Journal of The Korea Society of Computer and Information
Vol. 17 No. 6 , June 2012

2012-17-6-1-2

다차원 배낭 문제를 위한 정수계획법 기반 지역 탐색 기법

황 준 하*

Integer Programming-based Local Search Techniques for the Multidimensional Knapsack Problem

Junha Hwang*

요 약

정수계획법 기반 지역 탐색은 단순 언덕오르기 탐색을 기반으로 하는 지역 탐색의 일종으로서 기존의 지역 탐색과는 달리 이웃해 생성 시 정수계획법을 활용한다. 기존 연구 [1]에 의하면 정수계획법 기반 지역 탐색은 경영과학 및 인공지능 분야에서 많은 관심을 받아 온 다차원 배낭 문제를 해결하는 데 매우 효과적인 것으로 알려져 있다. 그러나 해당 연구에서는 OR-Library에 있는 다차원 배낭 문제들 중 규모가 가장 큰 문제들만을 대상으로 하여 정수계획법 기반 지역 탐색의 우수성을 검증하였다는 단점이 있다. 본 논문에서는 그 외의 문제들을 대상으로 정수계획법 기반 지역 탐색을 적용함으로써 보다 객관적으로 정수계획법 기반 지역 탐색의 우수성을 검증한다. 아울러 본 논문에서는 기존의 정수계획법 기반 지역 탐색이 단순 언덕오르기 탐색과 정수계획법을 결합한 것과는 달리 언덕오르기 탐색, 타부 탐색, 시뮬레이티드 어닐링과 같은 다른 지역 탐색 기법과 정수계획법을 결합하는 방안을 제시한다. 실험 결과, 정수계획법 기반 지역 탐색은 중소 규모의 다차원 배낭 문제들에 있어서도 기존의 가장 좋은 휴리스틱 탐색 기법에 비해 유사하거나 더 우수한 성능을 발휘함을 확인하였다.

▸ Keyword : 정수계획법 기반 지역 탐색, 다차원 배낭 문제, 정수계획법, 지역 탐색

Abstract

Integer programming-based local search(IPbLS) is a kind of local search based on simple hill-climbing search and adopts integer programming for neighbor generation unlike general local search. According to an existing research [1], IPbLS is known as an effective method for the multidimensional knapsack problem(MKP) which has received wide attention in operations research and artificial intelligence area. However, the existing research has a shortcoming that it verified the superiority of IPbLS targeting only largest-scale problems among MKP test problems in the OR-Library. In this paper, I verify the superiority of IPbLS more objectively by applying it to

• 제1저자 : 황준하 • 교신저자 : 황준하
• 투고일 : 2012. 02. 28, 심사일 : 2012. 04. 18, 게재확정일 : 2012. 05. 21.
* 금오공과대학교 컴퓨터공학과(Dept. of Computer Engineering, Kumoh National Institute of Technology)
※ 본 연구는 금오공과대학교 학술연구비에 의하여 연구된 논문임

other problems. In addition, unlike the existing IPbLS that combines simple hill-climbing search and integer programming, I propose methods combining other local search algorithms like hill-climbing search, tabu search, simulated annealing with integer programming. Through the experimental results, I confirmed that IPbLS shows comparable or better performance than the best known heuristic search also for mid or small-scale MKP test problems.

▸ Keyword : Integer Programming-based Local Search, Multidimensional Knapsack Problem, Integer Programming, Local Search

I. 서 론

본 논문의 목적은 두 가지이다. 첫 번째는 중소 규모의 다차원 배낭 문제(multidimensional knapsack problem)에 있어서 정수계획법 기반 지역 탐색(integer programming-based local search)의 성능을 검증하는 것이고, 두 번째는 조합 최적화 문제를 해결하기 위한 정수계획법 기반 지역 탐색의 다양한 적용 및 구현 방안을 제시하는 것이다.

정수계획법 기반 지역 탐색은 지역 탐색을 기본으로 하는 탐색 알고리즘으로서 기존의 지역 탐색과 마찬가지로 현재해를 기반으로 임의의 변수들의 값을 변경하여 이웃해를 생성한 후 그 결과에 따라 이동하는 과정을 반복적으로 수행한다. 그러나 일반적인 지역 탐색이 임의의 변수들의 값을 무작위로 변경하여 이웃해를 생성하는 것과는 달리 정수계획법 기반 지역 탐색에서는 정수계획법을 사용하여 이웃해를 생성한다. 기존 연구 [1]에서는 다차원 배낭 문제를 해결하기 위하여 정수계획법 기반 지역 탐색을 사용하였으며, 실험을 통해 지금까지 가장 좋은 결과를 보였던 탐색 기법보다 더 좋은 해를 도출할 수 있음을 확인하였다. 그러나 해당 연구에서는 OR-Library에서 제공하는 270개의 다차원 배낭 문제들 중 문제 규모가 가장 큰 30개의 문제들만을 대상으로 하였다. 따라서 본 논문에서는 그 외의 문제들을 대상으로 정수계획법 기반 지역 탐색을 적용함으로써 정수계획법 기반 지역 탐색의 전반적인 성능을 다시 한번 검증하고자 한다.

정수계획법 기반 지역 탐색은 비교적 최근에 제안된 메타휴리스틱 탐색의 일종으로서 지역 탐색 기법들 중 단순 언덕오르기 탐색(simple hill-climbing search)의 틀을 사용하고 있다. 즉, 현재해를 대상으로 몇 개의 변수들을 선택한 다음 선택된 변수들의 값을 변경하여 이웃해를 생성한 후 현재해보다 좋으면 이동하되 기존의 단순 언덕오르기 탐색과는 달리 이웃해 생성 시 정수계획법을 사용한다. 그러나 대상 문제 또는 데이터에 따라서는 단순 언덕오르기 탐색 외에 언덕오르기 탐색(hill-climbing search), 타부 탐색(tabu search), 시뮬레이티드 어닐링(simulated annealing)과 같은 지역 탐색 기법과 정수계획법을 결합한 알고리즘이 더 적합할 수도 있다. 따라서 본 논문에서는 기존의 단순 언덕오르기 탐색과 정수계획법을 결합한 방법 외에 다양한 지역 탐색 기법들과 정수계획법을 결합하는 방안 및 구현 방안을 제시하고, 다차원 배낭 문제를 대상으로 한 실험을 통해 일반적인 조합 최적화 문제에 대한 효용성을 검토한다.

실험 결과, 정수계획법 기반 지역 탐색은 중소 규모의 다차원 배낭 문제에 있어서도 기존의 가장 좋은 결과를 보였던 탐색 기법들과 유사하거나 더 좋은 결과를 도출할 수 있음을 확인할 수 있었다. 또한 비록 다차원 배낭 문제에 있어서 기존의 정수계획법 기반 지역 탐색이 본 논문에서 제시하는 또 다른 형태의 정수계획법 기반 지역 탐색들보다 더 좋은 성능을 발휘하였으나, 본 논문에서 제시한 정수계획법 기반 지역 탐색들 역시 메타휴리스틱 탐색으로서 효용 가능성이 충분히 있음을 확인할 수 있었다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 정수계획법 기반 지역 탐색과 다차원 배낭 문제에 대한 관련 연구를 검토하며, Ⅲ장에서는 다양한 형태의 정수계획법 기반 지역 탐색 알고리즘과 이에 대한 구현 방안을 제시한다. Ⅳ장에서는 Ⅲ장에서 제시한 정수계획법 기반 지역 탐색 알고리즘들을 다차원 배낭 문제에 적용한 실험 결과를 제시하고 기존 연구들과의 비교를 통해 정수계획법 기반 지역 탐색의 우수성을 검증한다. 마지막으로 Ⅴ장에서는 결론 및 향후 과제에 대해 설명한다.

II. 관련 연구

1. 정수계획법 기반 지역 탐색

기존 연구 [2]에서 제시한 일반적인 정수계획법 기반 지역 탐색의 구조는 그림 1과 같으며, 전체적인 구조는 지역 탐색 기법들 중 단순 언덕오르기 탐색과 유사하다. 일반적으로 조

합 최적화 문제에서 하나의 해는 일련의 변수들(x)로 구성되며 각 변수들은 몇 개의 값들 중 하나를 가지게 된다. 여기서 주어진 목적함수(Obj)를 최대화 또는 최소화할 수 있는 각 변수들의 값을 결정하는 것이 목표이다. 정수계획법 기반 지역 탐색에서는 대부분의 지역 탐색 기법들과 마찬가지로 먼저 휴리스틱 탐색 기법을 사용하여 초기해를 생성한 후 이를 현재해로 설정한다. 그리고 모든 변수들 중 k개의 변수를 선택하고 해당 변수들의 값을 변경하는 과정을 최대 탐색 시간 등 종료 조건을 만족할 때까지 반복적으로 수행한다. 이때 기존의 단순 언덕오르기 탐색의 경우 선택된 k개 변수들의 값을 무작위로 변경하는 것과는 달리 정수계획법 기반 지역 탐색에서는 k개의 변수들이 취할 수 있는 모든 값들 중 가장 좋은 값, 즉 최적해로 설정하게 되며 이때 최적해를 도출하는 데 사용되는 방법이 바로 정수계획법이다.

```
Algorithm IPbLS
    x : Variable vector (Current solution).
    Obj : Objective function.
    k : The number of selected variables to be changed.
    IP : An integer programming solver.
Begin
    x = Make an initial solution using a heuristic method
    While stopping condition is not met Do
        Select k variables from x
        Add Obj and all constraints to IP
            • Fix values of unselected variables from x
        x = Make a neighbor solution with IP
    End While
    return x
End Begin
```

그림 1. 일반적인 정수계획법 기반 지역 탐색
Fig. 1. General Integer Programming-based Local Search

정수계획법의 기본 탐색 알고리즘인 분지한계법(branch and bound)은 탐색 공간의 해들을 완전히 열거하는 방식을 사용하기 때문에 최적해의 도출이 보장된다. 더군다나 단순한 완전 열거 방식과는 달리 상한값과 하한값을 활용하여 최적해를 도출하는 데 불필요한 부분을 효과적으로 제거함으로써 중소 규모의 문제에 있어서 매우 빠르게 최적해를 도출할 수 있다는 장점이 있다[3]. 그러나 문제의 규모가 커질 경우 여전히 최적해를 도출하기까지 많은 시간을 요구하며, 특히 탐색 정보를 저장하기 위한 방대한 탐색 트리를 유지하는 데 과도한 메모리가 소요된다는 단점이 있다. 또한 정수계획법은 내부적으로 상한값을 구하기 위해 선형계획법(linear programming)을 사용하고 있는데, 선형계획법을 적용하기 위해서는 주어진 문제가 선형적으로 표현되어야만 한다. 즉, 주어진 문제의 목적함수와 제약조건이 모두 변수들에 대한 1차식으로 표현될 수 있어야만 한다는 것이다. 이와 같은 문제를 흔히 정수계획법 문제라 한다. 그러나 다행히도 배낭 문제(knapsack problem)를 비롯하여 집합 커버링 문제(set covering problem)[4], 위치 할당 문제(location allocation problem)[5], 순회 외판원 문제(traveling salesman problem)[6] 등 산업계의 주목을 받아 온 많은 문제들이 정수계획법 문제로 표현된다. 뿐만 아니라 그 외의 조합최적화 문제들에 대해서도 정수계획법 모델을 찾기 위한 노력이 계속되고 있다. 따라서 정수계획법 문제를 해결하기 위한 효과적인 탐색 기법을 개발하는 것은 매우 중요한 일이다.

정수계획법 문제를 효과적으로 해결하기 위해 지역 탐색과 정수계획법을 결합한 기존 연구로는 [7], [8] 등이 있다. 참고문헌 [7]에서는 승무일정계획 문제를 해결하기 위해 타부 탐색을 기본으로 사용하되 탐색의 다각화를 위해 정수계획법을 활용하였으며, [8]에서는 간호사 일정계획 문제에 있어서 정수계획법 문제로 표현되는 일부 문제를 풀기 위해 정수계획법을 활용하였다. 그러나 두 연구 모두 정수계획법과 지역 탐색을 결합했다는 점이 정수계획법 기반 지역 탐색과 유사하지만 구체적인 방법에 있어서는 그림 1과 다소 거리가 있다. 그림 1의 정수계획법 기반 지역 탐색의 개념을 그대로 적용한 연구로는 참고문헌 [1], [2], [9], [10]이 있으며, 각각 다차원 배낭 문제, 최대 커버링 문제, 네트워크 디자인 문제, N-Queens 최대화 문제를 해결하기 위해 정수계획법 기반 지역 탐색을 활용하였다. 해당 연구들 모두 기본적으로 그림 1과 같이 단순 언덕오르기 탐색에 기반한 정수계획법 기반 지역 탐색을 사용하고 있으며, 해당 문제의 특징을 활용하여 정수계획법 기반 지역 탐색의 효과를 극대화하고자 노력하였다. 본 논문에서는 다차원 배낭 문제를 대상으로 하되 기존 알고리즘을 확장하여 보다 다양한 방식으로 지역 탐색과 정수계획법을 결합하는 방안을 제시한다.

2. 다차원 배낭 문제

다차원 배낭 문제는 지금까지 경영과학 및 인공지능 분야에서 많은 관심을 받아온 NP-hard 조합 최적화 문제의 일종으로서 식 (1)~(3)과 같이 선형적으로 표현되는 정수계획법 문제이다[11]. 다차원 배낭 문제에서는 값($p_j > 0$)을 가진 n개의 아이템이 주어지며 m개의 자원에 대한 한계치($c_j > 0$)가 주어진다. 또한 각 아이템은 각 자원 i에 대해 일정량($w_{ij} \geq 0$)을 요구하게 된다. 아이템 j의 선택여부는 0 또는 1의 값을 가진 결정변수 x_j로 표현되며 1인 경우 해당 아이템이 선택되었음을 의미한다. 다차원 배낭 문제의 목적함수는

선택된 아이템들의 값의 합을 최대화하는 것이며 이는 식 (1)에 의해 표현된다. 단, 선택된 아이템들에 의한 자원 요구량이 각 자원 i에 있어서 한계치 c_i를 넘지 않아야만 하는데 이는 식 (2)에 의해 표현된다.

다차원 배낭 문제에 대한 가장 중요한 기존 연구로는 참고문헌 [12]를 꼽을 수 있다. 이 연구의 중요성은 유전 알고리즘을 사용하여 다차원 배낭 문제를 해결한 해법 그 자체뿐만 아니라 다양한 테스트 문제를 만들어 사용하고 공개함으로써 이후의 연구들이 이를 대상으로 객관적인 검증을 할 수 있게 되었다는 데 있다. 참고문헌 [12]에서 만든 다차원 배낭 문제의 테스트 문제들은 OR-Library 사이트에 있는데, OR-Library에는 다차원 배낭 문제뿐만 아니라 경영과학 분야에서 관심을 갖고 있는 다양한 조합 최적화 문제에 대한 테스트 문제들이 포함되어 있다[13].

$$\text{Maximize} \quad z = \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\text{Subject to} \quad \sum_{j=1}^{n} w_{ij} x_j \le c_i, \; i = 1...m, \tag{2}$$

$$x_j \in \{0,1\}, \; j = 1,...,n. \tag{3}$$

참고문헌 [12]에서는 다음과 같은 방식에 의해 다차원 배낭 문제를 위한 테스트 문제를 생성하였다. 다양한 규모의 문제를 생성하기 위해 자원의 수(m)로는 5, 10, 30을 사용하였고 아이템의 수(n)로는 100, 250, 500을 고려하였으며, 이와 같은 $m.n$의 각 조합에 대해 30개씩, 총 270개의 테스트 문제를 생성하였다. 즉, 자원의 수가 5개이고 아이템의 수가 100개인 30개의 문제들부터 자원의 수가 30개이고 아이템의 수가 500개인 30개의 문제까지 다양한 난이도의 문제를 만든 것이다. 각 문제에 있어서 아이템 별 값(p_j)과 자원별 요구량(w_{ij}), 그리고 자원별 한계치(c_i)는 다음과 같이 결정하였다. 먼저 w_{ij}는 0과 1000 사이의 임의의 값으로 설정하고 c_i는 식 (4)에 따라 생성하였는데, 이때 각 $m.n$의 30개 문제들을 기준으로 10개 문제는 α = 0.25, 다른 10개 문제는 α = 0.5, 나머지 10개의 문제 α = 0.75로 설정하여 생성하였다. 긴장률(tightness ratio)을 의미하는 α는 자원 제약조건을 만족하는 아이템들이 포함될 비율과 관련된 것으로서 긴장률이 0.25인 경우에는 전체 아이템들 중 약 25% 정도의 아이템들이 선택될 것으로 기대되며, 0.5는 약 50%, 0.75는 약 75% 정도의 아이템들이 선택될 것으로 기대된다. 아이템 별 p_j는 식 (5)에 따라 생성되며 여기서 q_j는 0과 1사이의 임의의 실수값을 의미한다. 따라서 p_j는 500과 1500 사이의 임의의 값을 갖되 해당 아이템의 자원 요구량(w_{ij})의 평균값과 밀접한 관련이 있게 된다. 일반적으로 다차원 배낭 문제의 경우 p_j, w_{ij}, c_i의 값으로 무작위 값을 가진 문제보다 상호 관련성이 있는 값으로 설정된 문제가 훨씬 어려운 것으로 알려져 있다.

$$c_i = \alpha \sum_{j=1}^{n} w_{ij}, \; \alpha = 0.25, 0.5, 0.75 \tag{4}$$

$$p_j = \sum_{i=1}^{m} w_{ij}/m + 500 q_j, \; j = 1,...,n \tag{5}$$

이후의 많은 연구들이 OR-Library의 테스트 문제를 대상으로 실험을 수행하였는데 크게 완전 탐색(exact search) 기법과 휴리스틱 탐색(heuristic search) 기법으로 나눌 수 있다. 완전 탐색 기법은 기본적으로 완전 열거 방식을 사용하기 때문에 최적해의 도출을 보장하지만 최적해를 찾기까지 과도한 시간을 요구하는 것으로 알려져 있다. 반면에 휴리스틱 탐색 기법은 최적해의 도출 여부를 검증할 수는 없지만 빠른 시간 내에 비교적 좋은 해를 구할 수 있는 것으로 알려져 있다.

완전 탐색 기법의 대표적인 연구로는 분지한계법을 기반으로 한 연구가 있다[14, 15]. 사실 분지한계법을 기반으로 하고 있는 정수계획법의 현재 기술 수준으로 볼 때 $m.n$이 5.100, 5.250, 5.500, 10.100, 10.250, 30.100인 문제들은 정수계획법만으로도 쉽게 최적해가 도출될 수 있다. 따라서 참고문헌 [14]와 [15]에서는 이외의 문제들에 있어서 최적해를 도출하기 위한 분지한계법의 적용 방안을 제시하였다. 그 결과, 참고문헌 [14]에서는 10.500 문제들에 대해 최적해를 도출할 수 있었으며 [15]에서는 30.250 문제들 중 일부 문제들에 대해 최적해를 도출할 수 있었다. 그러나 대다수의 30.250 문제들과 30.500 문제들에 대한 최적해 도출 및 검증은 여전히 어려운 상태이다.

휴리스틱 탐색 기법의 대표적인 연구로는 타부 탐색과 선형계획법을 결합한 연구[16]와 선형계획법만을 기반으로 한 연구[17, 18] 그리고 정수계획법과 지역탐색을 결합한 연구가 있다[1]. 참고문헌 [16]에서는 5.500, 10.500, 30.500 문제들에 대한 실험 결과를 제시하였으며 현재까지도 전반적인 성능에 있어서 가장 좋은 것으로 인정받고 있다. 참고문헌 [17]에서도 n이 500인 문제들에 대한 연구 결과를 제시하였는데, 5.500과 10.500 문제들에 있어서는 참고문헌 [16]에 근접하는 성능을 발휘하였으나 30.500 문제들에 있어서는 크게 미치지 못하는 결과를 보였다. 참고문헌 [18]에서는 10.500, 30.250, 30.500 문제들을 대상으로 한 실험 결과

를 제시하였는데, 10.500 문제들에 있어서는 참고문헌 [16]의 결과를 능가하는 결과를 보였으나 30.500 문제들에 있어서는 [16]의 결과에 미치지 못하는 결과를 보였다. 참고문헌 [1]에서는 30.500 문제들을 대상으로 정수계획법 기반 지역 탐색을 적용한 결과를 제시하였으며, 6개 문제에 있어서 참고문헌 [16]의 결과보다 더 좋은 해를 도출할 수 있으며 평균적으로도 성능이 더 좋음을 확인할 수 있었다. 본 논문에서는 참고문헌 [1]의 확장 연구로서 다양한 정수계획법 기반 지역 탐색을 제시할 뿐만 아니라 30.500 문제들 이외의 문제들에 대해 적용한 결과를 지금까지의 가장 좋은 결과들과 비교해 봄으로써 다시 한번 정수계획법 기반 지역 탐색의 성능을 검증한다.

III. 다차원 배낭 문제를 위한 정수계획법 기반 지역 탐색의 분류 및 구현

기존의 정수계획법 기반 지역 탐색에 있어서 지역 탐색은 단순 언덕오르기 탐색과 가장 유사하다. 그러나 정수계획법과 결합되는 지역 탐색에 따라 다양한 형태의 정수계획법 기반 지역 탐색이 만들어질 수 있다. 지역 탐색 기법들 중 가장 기본이 되는 탐색 기법은 언덕오르기 탐색과 단순 언덕오르기 탐색이라 할 수 있다. 그리고 이와 같은 단순한 지역 탐색의 단점인 지역 최적화 현상을 방지하기 위해 제안된 많은 탐색 기법들 중 타부 탐색과 시뮬레이티드 어닐링이 가장 대표적인 기법으로 알려져 있다. 본 논문에서는 정수계획법과 각각의 지역 탐색 기법을 결합한 방법을 해당 지역 탐색 기법의 이름을 붙여 다음과 같이 명명하였다.

① 정수계획법 기반 언덕오르기 탐색
② 정수계획법 기반 타부 탐색
③ 정수계획법 기반 단순 언덕오르기 탐색
④ 정수계획법 기반 시뮬레이티드 어닐링

모든 기법에 있어서 기본적인 수행 방식은 그림 1에서 설명한 기존의 정수계획법 기반 지역 탐색과 유사하다. 먼저 초기해를 생성하여 최초의 현재해를 만들고, 현재해를 대상으로 이웃해 하나를 만든 후 현재해의 갱신 여부를 결정하는 과정을 반복적으로 수행하게 된다. 이때 현재해를 기반으로 이웃해 하나를 만드는 과정을 정수계획법이 담당하게 되는데 정수계획법 입장에서는 내부적으로 많은 해들을 대상으로 하나의 해를 찾아내게 되는 것이다. 따라서 네 가지 정수계획법 기반 지역 탐색은 바로 결합되는 지역 탐색의 특성에 부합하도록 정수계획법을 어떻게 적용하느냐에 따라 구분되는 것이라 할 수 있다. 본 장에서는 각각의 정수계획법 기반 지역 탐색의 특징을 설명하고 이를 달성하기 위해 이웃해 생성 시 정수계획법을 어떻게 적용하는지를 설명한다. 여기서 정수계획법을 적용한다는 것은 결국 대상 문제에 대한 목적함수와 제약조건을 기술하는 것을 의미하며, 결국 이는 다차원 배낭 문제의 식 (1)~(3) 외에 해당 지역 탐색을 위해 새롭게 추가되는 제약조건을 기술하는 것을 의미한다.

1. 정수계획법 기반 언덕오르기 탐색

정수계획법 기반 언덕오르기 탐색 알고리즘은 그림 2와 같다. 일반적인 정수계획법 기반 지역 탐색 알고리즘인 그림 1과 비교했을 때 가장 큰 차이점은 k의 의미라 할 수 있다. 정수계획법 기반 언덕오르기 탐색에서 이웃해는 현재해의 배낭에 포함된 아이템들 중 일부 아이템들을 선택해 배낭으로부터 빼고, 배낭에 포함되지 않은 아이템들 중 일부 아이템들을 선택해 배낭에 포함시키는 것으로 정의한다. 따라서 기존 정수계획법 기반 지역 탐색에서 k가 이웃해 생성을 위해 정수계획법을 수행하는 데 참여하는 아이템의 개수를 의미하는 데 반해, 정수계획법 기반 언덕오르기 탐색에서 k는 현재해에서 선택된 아이템들 또는 선택되지 않은 아이템들로부터 선택될 아이템의 최대 개수를 의미한다. 그림 2에서 여기서 S_0과 S_1은 각각 현재해에서 x_i의 값이 0과 1인 인덱스 i의 집합으로서 배낭에 포함되지 않은 아이템들과 배낭에 포함된 아이템들의 집합을 의미한다.

Algorithm IPbHC (정수계획법 기반 언덕오르기 탐색)
  x : Variable vector (Current solution).
  Obj : Objective function.
  S_0 : The set of unselected items from x
  S_1 : The set of selected items from x
  k : The maximum number of selected variables to be exchanged from S_0 and S_1 respectively.
  IP : An integer programming solver.
Begin
  x = Make an initial solution using a heuristic method
  While stopping condition is not met **Do**
    Add *Obj* and all constraints to *IP* with k, S_0, S_1
    x = Make a neighbor solution with *IP*
  End While
  return x
End Begin

그림 2. 정수계획법 기반 언덕오르기 탐색
Fig. 2. Integer Programming-based Hill-Climbing Search

언덕오르기 탐색에서는 배낭에 포함된 아이템들과 배낭에

포함되지 않은 아이템들 중 각각 k개 이하를 선택하는 모든 조합에 대해 값을 변경해 보고 가장 좋은 해를 다음 현재해로 설정하되 이 해가 현재해보다 좋지 않으면 프로그램은 종료하게 된다. 그러나 이를 약간 변형하여 목적함수값이 현재해와 같을 경우에도 다음해로의 이동을 허용하는 것이 일반적이며 본 논문에서도 이와 같은 방식을 따른다. 따라서 그림2에서 제시한 바와 같이 정수계획법 기반 언덕오르기 탐색에서는 그림 1에서와 같이 사전에 k개의 변수를 선택하는 과정은 불필요하다. 이웃해를 만드는 대상이 고정된 k개가 아닌 어떠한 k개의 조합도 가능하기 때문이다. 또한 어떤 변수에 대한 값도 미리 정해지지 않기 때문에 그림 1에서와 같이 선택되지 않은 변수들의 값을 미리 고정하는 과정이 불필요하다. 단, 정수계획법 수행 시 정수계획법 기반 언덕오르기 탐색의 개념에 맞게 해를 탐색할 수 있도록 k, S_0, S_1을 사용하여 다음과 같은 세 가지 제약조건을 추가한다.

- 제약조건1 : 현재해에서 배낭에 포함된 아이템들 중 k개 이하를 빼고 배낭에 포함된 아이템들 중 k개 이하를 넣는 모든 이웃해들을 고려해야 한다. 본 제약조건은 식 (6)과 (7)에 의해 달성될 수 있다. 먼저 식 (7)은 현재 배낭에 포함된 아이템들 중 k개 이하만 배낭에서 제외될 수 있음을 의미한다. 즉 $|S_1|-k$개 이상의 아이템들은 또 다시 선택되어야만 한다. 그리고 식 (6)은 모든 변수들의 합이 현재 선택된 아이템들의 개수에 k를 더한 값과 같거나 작아야 됨을 의미한다. 결국 식 (6)과 (7)에 의해 현재 배낭에 포함되지 않은 아이템들 중 k개 이하만 배낭에 포함될 수 있게 된다.

$$\sum_{i=1}^{n} x_i \le |S_1| + k \quad (6)$$

$$\sum_{i \in S_1} x_i \ge |S_1| - k \quad (7)$$

- 제약조건2 : 목적함수값이 현재해와 같거나 그 이상이어야 한다. 이 제약조건은 식 (8)에 의해 간단히 표현될 수 있는데, 여기서 Obj_x는 현재해 x의 목적함수값을 의미한다.

$$Obj \ge Obj_x \quad (8)$$

- 제약조건3 : 지금까지 현재해로 설정되었던 해들과는 달라야 한다. 제약조건1과 제약조건2를 만족하는 이웃해들 중에는 현재해를 포함하여 지금까지 현재해였던 해들이 포함되어 있다. 그러나 일반적인 언덕오르기 탐색에서 이웃해라 함은 현재해는 제외하는 것이 보통이다. 또한 지금까지 현재해로 설정되었던 해로 되돌아가는 것 역시 동일한 과정을 반복하게 되므로 이들을 현재해에서 제외하는 것이 바람직하다. 따라서 식 (9)와 같은 제약조건을 통해 현재해와 이전 현재해로의 복귀를 방지한다. 지금까지 현재해로 설정된 해들이 총 b개라고 하자. 식 (9)를 통해 각각의 이전 현재해들을 기준으로 배낭에 포함된 아이템들 중 하나 이상이 제외되거나 배낭에 포함되지 않은 아이템들 중 하나 이상이 포함되도록 되어 있다. S_{j1}과 S_{j0}는 각각 j번째 현재해에 있어서 배낭에 포함된 아이템들의 인덱스 집합과 포함되지 않은 아이템들의 인덱스 집합을 의미한다.

$$\sum_{i \in S_{j1}} x_i + \sum_{i \in S_{j0}} (1 - x_i) \le n - 1, \quad j = 1, \ldots b \quad (9)$$

일반적인 언덕오르기 탐색에서는 k 값을 1, 2 정도로 아주 작게 설정하는 것이 일반적이다. k 값을 크게 설정하면 이웃해의 개수가 기하급수적으로 늘어나 한 번의 이웃해 이동을 위해 매우 많은 시간이 소요되기 때문이다. 이를 확인하기 위해 n이 500이고 긴장률이 0.5인 다차원 배낭 문제를 대상으로 간단한 실험을 수행해 보았다. 이 경우 일반적인 언덕오르기 탐색을 통해 k값을 1로 설정하여 한 번의 이웃해 이동을 하기 위해서는 약 0.5초가 소요됨을 확인하였다. 배낭에 포함된 아이템의 개수가 50%인 250개라고 가정하면 k가 1인 경우 대략 62,500개($_{250}C_1 \times _{250}C_1$)의 이웃해가 생성된다. 그렇다면 k가 2인 경우에는 약 968,765,625($_{250}C_2 \times _{250}C_2$)개가 생성되기 때문에 한 번의 이웃해 이동을 위해 약 7,750초가 소요된다. 이와 같은 방식으로 계산해 보면 k가 3인 경우에는 약 14,711시간이나 필요하다는 결론이 나오게 된다. 이와 같이 k 값이 조금만 커지더라도 이웃해의 규모가 기하급수적으로 늘어나 일반적인 언덕오르기 탐색으로는 시간 제약 상 실행이 불가능함을 알 수 있다. 그러나 실험에 의하면 정수계획법을 활용하여 이웃해를 생성하는 경우 k가 10일 때조차도 수백초 정도면 한 번의 이웃해 이동이 가능한 것으로 나타났다. 물론 현재해가 좋으면 좋을수록 소요 시간이 길어지며 메모리 소요량이 높아져 프로그램이 중단되는 경우가 많다. 그렇다 하더라도 k 값이 커짐에 따라 제한된 시간 내에 단 한 번의 이웃해 이동도 불가능했던 단순 언덕오르기 탐색에 비하면 활용도가 매우 높을 것으로 판단된다. k 값이 커지면 커질수록 이웃해의 범위는 넓어지게 되고 그만큼 지역최적해에 머무는 지역최적화 현상을 방지할 수 있게 되기 때문이다.

2. 정수계획법 기반 타부 탐색

타부 탐색은 언덕오르기 탐색의 지역최적화 현상을 방지하게 위해 만들어진 탐색 알고리즘으로서 가장 핵심적인 내용은 이웃해들 모두 현재해보다 좋지 않다 하더라도 그 중에서 가

장 좋은 해로의 이동을 허용하는 것이다[19]. 이때 이전 현재해로 이동하지 않도록 하여 동일한 탐색 과정이 반복되는 것을 방지하는데 이전 현재해에 대한 정보를 저장하는 곳을 타부 리스트(tabu list)라고 한다. 일반적으로는 이웃해의 이동이 빈번히 일어나게 되므로 타부 리스트에 모든 이전 현재해를 저장하고 매번 이 해들과 새로운 이웃해를 비교하는 것은 매우 어려운 일이다. 따라서 타부 리스트에는 해 자체보다는 이전해로 복귀하게 될 것으로 예상되는 행동의 특징을 저장하고 해당 행동을 취함으로써 생성되는 이웃해를 후보에서 제외하게 된다. 물론 몇 번의 이웃해 이동이 이루어진 후에는 해당 행동을 취한다 하더라도 이전해로 되돌아가 가능성이 낮아지므로 해당 행동의 특징을 타부 리스트에서 제거하게 된다. 그러나 정수계획법 기반 타부 탐색에서는 k 값으로 비교적 큰 값을 사용하게 되고 이에 따라 한 번의 이웃해 이동을 위해 많은 시간이 소요되므로 이전 현재해들을 모두 저장하고 비교한다 하더라도 큰 부담이 되지 않을 것으로 판단하였다.

타부 탐색은 언덕오르기 탐색과 유사하기 때문에 앞서 설명한 정수계획법 기반 언덕오르기 탐색에서 정의한 제약조건들을 그대로 사용하면 된다. 즉, 제약조건1을 통해 k개 이하의 아이템들에 대한 이웃해 탐색이 이루어지며 제약조건3을 통해 이전 현재해로 복귀하는 것을 방지하게 된다. 단, 현재해보다 좋지 않은 해로의 이동이 가능해야 하므로 현재해와 같거나 좋은 해만을 허용하는 제약조건2만 제외하면 된다.

3. 정수계획법 기반 단순 언덕오르기 탐색

정수계획법 기반 단순 언덕오르기 탐색은 그림 1의 알고리즘과 동일하게 동작한다. 먼저 탐색 대상이 되는 k개의 아이템들을 선택하는데 현재해를 기준으로 배낭에 포함되어 있는 아이템들은 모두 선택하게 되고 그 외에 포함되지 않은 아이템들 중에 나머지를 선택하게 된다. 따라서 이웃해 생성을 위한 정수계획법 실행 시에는 선택되지 않은 아이템들에 대한 변수들은 고려 대상에서 제외하면 된다. 이 과정은 일종의 문제 축소 과정으로 볼 수 있다. 그 외에 필요한 제약조건은 목적함수값이 현재해와 같거나 좋아야 한다는 것으로서 이 제약조건은 정수계획법 기반 언덕오르기 탐색의 제약조건2를 그대로 사용하면 된다.

여기서 한 가지 주의할 사항은 정수계획법 기반 단순 언덕오르기 탐색의 경우 현재해를 포함하여 이전 현재해로 복귀하는 것을 허용한다는 것이다. 일반적인 단순 언덕오르기 탐색에서도 현재해와 목적함수값이 같은 이전 현재해로의 복귀를 금지하지는 않는다. 그렇지만 이웃해의 범위에 현재해는 제외하는 것이 보통이다. 그런데 탐색이 진행됨에 따라 현재해가 상당히 좋은 수준에 이르게 되면 현재해를 이웃해로 허용하는 것이 허용하지 않는 것보다 정수계획법을 통해 훨씬 빨리 최적해를 도출할 수 있는 것으로 나타났다. 물론 현재해가 또 다시 다음 현재해로 도출되는 경우가 많다. 그러나 설령 현재해가 또 다시 현재해로 설정된다 하더라도 다음 이웃해 생성을 위해 포함되는 k개의 아이템을 어떻게 선택하느냐에 따라 더 좋은 해의 도출이 가능하기 때문에 굳이 더 많은 시간을 소모하면서 현재해와 같거나 더 좋지 않은 해를 찾을 필요가 없는 것이다. 물론 이전 현재해로의 복귀를 허용할 것인지 허용하지 않을 것인지는 대상 문제의 특성에 따라 달라질 수 있을 것으로 예상된다.

4. 정수계획법 기반 시뮬레이티드 어닐링

시뮬레이티드 어닐링은 단순 언덕오르기 탐색을 기반으로 하되 타부 탐색과 마찬가지로 언덕오르기 탐색의 지역최적화 현상을 방지하기 위해 현재해보다 좋지 않은 경우에도 이웃해로의 이동을 허용한다[20]. 현재해보다 좋지 않은 이웃해로의 이동 여부는 식 (10)의 확률 p에 따라 결정된다. 여기서 e는 자연상수를 의미하고 ΔE는 이웃해의 목적함수값에서 현재해의 목적함수값을 뺀 값을 의미하며 T는 온도를 의미한다. 이 확률이 적용되는 경우는 이웃해가 현재해보다 좋지 않을 경우, 즉 이웃해의 목적함수값이 현재해의 목적함수값보다 작을 경우이므로 ΔE는 음수가 되고 결국 이웃해의 값이 크면 클수록 확률값은 커지게 된다. 또한 T는 양수값으로 설정하는데 이 값이 크면 클수록 확률값은 커지게 된다. 탐색 초반에는 이 값을 크게 설정하여 이동 확률을 높여주며 탐색이 진행될수록 이 값을 점점 작게 만들어 이동 확률을 낮춰주게 된다. 본 연구에서는 T의 초기값을 10으로 설정하였고 이웃해로의 이동 여부를 결정할 때마다 0.05씩 감소시켰다.

$$p = e^{\Delta E / T} \tag{10}$$

정수계획법 기반 시뮬레이티드 어닐링에서 이웃해 생성을 위한 정수계획법을 구현할 때는 먼저 정수계획법 기반 단순 언덕오르기 탐색과는 달리 현재해와 같거나 좋아야 한다는 제약조건2를 제외한다. 그리고 다음과 같은 제약조건4를 추가한다.

- 제약조건4 : 이웃해는 현재해와 달라야 한다. 만약 현재해를 이웃해로 허용한다면 현재해보다 좋지 않은 해의 도출이 불가능하게 되며 이는 시뮬레이티드 어닐링의 개념이 맞지 않다. 따라서 식 (11)과 같은 제약조건을 통해 정수

계획법에 의해 현재해가 도출되지 않도록 한다. 식 (11)에서 S_1과 S_0은 각각 현재해에 있어서 배낭에 포함된 아이템들의 인덱스 집합과 포함되지 않은 아이템들의 인덱스 집합을 의미한다.

$$\sum_{i \in S_1} x_i + \sum_{i \in S_0} (1 - x_i) \leq n - 1 \quad (11)$$

정수계획법을 통해 최종적으로 도출된 이웃해가 현재해와 같거나 더 좋다면 다음 현재해로 결정하게 되고 그렇지 않다면 식 (10)의 확률에 따라 이동 여부를 결정하게 된다.

IV. 실험 결과

1. 실험 환경 및 데이터

본 연구에서는 정수계획법 개발 도구로 IBM ILOG CPLEX 12.1을 사용하였다[21]. CPLEX는 선형계획법 및 정수계획법 개발을 위한 상용 라이브러리로서 현재 상업적 또는 학술적 목적으로 전 세계적으로 가장 많이 사용되고 있으며 그 성능 또한 탁월한 것으로 인정받고 있다.

실험 데이터로는 OR-Library의 270개 테스트 문제들 중 30.500 문제들을 제외한 240개 문제들을 대상으로 하였다. 그런데 이 문제들 중에는 정수계획법만으로 쉽게 최적해를 도출할 수 있는 문제들이 포함되어 있다. 따라서 CPLEX를 활용한 정수계획법만으로 최적해의 도출 여부를 판정하는 것이 불가능한 문제들을 대상으로 하였으며, 이는 표 1과 같이 총 62개의 문제들로 구성된다. 표 1에서 Problem은 각 문제의 이름을 의미하는 것으로서 자원의 개수(m), 아이템의 개수(n), 해당 그룹 내 번호로 표기하였으며, Objective는 정수계획법만으로 도출한 최종해의 목적함수값을 의미한다.

본 논문의 모든 실험은 Intel Core2 Duo 3GHz, 2GB RAM PC 상에서 수행되었으며, 각 실험 당 수행 시간은 10시간으로 제한하였다. 기존 연구들을 고려해 볼 때 10시간 내에 해당 연구들과 유사한 결과를 도출할 수 있다면 유의미한 성과로 인정할 수 있을 것으로 판단하였다.

2. 실험 결과 및 분석

실험은 크게 세 부분으로 나누어진다. 첫 번째는 정수계획법 기반 언덕오르기 탐색(IPbHC)과 정수계획법 기반 타부 탐색(IPbTS)에 있어서 k 값의 변화에 따른 실험 결과를 살펴본다. 두 번째는 정수계획법 기반 단순 언덕오르기 탐색(IPbSHC)과 정수계획법 기반 시뮬레이티드 어닐링(IPbSA)에 있어서 k 값에 따른 실험 결과를 살펴본다. 마지막으로 정수계획법 기반 단순 언덕오르기 탐색(IPbSHC)과 기존 연구들 중 가장 좋은 결과를 보인 연구들을 비교 분석한다. 탐색 기법에 따라서는 매 실험 시 마다 다른 결과를 보일 수도 있기 때문에 통계적 유효성을 확보하기 위해서는 동일한 테스트 문제 및 파라미터 값에 대해 수차례의 실험이 불가피하다. 또한 각 문제 별로 1회 실험 당 10시간의 수행 시간을 허용했기 때문에 모든 탐색 기법에 있어서 모든 테스트 문제 및 파라미터에 대해 실험을 수행하는 것은 현실적으로 어려운 일이다. 이에 따라 첫 번째와 두 번째 실험에서는 일부 테스트 문제들을 대상으로 하여 실험을 수행함으로써 전체적인 경향을 살펴보았으며, 본 논문의 결론에 해당하는 마지막 실험인 기존 연구와의 비교를 위해서는 모든 문제들을 대상으로 실험을 수행하였다.

표 1. 실험 데이터
Table 1. Experimental Data

No.	Problem	Objective	No.	Problem	Objective
1	5.500-3	120794	32	10.500-24	304352
2	5.500-8	121575	33	10.500-25	301796
3	10.250-3	60989	34	10.500-26	304949
4	10.250-7	58933	35	10.500-27	296457
5	10.250-11	108715	36	10.500-28	301353
6	10.250-12	108922	37	10.500-29	307078
7	10.250-18	109822	38	30.250-0	56824
8	10.500-0	117792	39	30.250-1	58520
9	10.500-1	119139	40	30.250-2	56572
10	10.500-2	119211	41	30.250-3	56930
11	10.500-3	118813	42	30.250-4	56629
12	10.500-4	116471	43	30.250-5	57205
13	10.500-5	119481	44	30.250-6	56353
14	10.500-6	119777	45	30.250-7	56457
15	10.500-7	118320	46	30.250-8	57442
16	10.500-8	117781	47	30.250-9	56447
17	10.500-9	119191	48	30.250-10	107752
18	10.500-10	217377	49	30.250-11	108392
19	10.500-11	219063	50	30.250-12	106442
20	10.500-12	217792	51	30.250-13	106837
21	10.500-13	216843	52	30.250-14	107414
22	10.500-14	213846	53	30.250-15	107271
23	10.500-15	215039	54	30.250-16	106339
24	10.500-16	217887	55	30.250-17	104032
25	10.500-17	219984	56	30.250-18	106835
26	10.500-18	214317	57	30.250-19	105780
27	10.500-19	220846	58	30.250-20	150097
28	10.500-20	304347	59	30.250-23	153182
29	10.500-21	302379	60	30.250-25	148574
30	10.500-22	302408	61	30.250-28	149554
31	10.500-23	300747	62	30.250-29	149668

2.1 IPbHC와 IPbTS

앞서 설명한 바와 같이 일반적으로 언덕오르기 탐색과 타부 탐색에서는 현재해를 기준으로 고려할 수 있는 모든 이웃해들을 대상으로 다음 현재해를 결정한다. 따라서 동일한 목

적함수값을 가진 이웃해들에 대한 처리만 일관성이 있다면 매 실험 시마다 동일한 결과가 도출된다. 실험에 의하면 IPbHC와 IPbTS 역시 항상 동일한 결과가 도출되었다. 따라서 각 테스트 문제 및 k 값에 대해 1회의 실험만 수행하였다. 대상 문제로는 아이템의 개수, 자원의 개수, 긴장률을 고려하여 총 15개의 문제를 대상으로 하였다. 이웃해 생성 시 교체하는 아이템의 개수인 k 값은 3부터 10까지 다양한 값을 사용하였다. 그리고 초기해로는 정수계획법과 그리디 탐색을 통해 도출한 값을 각각 사용하였다. 정수계획법에 의한 초기해는 표 1과 같다. 문제 별로 정수계획법에 의해 해당 초기해가 생성되기까지 소요되는 시간에 차이가 있지만 짧게는 1초, 길게는 1200초 정도로 비교적 짧은 시간 내에 도출이 가능하다. 그리디 탐색 방법으로는 $p_j/\sum_{i=1}^{m} w_{ij}$의 값이 큰 아이템, 즉 아이템의 값은 크고 전체 자원 요구량은 작은 아이템부터 하나씩 배낭에 추가하되 각 자원 별 한계치를 넘지 않는 범위 내에서 마지막 아이템까지 추가를 시도하였다. 실험 결과에 의하면 정수계획법 초기값이 그리디 탐색 초기값보다 월등히 좋은 것으로 나타났다. 예를 들면 10.500-0의 경우 정수계획법 초기해와 그리디 탐색 초기해는 각각 목적함수값이 117792와 110748인 해가 도출되었다.

IPbHC와 IPbTS에 대한 실험 결과로는 원칙적으로 대상 문제들에 대해 네 가지 결과가 제시될 수 있다. 정수계획법 초기해를 사용한 IPbHC, 그리디 탐색 초기해를 사용한 IPbHC, 정수계획법 초기해를 사용한 IPbTS, 그리디 탐색 초기해를 사용한 IPbTS가 그것이다. 하지만 정수계획법 초기해를 사용한 IPbHC의 경우 초기해 자체가 매우 좋은 상태이기 때문에 그 자체로 지역 최적해인 경우가 많았으며 k 값이 5 이상만 되더라도 메모리 문제로 인해 이웃해로의 이동이 불가능한 경우가 대부분이었다. 그리고 예상한 바와 같이 좋은 초기해로부터 출발할 때 최종 결과가 전반적으로 더 좋아지는 경향이 있었다. 따라서 여기서는 상대적으로 가장 흥미로운 결과를 보인 정수계획법 초기해를 사용한 IPbTS에 대한 실험 결과만을 제시하였으며 그 결과는 표 2와 같다.

각 테스트 문제 별로 가장 좋은 결과에 대해서는 배경 및 글자색을 진하게 표시하였으며 'MEM'은 메모리 문제로 인해 이웃해 생성이 불가능함을 표시한 것이다. 먼저 일반적인 언덕오르기 탐색에서는 k가 3만 되더라도 모든 이웃해를 고려하는 것이 불가능한 반면에 정수계획법을 활용한 경우에는 문제의 난이도가 비교적 낮은 문제에 있어서는 6 이상의 k 값에 대해서도 이웃해로의 이동이 가능함을 알 수 있다. 그러나 본 실험만으로는 어떤 k 값이 더 효과적인지에 대해서는 판단이 어려운 상황이다. k 값이 작으면 단위 시간 당 이동 횟수가 많다는 장점이 있지만 k 값이 크면 한 번에 더 많은 이웃해들을 고려할 수 있다는 장점이 있다. 결국 최적의 k 값은 대상 문제 또는 데이터마다 달라질 것으로 판단된다. 한편 표 2에 의하면 초기해가 매우 좋은 상태이기 때문에 15개 문제 중 10개 문제에 있어서는 초기해보다 더 좋은 해를 도출하지 못하였다. 그러나 나머지 5개 문제에 있어서는 초기해보다 더 좋은 해를 도출할 수 있었으며 심지어 4개 문제에 있어서는 2.3절에서 제시할 IPbSHC의 최종 결과와 동일한 해인 최적해를 도출할 수 있었다. 비록 IPbTS가 다차원 배낭 문제에 대한 전체적인 성능에 있어서는 IPbSHC에 미치지 못하지만 대상 문제에 따라서는 효과적인 탐색 기법이 될 수 있을 것으로 예상된다.

표 2. IPbTS의 실험 결과
Table 2. Experimental Results of IPbTS

Problem	초기해 (IP)	k								IPbSHC와 동일 여부
		3	4	5	6	7	8	9	10	
5.500-3	120794	**120804**	**120804**	**120804**	**120804**	**120804**	**120804**	**120804**	MEM	O
10.250-3	**60989**	**60989**	**60989**	**60989**	**60989**	MEM	MEM	MEM	MEM	
10.250-11	108715	108715	**108717**	**108717**	**108717**	**108717**	MEM	MEM	MEM	O
10.500-0	**117792**	**117792**	MEM	MEM	MEM	MEM	MEM	MEM	MEM	
10.500-1	119139	**119150**	MEM	MEM	MEM	MEM	MEM	MEM	MEM	
10.500-10	**217377**	**217377**	MEM	MEM	MEM	MEM	MEM	MEM	MEM	
10.500-11	**219063**	**219063**	MEM	MEM	MEM	MEM	MEM	MEM	MEM	
10.500-20	**304347**	**304347**	**304347**	MEM	MEM	MEM	MEM	MEM	MEM	
10.500-21	**302379**	**302379**	**302379**	MEM	MEM	MEM	MEM	MEM	MEM	
30.250-0	**56824**	**56824**	**56824**	**56824**	MEM	MEM	MEM	MEM	MEM	
30.250-1	**58520**	**58520**	**58520**	**58520**	MEM	MEM	MEM	MEM	MEM	
30.250-10	**107752**	**107752**	**107752**	**107752**	**107752**	MEM	MEM	MEM	MEM	
30.250-11	**108392**	**108392**	**108392**	**108392**	**108392**	MEM	MEM	MEM	MEM	
30.250-20	150097	**150163**	**150163**	**150163**	**150163**	150138	**150163**	MEM	MEM	O
30.250-23	153182	**153234**	153221	**153234**	153221	153202	153190	MEM	MEM	O

2.2 IPbSHC와 IPbSA

IPbSHC와 IPbSA의 실험에서는 표 1의 테스트 문제들 중 8개의 문제들을 대상으로 하였다. IPbSHC와 IPbSA에 있어서 파리미터는 두 가지가 있다. 첫 번째는 이웃해 생성 시 정수계획법의 대상이 될 아이템의 개수 k이며, 두 번째는 정수계획법의 최대 수행 시간이다. k 값이 커지면 정수계획법을 사용한다 하더라도 빠른 시간 내에 최적해를 구하는 것이 매우 어려워진다. 따라서 최대 시간을 지정해야 하는데 이 시간을 너무 짧게 설정하면 좋은 해를 도출하기 어려워지며 너무 길게 설정하면 더 좋은 해를 찾지 못하는 상황에서도 불필요하게 시간을 소모할 가능성이 높아진다. 본 논문에서는 기존 연구 [1]과 같이 이웃해 생성을 위한 정수계획법의 최대 수행 시간을 200초로 설정하였다. k 값으로는 전체 아이템 개수의 60%, 70%, 80%, 90%, 96%로 지정하여 실험을 수행하였다.

IPbSA의 경우 대부분의 문제에 있어서 IPbSHC를 통해 찾은 가장 좋은 해를 찾을 수 있었지만 평균적으로는 IPbSHC보다 약간 좋지 않은 결과를 보였다. IPbSA는 현재해보다 좋지 않은 해로의 이동을 허용하는데 이와 같은 이동이 더 좋은 해로 이동하는 데 큰 영향을 미치지 않는 것으로 보인다. 오히려 k 값이 충분히 크기 때문에 IPbSHC와 같이 현재해로부터 다음해를 결정하는 데 포함시킬 아이템들을 어떤 조합으로 선택하느냐가 해를 개선시키는 데 더 큰 영향을 미치는 것으로 보인다. 따라서 본 논문에서는 표 3을 통해 IPbSHC의 결과만을 제시하였다. 물론 전체적으로는 IPbSA 또한 유사한 경향을 보이고 있다. 표 3의 수치는 3회 실험을 실시한 후 평균값을 취한 것이며 각 테스트 문제별로 가장 좋은 결과는 배경 및 글자색을 진하게 표시하였다. 긴장률이 0.75인 30.250-20과 10.500-20에 있어서는 기본적으로 배낭에 포함되는 아이템의 수가 약 75% 정도이기 때문에 k 값이 60%와 70%인 경우는 무의미하다.

표 3에 의하면 IPbSHC를 통해 30.250-0, 10.500-1, 10.500-11, 10.500-20에 있어서도 IPbTS보다 더 좋은 결과를 도출할 수 있음을 알 수 있다. k 값의 영향을 살펴보면 비교적 문제의 난이도가 낮은 10.250-11, 5.500-3에 있어서는 60% 이상이면 어떤 값이라도 큰 영향을 미치지 않는 것으로 보인다. 내부적으로 수렴 속도에는 차이가 있을 수 있으나 10시간이라는 최대 수행 시간 내에서는 모든 경우에 있어서 가장 좋은 결과를 도출할 수 있는 것으로 보인다. 반면에 문제의 난이도가 높고 긴장률이 높은 30.250-10, 30.250-20, 10.500-10, 10.500-20에 있어서는 비교적 k 값이 큰 경우에 더 좋은 결과를 도출할 수 있었다. 그런데 긴장률이 0.25인 30.250-0과 10.500-1의 경우에는 긴장률이 0.5 또는 0.75일 때보다는 작은 k 값에 있어서 더 좋은 결과가 도출되는 것으로 보이나 단정적으로 얘기하기는 힘들다. 30.250-0의 경우 더 큰 k 값에 대해서도 큰 차이가 없는 것으로 보이며 10.500-1의 경우 k 값이 480일 때에도 상당히 좋은 결과를 보이고 있다. 이 부분에 대해서는 추후 더 많은 실험이 필요할 것으로 판단된다.

표 3. IPbSHC의 실험 결과
Table 3. Experimental Results of IPbSHC

(a) n = 250

Problem	초기해 (IP)	k				
		150	175	200	225	240
10.250-11	108715	**108717**	**108717**	**108717**	**108717**	**108717**
30.250-0	56824	56830	**56842**	**56842**	56836	56836
30.250-10	107752	107752	107752	107758	**107770**	107765
30.250-20	150097	-	-	150096	**150163**	**150163**

(b) n = 500

Problem	초기해 (IP)	k				
		300	350	400	450	480
5.500-3	120794	120802	**120804**	**120804**	**120804**	**120804**
10.500-1	119139	119209	**119220**	119206	119213	119217
10.500-11	219063	219063	219064	219072	**219076**	**219076**
10.500-20	304347	-	-	304347	**304379**	304371

표 3의 결과는 정수계획법 초기해를 기반으로 수행한 것이다. 따라서 초기해 자체가 매우 좋은 해로부터 출발하였다고 볼 수 있으며, 그렇기 때문에 더 좋은 해를 보다 쉽게 도출할 수 있었다고 생각할 수도 있다. 그렇다면 그리디 탐색 초기해와 같이 상대적으로 좋지 않은 해로부터 출발할 경우 최종 결과에 미치는 영향을 확인할 필요가 있다. 표 4는 표 3의 각 문제에 대해서 그리디 탐색 초기해로부터 출발하여 IPbSHC를 수행한 결과로서 총 5회 실험 후 평균값을 취하였다. 10.250-11, 30.250-10, 30.250-20, 5.500-3에 있어서는 정수계획법 초기해와 동일한 결과를 보였다. 10.500-1과 10.500-11에 있어서도 그 차이는 미미한데 10.500-11의 경우 오히려 그리디 탐색 초기해가 더 좋은 결과를 보였다. 30.250-0의 경우에는 그리디 탐색 초기해의 결과가 다소 좋지 않게 나왔지만 그 차이는 크지 않은 편이다. 다만 10.500-20의 경우 그 차이가 큰 것으로 나타났다. 결론적으로 그리디 탐색 초기해를 사용한 경우에도 매우 빠른 속도로 정수계획법 초기해 수준의 해를 도출할 수 있는 것으로 보이며 이에 따라 전반적인 성능은 정수계획법 초기해에 비해 크게 뒤떨어지지 않을 것으로 예상된다. 그렇지만 10.500-20과 같이 문제의 난이도가 높은 경우 정수계획법

에 의한 초기해 수준의 해를 도출하는 데 더 많은 시간이 소요되어 전체적인 성능 또한 저하될 것으로 판단된다.

표 4. 초기해에 따른 IPbSHC의 실험 결과
Table 4. Experimental Results of IPbSHC for Initial Solution

Problem	k	초기해(IP)		초기해(Greedy)	
		초기해	최종해	초기해	최종해
10.250-11	225	108715	**108717**	106020	**108717**
30.250-0	175	56824	**56842**	52748	56835
30.250-10	225	107752	**107770**	103406	**107770**
30.250-20	225	150097	**150163**	147573	**150163**
5.500-3	350	120794	**120804**	120696	**120804**
10.500-1	350	119139	**119220**	116908	119219
10.500-11	450	219063	219076	212072	**219077**
10.500-20	450	304347	**304379**	299136	304362

2.3 기존 기법들과의 비교

마지막으로 모든 테스트 문제들을 대상으로 정수계획법 기반 단순 언덕오르기 탐색과 기존 연구들 중 가장 우수한 결과들을 비교하였다. IPbSHC에 있어서 k 값으로는 아이템의 개수가 250개인 경우 긴장률 0.25, 0.5. 0.75인 문제에 대해 각각 175, 225, 225를 사용하였으며, 아이템의 개수가 500개인 경우에는 각각 350, 450, 450을 사용하였다. 그리고 모든 문제에 대해서 총 10회 실험 후 그 결과를 정리하였다. 비교를 위한 기존 연구로는 [16], [17], [15], [18]을 사용하였는데 연구에 따라서는 대상으로 한 테스트 문제가 서로 다를 수도 있다. 예를 들어 [16]과 [17]에서는 5.500, 10.500 문제들만을 대상으로 하였으며 [18]에서는 10.500, 30.250 문제들만을 대상으로 하였다.

표 5는 비교적 난이도가 낮은 문제라 할 수 있는 5.500과 10.250 문제들에 대한 실험 결과이다. 표 5를 포함한 이후의 표에 있어서 각 기법 별로 참고문헌 번호와 연도 및 저자를 표기하였으며, CPLEX는 정수계획법만을 사용했을 때의 최종해로서 표 1과 동일하다. z는 해당 기법에 의해 도출된 가장 좋은 해의 목적함수값이고 t는 가장 좋은 해가 도출되기까지의 초 단위 시간을 의미하며 $\bar{z}$는 여러 번 실험을 수행한 경우 목적함수값들의 평균값을 의미한다. 각 문제에 있어서 가장 좋은 목적함수값에 대해서는 배경 및 글자색을 진하게 표시하였으며 최적해인 경우에는 목적함수값 뒤에 '*' 표시를 추가하였다. 최적해 여부는 완전 탐색 기법인 [15]의 결과를 참조하였다. 기존 연구들의 경우 z, t, $\bar{z}$ 별로 해당 논문을 통해 확인이 가능한 경우에만 이를 기술하였다. 표 5에 의하면 IPbSHC의 경우 모든 문제에 대해 최적해를 도출할 수 있을 뿐만 아니라 10회 모두 최적해를 도출하였음을 알 수 있다. 최적해를 도출하기까지의 소요 시간 또한 문제에 따라 다르지만 비교적 빠른 시간 내에 최적해를 도출할 수 있었다.

표 6은 10.500 문제들에 대한 실험 결과이며 표 7은 긴장률에 따른 그룹 별 평균값을 나타낸 것이다. 표 7에서 평균값은 소수점 이하 첫째 자리에서 반올림하였으나 IPbSHC와 [18]의 경우 공교롭게도 1의 자리까지 동일하여 두 기법에 대해서는 소수점 이하 둘째 자리에서 반올림한 결과를 표기하였다. 이와 같이 필요한 경우에는 소수점 이하 첫째 자리까지 표기하였다. 10.500 문제들의 경우에도 [15]에 의해 최적해가 판명되어 있는 상태이다. IPbSHC의 경우 30개 문제들 중 23개 문제들에 대한 최적해를 도출할 수 있었으며 나머지 7개 문제들에 있어서는 최적해를 도출하지 못하였다. 그러나 IPbSHC는 완전 탐색 기법인 [15]를 제외한 휴리스틱 방법들 중에서 가장 우수한 성능을 보인 [18]에 비해 매우 근소한 차로 더 좋은 결과를 도출할 수 있었다.

표 8은 30.250 문제들에 대한 실험 결과이며 표 9는 긴장률에 따른 그룹 별 평균값을 나타낸 것이다. 30.250 문제들의 경우 일부 문제들에 대해서만 [15]에 의해 최적해가 판명되어 있는 상태이다. 결과적으로 볼 때 IPbSHC는 모든 문제들에 있어서 가장 좋은 해를 도출할 수 있었으며 평균적으로도 다른 모든 기법들보다 더 좋은 결과를 도출할 수 있었다. 그런데 자세히 살펴보면 이는 초기해 및 이웃해 생성을 위해 사용한 정수계획법의 우수성에 기인한 것임을 알 수 있다. 예를 들면 IPbSHC는 30.250-11, 30.250-25에 있어서 [18]의 결과보다 더 좋은 결과를 도출할 수 있었는데 이는 사실상 정수계획법을 통해 생성된 초기해와 동일하다. 그만큼 정수계획법은 다차원 배낭 문제를 해결하는 데 효과적임을 알 수 있고 정수계획법 기반 지역 탐색 또한 이와 같은 정수계획법의 성능을 적극적으로 활용하기 위해 도입된 기법이다. 따라서 향후에도 정수계획법의 성능이 향상됨에 따라 정수계획법 기반 지역 탐색 또한 그 성능이 향상될 수 있을 것으로 판단된다.

V. 결 론

본 논문에서는 정수계획법 기반 지역 탐색을 활용하여 중소 규모의 다차원 배낭 문제를 해결하였으며, 실험 결과 기존의 가장 뛰어난 연구 결과들에 비해 비슷하거나 더 좋은 결과를 도출할 수 있음을 확인하였다. 이를 통해 정수계획법 기반 지역 탐색이 다차원 배낭 문제에 있어서 우수한 성능을 발휘함을 다시 한번 확인할 수 있었다. 또한 본 논문에서는

표 5. 5.500 및 10.250 문제에 대한 실험 결과
Table 5. Experimental Results for 5.500 and 10.250 Problems

Problem	CPLEX	[16] 2005 Vasquez		[17] 2009 WilBaut			[15] 2010 Boussier		IPbLS (IPbSHC)		
	z	z	t	z	t	$\bar{z}$	z	t	z	t	$\bar{z}$
5.500-03	120794	**1208054***	566	**120804***	3671	120755	**120804***	9	**120804***	925	**120804***
5.500-08	121575	121575	56360	**121586***	3767	121524	**121586***	32	**121586***	861	**121586***
10.250-03	60989	-	-	-	-	-	**61000***	1143	**61000***	4031	**61000***
10.250-07	58933	-	-	-	-	-	**58936***	2055	**58936***	789	**58936***
10.250-11	108715	-	-	-	-	-	**108717***	378	**108717***	1043	**108717***
10.250-12	108922	-	-	-	-	-	**108932***	2	**108932***	329	**108932***
10.250-18	109822	-	-	-	-	-	**109829***	10	**109829***	556	**109829***

표 6. 10.500 문제에 대한 실험 결과
Table 6. Experimental Results for 10.500 Problems

Problem	CPLEX	[16] 2005 Vasquez		[17] 2009 WilBaut			[15] 2010 Boussier		[18] 2012 Della	IPbLS (IPbSHC)		
	z	z	t	z	t	$\bar{z}$	z	t	z	z	t	$\bar{z}$
10.500-00	117792	117811	21845	117809	3086	117639	**117821***	88200	117809	117811	15577	117804
10.500-01	119139	119232	7163	119217	2068	119047	**119249***	246240	**119249***	119229	11661	119218
10.500-02	119211	**119215***	19205	**119215***	5028	**119059***	**119215***	66960	**119215***	**119215***	10684	119213
10.500-03	118813	118813	288	118825	3400	118622	**118829***	170640	118814	**118829***	10774	118819
10.500-04	116471	116509	58353	116514	241	116354	**116530***	302760	**116530***	116524	8813	116510
10.500-05	119481	**119504***	18720	119490	3130	119326	**119504***	8280	**119504***	**119504***	845	119503
10.500-06	119777	**119827***	82719	119794	2810	119628	**119827***	9720	**119827***	**119827***	6808	**119827***
10.500-07	118320	118329	98291	118333	6146	118159	**118344***	582120	118333	**118344***	6028	118327
10.500-08	117781	**117815***	12558	117781	2200	117633	**117815***	310680	117789	117786	7086	117782
10.500-09	119191	119231	30510	**119251***	4282	119075	**119251***	11160	**119251***	**119251***	22350	119236
10.500-10	**217377***	**217377***	64165	**217377***	9	217244	**217377***	360	**217377***	**217377***	463	**217377***
10.500-11	219063	**219077***	750	**219077***	1727	218940	**219077***	1800	**219077***	**219077***	1672	219076
10.500-12	217792	217806	99480	**217847***	428	217708	**217847***	360	**217847***	**217847***	1111	217843
10.500-13	216843	**216868***	476	**216868***	1032	216732	**216868***	360	**216868***	**216868***	487	**216868***
10.500-14	213846	213850	9252	213853	1707	213737	**213873***	213840	213859	213860	9064	213858
10.500-15	215039	**215086***	4953	**215086***	1317	214953	**215086***	360	**215086***	**215086***	631	215084
10.500-16	217887	**217940***	41803	**217940***	4606	217818	**217940***	48240	**217940***	**217940***	5170	217937
10.500-17	219984	219984	646	**219990***	5729	219843	**219990***	542880	219984	219984	631	219984
10.500-18	214317	214375	6978	214375	329	214233	**214382***	45720	**214382***	**214382***	33034	214374
10.500-19	220846	220899	34838	220886	3174	220750	**220899***	720	**220899***	**220899***	29079	220888
10.500-20	304347	**304387***	830	**304387***	5118	304275	**304387***	23760	**304387***	**304387***	6061	304373
10.500-21	**302379***	**302379***	59226	**302379***	1462	302259	**302379***	360	**302379***	**302379***	945	**302379***
10.500-22	302408	302416	3235	302416	85	302300	**302417***	241920	**302417***	302416	6758	302412
10.500-23	300747	300757	569	**300784***	2004	300637	**300784***	3240	**300784***	**300784***	2531	300769
10.500-24	304352	304374	17251	**304374***	1561	304224	**304374***	360	**304374***	**304374***	5016	304372
10.500-25	301796	**301836***	67073	**301836***	5314	301769	**301836***	106920	**301836***	**301836***	702	301808
10.500-26	304949	**304952***	27970	**304952***	326	304818	**304952***	360	**304952***	**304952***	13843	304951
10.500-27	296457	**296478***	22862	296472	785	296363	**296478***	3960	**296478***	**296478***	32225	296464
10.500-28	301353	**301359***	11508	301357	1015	301226	**301359***	29160	301357	**301359***	18086	301357
10.500-29	307078	**307089***	1037	**307089***	839	306981	**307089***	4320	**307089***	**307089***	1287	**307089***

표 7. 10.500 문제에 대한 실험 결과 요약
Table 7. Summary of Experimental Results for 10.500 Problems

α	CPLEX	[16] 2005 Vasquez		[17] 2009 WilBaut			[15] 2010 Boussier		[18] 2012 Della	IPbLS (IPbSHC)		
	z	z	t	z	t	$\bar{z}$	z	t	z	z	t	$\bar{z}$
0.25	118598	118629	34965	118623	3239	118454	**118639**	179676	**118632.1**	118632.0	10063	118624
0.5	217299	217326	26334	217330	2006	217196	**217334**	85464	217331.9	**217332.0**	8134	217329
0.75	302587	302603	21156	**302605.5**	1851	302485	**302606**	41436	302605.3	302605.4	8745	302597
평균	212828	212853	27485	212852	2365	212712	**212859**	102192	212856.4	**212856.5**	8981	212850

표 8. 30.250 문제에 대한 실험 결과
Table 8. Experimental Results for 30.250 Problems

Instance	CPLEX	[15] 2010 Boussier		[18] 2011 Della	IPbLS (IPbSHC)		
	z	z	t	z	z	t	$\bar{z}$
30.250-00	56824	**56842**	142920	**56842**	**56842**	841	56838
30.250-01	**58520**	58418	111240	**58520**	**58520**	257	58520
30.250-02	56572	**56614**	131760	**56614**	**56614**	2800	56582
30.250-03	**56930**	**56930**	360	**56930**	**56930**	8	56930
30.250-04	**56629**	**56629**	78120	**56629**	**56629**	37	56629
30.250-05	**57205**	**57205**	35280	**57205**	**57205**	297	57205
30.250-06	56353	56348	194040	**56357**	**56357**	7517	56354
30.250-07	**56457**	**56457**	720	**56457**	**56457**	58	56457
30.250-08	57442	57447	304920	**57474**	**57474**	24562	57447
30.250-09	**56447**	**56447**	360	**56447**	**56447**	1	56447
30.250-10	107752	107755	14760	**107770**	**107770**	1558	107770
30.250-11	**108392**	**108392**	2520	108379	**108392**	1442	108392
30.250-12	**106442**	**106442**	1080	**106442**	**106442**	298	106442
30.250-13	106837	**106876**	312840	**106876**	**106876**	8217	106876
30.250-14	**107414**	**107414**	11880	**107414**	**107414**	259	107414
30.250-15	**107271**	**107271**	17640	**107271**	**107271**	298	107271
30.250-16	106339	**106372**	73080	**106372**	**106372**	2856	106372
30.250-17	**104032**	**104032**	360	**104032**	**104032**	2508	104032
30.250-18	106835	**106856**	4320	**106856**	**106856**	4438	106856
30.250-19	**105780**	**105780**	12960	**105780**	**105780**	243	105780
30.250-20	150097	**150163***	63000	**150163***	**150163***	1325	**150163***
30.250-23	153182	**153234***	360	**153234***	**153234***	1587	**153234***
30.250-25	**148574**	**148574**	360	148559	**148574**	607	148574
30.250-28	149554	**149570**	60480	**149570**	**149570**	1281	149570
30.250-29	**149668**	**149668**	3240	**149668**	**149668**	540	149668

표 9. 30.250 문제에 대한 실험 결과 요약
Table 9. Summary of Experimental Results for 30.250 Problems

α	CPLEX	[15] 2010 Boussier		[18] 2011 Della	IPbLS (IPbSHC)		
	z	z	t	z	z	t	$\bar{z}$
0.25	56938	56934	99972	**56948**	**56948**	3638	56941
0.5	110654	110668	46767	110669	**110670**	2131	110670
0.75	150215	**150242**	25488	150239	**150242**	1068	150242
평균	95502	95509	63144	95514	**95516**	2553	95513

정수계획법 기반 지역 탐색의 다양한 적용 및 구현 방안을 제시하였다. 비록 기존의 정수계획법 기반 지역 탐색인 정수계획법 기반 단순 언덕오르기 탐색이 중소 규모의 다차원 배낭 문제에 있어서도 가장 좋은 결과를 보였지만, 대상 문제에 따라서는 정수계획법 기반 타부 탐색 등 다른 형태의 정수계획법 기반 지역 탐색 역시 활용 가능성이 높을 수 있음을 확인하였다.

그러나 과도한 실험 시간 등으로 인해 정수계획법 기반 지역 탐색에 대한 파라미터 실험에 있어서 최적의 값을 도출하는 데 어려움이 있었던 것으로 판단된다. 예를 들면, 정수계획법 기반 지역 탐색에서 k 값에 따라 그 결과가 달라질 수 있는데 다양한 k 값을 고려하지 못하고 충분한 횟수의 실험이 수반되지 못함에 따라 결국 최적의 k 값을 찾지 못한 것으로 추정된다. 그럼에도 불구하고 매우 우수한 성능을 발휘할 수 있었지만 파라미터를 적절히 조절하면 보다 좋은 결과를 도출할 수 있을 것으로 예상된다. 타부 탐색이나 시뮬레이티드 어닐링 등 다른 메타 휴리스틱 탐색들 역시 파라미터 조정에 어려움을 겪고 있는 것처럼 정수계획법 기반 지역 탐색의 파라미터를 최적화하는 것 자체도 쉽지 않을 것으로 보인다. 하지만 파라미터 최적화에 관한 연구는 정수계획법 기반 지역 탐색의 활용도를 높이기 위해 향후 과제로서 반드시 필요한 일이라 할 수 있다. 아울러 아직까지 정수계획법 기반 지역 탐색의 적용 사례가 드문 만큼 보다 다양한 조합최적화 문제로의 적용 및 분석을 통해 정수계획법 기반 지역 탐색의 우수성을 재검증하고 장단점을 파악할 필요가 있다.

참고문헌

[1] J. Hwang, S. Park, and I. Y. Kong, "An Integer Programming-based Local Search for Large-scale Multidimensional Knapsack Problems," International Journal on Computer Science and Engineering, Vol. 3, No. 6, pp. 2257-2264, June 2011.

[2] J. Hwang, and S. Kim, "An Integer Programming-based Local Search for Large-scale Maximal Covering Problems," International Journal on Computer Science and Engineering, Vol. 3, No. 2, pp. 837-843, Feb. 2011.

[3] L. A. Wolsey, "*Integer Programming*," Wiley, pp. 91-111, 1998.

[4] J. E. Beasley, "A Lagrangian Heuristic for Set Covering Problems," Naval Research Logistics, Vol. 37, No. 1, pp. 151-164, Feb. 1990.

[5] N. Mladenovic, J. Brimberg, P. Hansen and J. A. Moreno Perez, "The p-median Problem: A Survey of Metaheuristic Approaches," European Journal of Operational Research, Vol. 179, No. 3, pp. 927-939, June 2007.

[6] M. Diaby, "The Traveling Salesman Problem: A Linear Programming Formulation," WSEAS Transactions on Mathematics, Vol. 6, No. 6, pp. 745-754, May 2007.

[7] J. Hwang, and K. R. Ryu, "A Hybrid of Neighborhood Search and Integer Programming for Crew Schedule Optimization," Journal of KISS : Software and Applications, Vol. 31, No. 6, pp. 829-839, June 2004.

[8] S. Hasegawa, and Y. Kosugi, "Solving Nurse Scheduling Problem by Integer-programming-based Local Search," 2006 IEEE International Conference on Systems, Man and Cybernetics, pp. 1474-1480, Oct. 2006.

[9] M. Hewitt, G. L. Nemhauser, and M. W. Savelsbergh, "Combining Exact and Heuristic Approaches for the Capacitated Fixed Charge Network Flow Problem," INFORMS Journal on Computing, Vol. 22, No. 2, pp. 314-325, Spring 2010.

[10] J. Hwang, and S. Kim, "Integer Programming-based Local Search Technique for Linear Constraint Satisfaction Optimization Problem," Journal of The Korea Society of Computer and Information, Vol. 15, No. 9, pp. 47-55, Sep. 2010.

[11] J. Puchinger, G. R. Raidl, and U. Pferschy, "The Multidimensional Knapsack problem: Structure and Algorithms," INFORMS Journal on Computing, Vol. 22, No. 2, pp. 250-265, Spring 2010.

[12] P. C. Chu, and J. E. Beasley, "A Genetic Algorithm for the Multidimensional Knapsack Problem," Journal of Heuristics, Vol. 4, No. 1, pp. 63-86, 1998.

[13] J. E. Beasley, "OR-Library: Distributing Test Problems by Electronic Mail," The Journal of the Operational Research Society, Vol. 41, No. 11, pp. 1069-1072, 1990.

[14] S. Boussier, M. Vasquez, Y. Vimont, S. Hanafi, and P. Michelon, "Solving the 0-1 Multidimensional Knapsack Problem with Resolution Search," VI ALIO/EURO Workshop on Applied Combinatorial Optimization, May 2009.

[15] S. Boussier, M. Vasquez, Y. Vimont, S. Hanafi, and P. Michelon, "A Multi-level Search Strategy for the 0-1 Multidimensional Knapsack Problem," Discrete Applied Mathematics, Vol. 158, No. 2, pp. 97-109, Jan. 2010.

[16] M. Vasquez, and Y. Vimont, "Improved Results on the 0-1 Multidimensional Knapsack Problem," European Journal of Operational Research, Vol. 165, No. 1, pp. 70-81, August 2005.

[17] C. Wilbaut, and S. Hanafi, "New Convergent Heuristics for 0-1 Mixed Integer Programming," European Journal of

Operational Research, Vol. 195, No. 1, pp. 62-74, May 2009.

[18] F. Della Croce, and A. Grosso, "Improved Core Problem based Heuristics for the 0/1 Multidimensional Knapsack Problem," Computers & Operations Research, Vol. 39, No. 1, pp. 27-31, Jan. 2012.

[19] F. Glover, and M. Laguna, "*Tabu search*," Kluwer Academic Publishers, pp. 1-57, 1997.

[20] S. Russell, and P. Norvig, "*Artificial Intelligence: A Modern Approach*," Prentice Hall, pp. 115-116, 2005.

[21] "*IBM ILOG CPLEX V12.1: User's Manual for CPLEX*," International Business Machines Corporation, 2009.

저 자 소 개

황 준 하

1995: 부산대학교
컴퓨터공학과 공학사

1997: 부산대학교
컴퓨터공학과 공학석사

2002: 부산대학교
컴퓨터공학과 공학박사

현 재: 금오공과대학교
컴퓨터공학과 교수

관심분야: 인공지능, 최적화,
기계학습, 프로그래밍언어

Email : jhhwang@kumoh.ac.kr