 R^2이 0.03에서 0.18의 범위에 있어 모두 확실한 선형관계가 있다고 보기 어렵다. 또한, 대부분의 기울기 값이 적어 초기해가 LOA의 해에 큰 영향을 미치지 못한다고 볼 수 있다. 따라서 우수한 초기해를 찾기 위해 노력하기보다는 적정 개수의 초기해를 가지고 알고리즘을 적용하는 것이 바람직하다는 것을 알 수 있다. 특히, FBIn의 경우 기울기도 작고, 산포도 작아서 초기해에 가장 강건한 결과를 보였다.

5. 결 론

NP-hard 문제에서 부분 최적화 알고리즘은 메타 휴리스틱과 결합되어 매우 우수한 해를 가져온다고 알려져 있다. RCPSP에서도 동일한 결과를 예측할 수 있다. 본 연구에서는 RCPSP에 적용될 수 있는 부분 최적화 알고리즘을 정의하였고, 실험을 통해 이들의 성능을 분석하였다. 정의된 부분 최적화 알고리즘들은

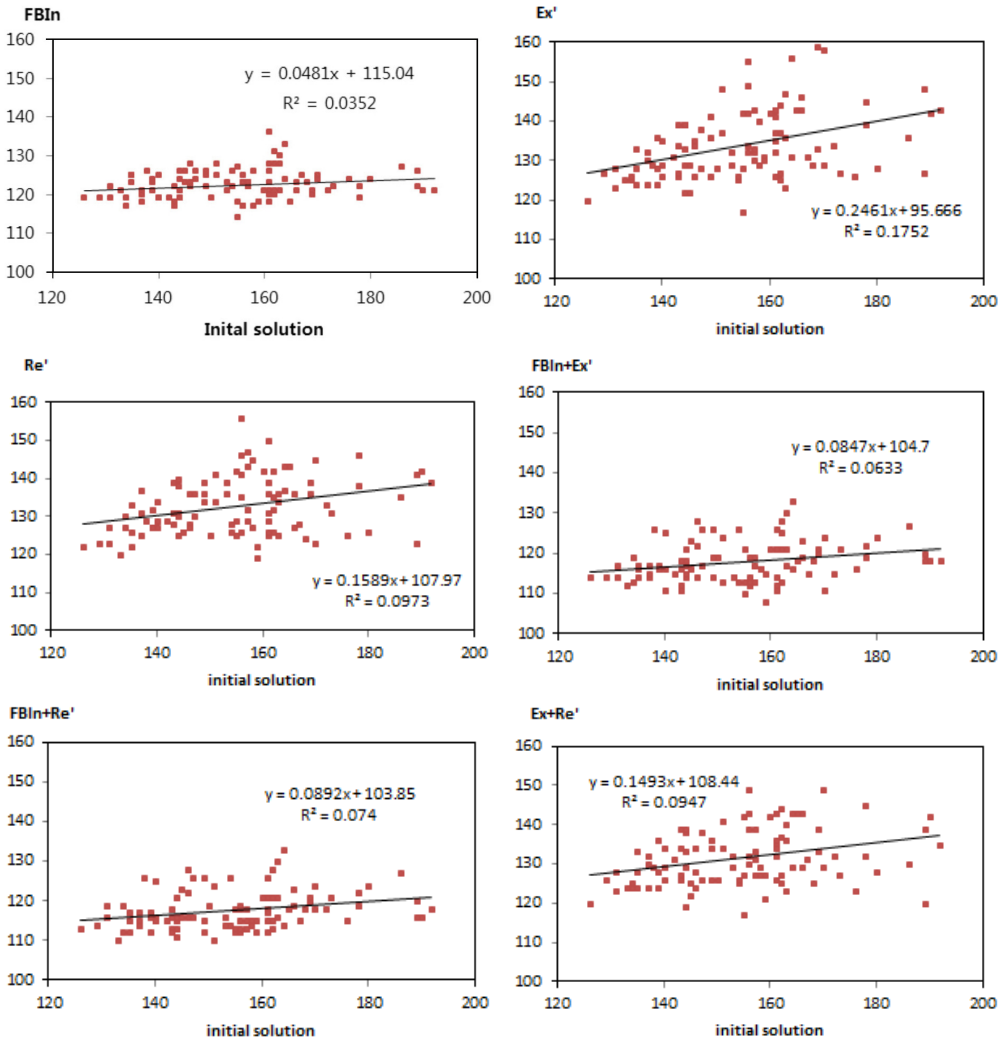

Figure 3. Relationship between initial solutions and LOA solutions

Table 3. Linear Regression ANOVA

algorithm	mean square of regression line	mean square of residual	F_T	F table
FBIn	48.6035	13.5950	3.5751	F(1, 98; 0.1) = 2.7647 F(1, 98; 0.05) = 3.9493 F(1, 98; 0.025) = 5.2013 F(1, 98; 0.01) = 6.9343
Ex'	1272.5056	48.1263	26.4410	
Re'	530.6736	44.8328	11.8367	
FBIn+Ex'	150.6422	21.2232	7.0980	
FBIn+Re'	166.9822	19.6081	8.5160	
Ex'+Re'	468.1087	40.8855	11.4492	

활동리스트로 해를 표현하고, 전진 스케쥴링에 기반한 디코딩을 통해 유효한 스케쥴을 생성하는 방법에 기초하였다. FBI를 더 이상의 향상이 없을 때까지 반복하는 FBIn, 활동리스트의 두 활동을 서로 교환하는 방법인 Ex, 그리고, 활동리스트에서 한 활동의 위치를 다른 곳으로 재위치시키는 Re의 3가지 방법을 기본으로 하여 이들을 결합한 FBIn+Ex, FBIn+Re, 그리고, Ex+Re를 정의하였다. 또한, 알고리즘에서 향상된 해를 탐색하는 방법에 따라 Partial 과 Explicit의 두 가지 부분 최적화 알고리즘을 정의하였다. PSPLIB의 J120문제 집합에 대해 실험한 결과 FBIn과 Ex, Re를 결합한 FBIn+Ex, FBIn+Re 방법이 우수한 해를 생성하였다. 특히, FBIn과 partial Ex를 결합한 FBIn+Ex'가 매우 효율적이어서 비교적 적은 시간 동안에 우수한 해를 생성하였다.

논문에서 정의된 LOA들은 초기해가 우수할수록 보다 좋은 해를 생성할 가능성이 커진다. 따라서 메타 휴리스틱과 LOA를 결합한 하이브리드 메타 휴리스틱을 적용할 때 메타 휴리스틱의 우수한 해를 선별하여 이들에 대해 LOA를 적용하는 것이 보다 효과적일 것이다. 앞으로 이러한 하이브리드 메타 휴리스틱의 성능에 대한 연구가 계속되어야 할 것이다.

참고문헌

Ahuja, R., Orlin, J., and Sharma, D. (2000), Very Large-Scale Neighborhood Search, *International Transactions in Operations Research*, **7**, 301-317.

Blazewicz, J., Lenstra, L., and Kan R. (1998), Scheduling Subject to Resource Constraints : Classification and Complexity, *Discrete Mathematics*, **5**, 11-24.

Brucker, P., Drexl, A., Mohring, R., Neumann, K., and Presch, E. (1999), Resource-Constrained Project Scheduling : Notation, Classification, Models, and Methods, *European Journal of Operational Research*, **112**, 3-41.

Chen, W., Shi, Y., Teng, H., Lan, X., and Hu, L. (2010), An Efficient Hybrid Algorithm for Resource-Constrained Project Scheduling, *Information Sciences*, **180**, 1031-1039.

Kolisch, R. and Hartman, S. (1999), *Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem* : *Classification and Computational Analysis* in Handbook on Recent Advances in Project Scheduling, Kluwer; Weglarz, J(Ed), 147-178.

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

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

Tormos, P. and Lova, A. (2003), An Efficient Multi-Pass Heuristic for Project Scheduling with Constrained Resources, *International Journal of Production Research*, **41**(5), 1071-1086.

Tseng, L. and Chen, S. (2006), A Hybrid Metaheuristic for the Resource-Constrained Project Scheduling Problem, *European Journal of Operational Research*, **175**, 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**, 305-324.

Valls, V., Ballestin, F. and Quintanilla, S. (2005), Justification and RCPSP : A Technique that Pays, *European Journal of Operational Research*, **165**, 375-386.

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**, 495-508.