Journal of The Korea Society of Computer and Information
Vol. 23 No. 12, pp. 1-9, December 2018

www.ksci.re.kr
https://doi.org/10.9708/jksci.2018.23.12.001

An Integer Programming-based Local Search for the Multiple-choice Multidimensional Knapsack Problem

Junha Hwang*

Abstract

The multiple-choice multidimensional knapsack problem (MMKP) is a variant of the well known 0-1 knapsack problem, which is known as an NP-hard problem. This paper proposes a method for solving the MMKP using the integer programming-based local search (IPbLS). IPbLS is a kind of a local search and uses integer programming to generate a neighbor solution. The most important thing in IPbLS is the way to select items participating in the next integer programming step. In this paper, three ways to select items are introduced and compared on 37 well-known benchmark data instances. Experimental results shows that the method using linear programming is the best for the MMKP. It also shows that the proposed method can find the equal or better solutions than the best known solutions in 23 data instances, and the new better solutions in 13 instances.

▸Keyword: Multiple-choice Multidimensional Knapsack Problem, Integer Programming, Local Search, Linear Programming

I. Introduction

다선택 다차원 배낭 문제(multiple-choice multidimensional knapsack problem, MMKP)는 배낭 문제의 일종으로 배낭 문제 중에서도 가장 어려운 문제로 알려져 있다[1]. 중복 할당 문제[2], 자본 예산 편성 문제[3], 창고 관리 문제[4] 등 다양한 실세계 문제들이 다선택 다차원 배낭 문제로 모델링될 수 있다.

총 N개의 아이템들이 존재하며 이 아이템들은 n개의 서로 중첩되지 않는 그룹으로 나뉜다고 가정하자. 각 아이템은 고유의 값을 가지고 있으며 m개의 차원 별로 무게를 가지고 있다. 다선택 다차원 배낭 문제는 각 그룹 별로 단 하나의 아이템만 선택하여 값의 합계를 최대화하는 문제이다. 단, m개의 무게에 대한 차원 별로 선택된 아이템들의 무게 합은 주어진 최대 무게를 초과해서는 안된다.

다선택 다차원 배낭 문제를 수식으로 나타내면 식 (1)~(4)와 같다. G_i는 그룹 i에 포함된 아이템들의 집합을 의미하며, p_{ij}는 그룹 i에 포함된 아이템 j의 값을 의미한다. w_{ij}^k는 그룹 i에 포함된 아이템 j의 k 차원 무게를 의미하며, W^k는 k 차원의 최대 무게를 의미한다. x_{ij}는 그룹 i의 아이템 j를 선택하는 경우 1이 되며 선택하지 않을 경우 0이 되는 결정 변수이다. 따라서 목적 함수인 식 (1)은 선택된 아이템들의 값의 합을 최대화함을 나타낸다. 식 (2)는 m개의 각 무게 별로 선택된 아이템들의 무게 합이 주어진 최대 무게를 넘지 않아야 함을 나타내는 제약 조건이며, 식 (3)은 각 그룹 별로 단 하나의 아이템만 선택되어야 함을 나타내는 제약조건이다. 마지막으로 식 (4)에 의해 x_{ij}의 값은 0 또는 1만 가능하게 된다.

본 논문에서는 다선택 다차원 배낭 문제를 해결하기 위한 정수 계획법 기반 지역 탐색의 적용 방안을 제시하고 있다. 정수 계획법 기반 지역 탐색은 이웃해를 생성하기 위해 정수 계획법을 사용하고 있기 때문에 대상 문제가 선형식, 즉 결정 변수에 대한 1차식으로 표현이 가능해야 한다. 다선택 다차원 배낭 문제는 식 (1)~(4)와 같이 선형식으로 표현되므로 정수 계획법 기반 지역 탐색의 적용이 가능하다.

• First Author: Junha Hwang, Corresponding Author: Junha Hwang
*Junha Hwang (jhhwang@kumoh.ac.kr), Dept. of Computer Engineering, Kumoh National Institute of Technology
• Received: 2018. 09. 17, Revised: 2018. 10. 28, Accepted: 2018. 11. 12.
• This research was supported by Kumoh National Institute of Technology(2017-104-062).

$$\text{Maximize} \quad z = \sum_{i=1}^{n} \sum_{j \in G_i} p_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \sum_{j \in G_i} w_{ij}^{k} x_{ij} \leq W^{k}, k = 1, \ldots, m \tag{2}$$

$$\sum_{j \in G_i} x_{ij} = 1, k = 1, \ldots, n \tag{3}$$

$$x_{ij} \in \{0, 1\}, i = 1, \ldots, n, j \in G_i \tag{4}$$

정수 계획법 기반 지역 탐색은 현재해를 기반으로 이웃해를 생성하는 과정을 반복 수행하되 이웃해를 생성할 때 전체가 아닌 일부 아이템들을 대상으로 정수 계획법을 사용한다. 선형 최적화 문제에 대해 적용이 가능한 정수 계획법은 최적해의 도출을 보장하는 알고리즘으로 소규모 최적화 문제에 대한 최적해 도출에 매우 효과적인 것으로 알려져 있다. 정수 계획법 기반 지역 탐색 적용 시 핵심 요소는 이웃해 생성을 위해 정수 계획법에 참여할 아이템들을 어떻게 결정하느냐 하는 것이다. 본 논문에서는 총 3가지 방법을 제시하고 비교 검토한다. 첫 번째는 선형 계획법을 이용하는 방법이고 두 번째는 값/무게 비를 이용하는 방법이며, 마지막으로는 무작위 선택 방법이다.

실험 결과, 선형 계획법을 이용한 방법이 가장 우수한 것으로 나타났다. 더군다나 해당 방법을 통해 37개의 테스트 데이터 중 23개 테스트 데이터에서 기존의 가장 좋은 해와 같거나 더 좋은 해를 발견할 수 있었으며, 13개의 테스트 데이터에 대해서는 더 좋은 해를 발견할 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 정수 계획법 기반 지역 탐색과 다선택 다차원 배낭 문제와 관련된 기존 연구에 대해 살펴보고, 3장에서는 정수 계획법 기반 지역 탐색 적용을 위한 아이템 선택 방법에 대해 자세히 설명한다. 4장에서는 실험 결과를 제시하고 분석하며, 마지막으로 5장에서 결론 및 향후 과제에 대해 기술한다.

II. Related Work

1. Multi-choice Multidimensional Knapsack Problem

지금까지 다선택 다차원 배낭 문제를 해결하기 위해 제시된 많은 기법들은 [표 1]에 제시된 37개의 실험 데이터를 대상으로 성능을 검증해 왔다. 실험 데이터는 크게 2개의 그룹으로 나뉜다. 첫 번째 그룹은 I07~I13과 INST01~INST20을 포함하는 총 27개의 데이터이다. I07~I13은 기존 연구 [5]에서 제시되었고, INST01~INST20은 [6]에서 제시된 데이터로서 [5]와 동일한 방식으로 만들어졌다. 두 번째 그룹은 INST21~INST30을 포함하는 10개 데이터로서 [7]에서 처음 제시되었다. [표 1]의 모든 데이터들은 http://www.info.univ-angers.fr/pub/hao/mmkp.html 사이트를 통해 다운로드가 가능하다.

첫 번째 그룹의 데이터들은 각각 그룹 별로 동일한 개수의 아이템들을 포함하고 있다. 예를 들어 I07~INST12는 그룹 별로 10개의 아이템들을 포함한다. 반면에 두 번째 그룹의 데이터들은 각 그룹 별로 다양한 개수의 아이템들을 포함하고 있다. 예를 들어 INST21의 경우 각 그룹 별로 10개 이내의 아이템들을 포함한다. 즉, 어떤 그룹에는 3개의 아이템들이 포함되어 있고 어떤 그룹에는 8개의 아이템들이, 어떤 그룹에는 10개의 아이템들이 포함되기도 한다. [표 1]에는 각 데이터에 대해 그룹 개수(n), 각 그룹 별 아이템 개수($|G_i|$), 무게의 차원 개수(m)를 표시하였다. 아울러 현재까지의 많은 연구들 중 가장 우수한 결과가 도출된 것으로 인정받고 있는 [7], [8], [9]의 연구를 토대로 한 가장 좋은 목적함수 값(BKS, Best Known Solution)과 관련 문헌(reference)을 함께 표기하였다.

Table 1. MMKP Data Instances

data	n	$\|G_i\|$	m	BKS	reference
I07	100	10	10	24595	[8][9]
I08	150	10	10	36894	[7][8]
I09	200	10	10	49187	[9]
I10	250	10	10	61479	[9]
I11	300	10	10	73791	[8][9]
I12	350	10	10	86095	[8]
I13	400	10	10	98445	[7][8]
INST01	50	10	10	10738	[7][8][9]
INST02	50	10	10	13598	[7][8][9]
INST03	60	10	10	10955	[7]
INST04	70	10	10	14456	[8][9]
INST05	75	10	10	17061	[8][9]
INST06	75	10	10	16843	[9]
INST07	80	10	10	16444	[8]
INST08	80	10	10	17521	[9]
INST09	80	10	10	17763	[8][9]
INST10	90	10	10	19320	[7][8][9]
INST11	90	10	10	19449	[8]
INST12	100	10	10	21742	[9]
INST13	100	30	10	21580	[7][8][9]
INST14	150	30	10	32875	[8]
INST15	180	30	10	39163	[8][9]
INST16	200	30	10	43367	[8][9]
INST17	250	30	10	54363	[8][9]
INST18	280	20	10	60469	[9]
INST19	300	20	10	64933	[9]
INST20	350	20	10	75616	[8][9]
INST21	100	10	10	44284	[9]
INST22	100	10	20	41976	[7][8]
INST23	100	10	30	42584	[8]
INST24	100	10	40	41998	[9]
INST25	100	20	10	44159	[8]
INST26	100	20	20	44879	[8]
INST27	200	10	10	87634	[9]
INST28	300	10	10	134654	[9]
INST29	400	10	10	179228	[8]
INST30	500	10	10	214242	[9]

기존 연구 [7]에서는 Pareto algebra 원리에 기반한 휴리스틱 알고리즘을 제시하였으며, 총 8개의 데이터에 있어서 가장 좋은 결과를 보였다. 다만 이후의 연구인 [8]과 [9]에 비해서는 성능이 뒤떨어지는 편이다. [8]에서는 선형 계획법을 활용한 “reduce and solve” 휴리스틱을 제시하였으며, 총 24개의 데이터에 있어서 가장 좋은 결과를 보이는 등 매우 탁월한 성

능을 발휘한 것으로 인정받아 왔다. 보다 최근에 발표된 [9]에서는 정수 계획법 자체의 성능을 향상시키기 위한 방안으로 pseudo-gap의 개념을 제시하였으며, 이를 통해 총 25개의 데이터에 있어서 가장 좋은 결과를 도출하는 등 전반적으로 [8]을 능가하는 성능을 보였다.

이상의 기존 연구들의 공통점은 매번 실행 시마다 항상 동일한 결과가 도출된다는 것이다. 이에 따라 실험 결과 또한 단 한 번의 실험을 통해 도출된 결과가 제시되었다. 본 연구에서는 정수 계획법 기반 지역 탐색 내에서 이웃해 생성을 위한 아이템 선택 시 선형 계획법을 적극적으로 활용하되 무작위적 요소를 추가하여 매번 다른 결과가 도출될 수 있도록 하였다. 비록 평균적으로는 최선의 결과가 도출되지 않는다 하더라도 반복적인 실행 또는 병렬적인 실행을 통해 지금까지 도출된 가장 좋은 해와 같거나 더 좋은 해의 도출 가능성을 높일 수 있을 것으로 기대된다. 본 연구에서는 [표 1]의 데이터를 활용하여 제시한 기법의 성능을 검증한다.

2. Integer Programming-based Local Search

주어진 조합 최적화 문제의 목적 함수와 제약 조건이 선형적으로 표현될 때, 이를 해결하기 위해 정수 계획법(integer programming)의 적용이 가능하다. 정수 계획법은 트리 탐색 기법의 일종으로 분지 한계법(branch and bound)을 기반으로 하며, 정수 계획법 자체의 성능을 향상시키기 위한 많은 연구가 진행되어 왔다[10]. 정수 계획법의 경우 최적해의 도출이 보장되지만 대상 문제의 규모가 커질 경우 최적해를 도출하기까지 많은 시간과 메모리를 요구하게 된다.

```
Algorithm IPbLS
    X : Current solution vector.
    k : The number of selected variables to be changed.
    IP : An integer programming solver.
Begin
1:  X = Make an initial solution using a heuristic method
2:  While stopping condition is not met Do
3:      Select k variables from X
4:      Add objective and all constraints to IP
5:          • Fix values of unselected variables
6:      X = Make a neighbor solution by solving IP
7:  End While
8:  return X
End Begin
```

Fig. 1. General Integer Programming-based Local Search

정수 계획법 기반 지역 탐색(integer programming-based local search, IPbLS)은 지역 탐색(local search)의 일종으로 기본적인 알고리즘은 [그림 1]과 같이 매우 단순하다[11]. 1라인, 먼저 휴리스틱 탐색 기법을 사용하여 초기해를 생성하여 현재해를 만든다. 2라인~7라인, 그리고 다음 과정이 반복적으로 수행된다. 3라인, 현재해를 기반으로 정수 계획법에 참여할 변수를 선택한다. 이 과정은 일종의 문제 축소 과정으로 볼 수 있다. 4라인~6라인, 그러고 나서 선택된 변수들을 대상으로 정수 계획법을 적용하여 새로운 현재해를 만들게 된다. 5라인, 이때 선택되지 않은 변수들의 값을 고정 또는 삭제함으로써 탐색 공간을 줄일 수 있다. IPbLS의 성공 여부는 다음 정수 계획법에 참여할 변수를 얼마나 잘 선택하느냐에 달려 있다. 기본적으로 변수 선택 결과, 최적의 조합을 포함할 수 있어야 한다. 그렇다고 k 값을 크게 설정하면 최적의 조합을 포함할 가능성은 높아지지만 정수 계획법을 통해 최적해를 도출하기까지 너무 많은 시간이 소요될 수 있다. 결론적으로 얘기하면 가급적 적은 변수를 선택하여 최적해의 조합을 포함할 수 있어야 한다.

일반적으로 지역 탐색은 1개 또는 2개 정도의 변수값을 변경하여 이웃해를 생성하게 된다. 이로 인해 지역 최적해에 머물게 되는 지역 최적화 현상이 나타난다. 그러나, IPbLS는 정수 계획법을 활용하여 한 번에 훨씬 많은 변수들의 값을 변경해 봄으로써 지역 최적화 문제를 극복하게 된다. IPbLS는 지금까지 집합 커버링 문제, 다차원 배낭 문제 등 다양한 문제의 해결을 위해 적용되어 왔으며, 다른 휴리스틱 탐색 기법들에 비해 성능이 우수한 것으로 확인되었다[11, 12]. 본 논문에서는 IPbLS를 활용하여 다선택 다차원 배낭 문제를 해결한다.

III. IPbLS for MMKP

본 연구에서는 정수 계획법 기반 지역 탐색을 활용하여 다선택 다차원 배낭 문제를 해결하기 위해 기본적으로 [그림 1]의 구조를 따른다. 먼저 초기해 생성을 위해 정수 계획법을 사용한다. 즉, 모든 아이템들을 대상으로 정수 계획법을 적용하는 것이다. 단, 수행 시간은 300초로 설정하였다. 다선택 다차원 배낭 문제의 해결을 위해 정수 계획법을 적용하는 경우 탐색 초기에 어느 정도 좋은 해가 도출되는 경향이 있다. 따라서 다른 휴리스틱 탐색 기법들보다 짧은 시간 내에 더 좋은 해의 도출이 가능하다.

초기해가 도출되면 이를 토대로 다음 정수 계획법에 참여할 아이템들을 선택하고 정수 계획법을 적용하여 새로운 해를 도출하는 과정을 반복 수행한다. 결국 핵심 사항은 아이템을 선택하는 방법으로 본 연구에서 제안한 아이템 선택 방법은 총 3가지이다.

① 선형 계획법을 활용한 방법(LP, Linear Programming)

정수 계획법 문제에서 결정 변수의 타입이 정수라는 제약 조건을 실수라는 제약조건으로 완화하면 선형 계획법 문제가 된다. 선형 계획법은 심플렉스법(simplex method)을 사용하여 매우 빠른 시간 내에 최적해의 도출이 가능하다. 정수 계획법 또한 분지 한계법을 통한 문제 해결 시 내부적으로 선형 계획법을 활용한다. 선형 계획법을 통해 도출된 최적해의 결정 변수들의 값은 비록 0 또는 1이 아닌 0과 1 사이의 실수값을 가지게 되지만 그 값 자체가 정수 계획법의 최적해에 대한 매우 중요한 실마리를 제공해 준다.

[표 2]는 그룹의 개수가 5이고 무게 차원의 수가 2이며, 각

그룹에는 2개의 아이템을 포함되어 있는 다선택 다차원 배낭 문제의 예를 보인 것이다. 아울러 선형 계획법의 결과(L)와 최적해(Optimal)를 함께 표기하였다. L과 Optimal을 비교해 보면 그룹 3을 제외하면 각 그룹 별로 L의 값이 1이거나 더 큰 아이템이 최종적으로 선택된 것을 알 수 있다. 이와 같이 선형 계획법의 결과는 최적해 도출에 결정적인 역할을 할 수 있을 것으로 판단된다.

Table 2. An Example of MMKP

Group No.	Item No.	p_{ij}	w_{ij}^k		L	Optimal x_{ij}
			1	2		
1	1	3	2	6	0	0
	2	2	1	4	1	1
2	1	5	4	3	0.36	0
	2	4	1	7	0.64	1
3	1	3	2	4	1	0
	2	3	4	2	0	1
4	1	4	3	4	0	0
	2	5	2	5	1	1
5	1	3	2	1	0.55	1
	2	5	4	2	0.54	0
W^k			10	20		

예를 들어, I07 데이터의 경우 선형 계획법을 통해 도출된 최적해에서 0을 초과하는 값을 가진 결정 변수는 총 1000개 중 109개이다. 그리고 I07 데이터에 대해 현재까지 도출된 가장 좋은 해의 목적함수 값은 24595이고, 이때 문제에서 주어진 제약조건에 따라 결정 변수 1000개 중 100개가 선택된 상태이다. 그런데, 선택된 100개의 결정 변수들 중 93개가 선형 계획법을 통해 도출된 최적해에서 0을 초과하는 값을 가진 결정 변수들이었다. 다시 말하면 선형 계획법을 통해 0을 초과하는 값을 가진 결정 변수 109개를 선택하고, 거기에 7개의 결정 변수만 잘 선택하면 현재까지 도출된 가장 좋은 해를 쉽게 도출할 수 있다는 것이다. 물론 실제로는 7개의 결정 변수를 쉽게 선택할 수 있는 것은 아니다. 여기서 시사하는 바는 선형 계획법의 결과가 원문제의 최종 결과와 밀접한 관련이 있다는 것이다. 첫 번째 아이템 선택 방법에서는 이와 같은 선형 계획법의 특성을 활용한다.

그런데 최적해 또는 준최적해의 도출을 위해 반드시 포함되어야 하는 아이템들 중 일부는 선형 계획법만으로는 선택이 되지 않는 상황이 발생할 수 있다. 이와 같은 경우를 방지하기 위해 일부 아이템들은 무작위로 선택하는 방법을 추가하였다. 본 연구에서는 80%는 선형 계획법을 활용하여 선택하고 나머지 20%는 무작위로 선택하였다.

이상을 정리하면 선형 계획법을 활용한 아이템 선택 방법은 [그림 2]와 같다. 특정 아이템의 선택 여부는 J 배열로 표현되며 선택된 아이템들은 1, 선택되지 않은 아이템들은 0의 값을 가진다. 1라인, 모든 아이템들에 대해 선택되지 않은 것으로 설정한다. 2라인, 기본적으로 현재해에서 선택된 아이템들을 포함한다. 3라인, 원문제에 대한 선형 계획법 적용 결과 0을 초과하는 값을 가진 아이템들을 포함한다. 4~9라인, 다음으로는 (k × 0.8)개의 아이템들이 선택될 때까지 또 다시 선형 계획법을 적용하고, 0을 초과하는 아이템들을 포함하는 과정을 반복 수행한다. 6라인, 이때 유효한 해가 도출될 수 있도록 각 그룹 별로 결정 변수 값들의 합이 1이라는 제약조건을 완화하여 0과 1 사이의 값을 갖도록 하였고, 7라인, 기존에 선택된 아이템들은 다시 선택되지 않도록 하는 제약 조건을 추가하였다. 10라인, 마지막으로 나머지 아이템들은 무작위로 선택한다.

```
Algorithm SelectLP
Input
    k : The number of items to be selected.
    X : Current solution vector.
    L : Linear programming value vector for the original problem.
    G_i : The item set of i-th group.
    LP_ratio(0.8) : The ratio of items to be selected by LP.
Output
    J : Selected item vector.
Begin
1:  J[i] = 0
2:  J[i] = 1 if X[i] == 1
3:  J[i] = 1 if L[i] > 0
4:  While Sum(J) < (k × LP_ratio) Do
5:      L = Run Linear Programming with
6:          • constraint 0 <= Sum(L[G_i]) <= 1
7:          • constraint L[i] = 0 if J[i] == 1
8:      J[i] = 1 if L[i] > 0
9:  End While
10: J[i] = 1 if i is included in (k - Sum(J)) selected items randomly
11: return J
End Begin
```

Fig. 2. Selection Method using Linear Programming

② 값/무게 비를 활용한 방법(P/W, Profit/Weight)

다선택 다차원 배낭 문제에서 목적함수 값을 향상시키고 제약조건을 준수하기 위해 값이 크고 무게가 작은 아이템이 중요한 역할을 할 것으로 추정할 수 있다. 두 번째 아이템 선택 방법에서는 값과 무게의 비를 활용하여 이 값이 큰 아이템들이 많이 선택되도록 한다. 여기서 무게란 m차원의 무게의 합을 의미한다. 그렇다 하더라도 원문제의 선형 계획법 결과는 매우 중요하므로 [그림 3]에서 보는 바와 같이 첫 번째 방법인 선형 계획법을 활용한 방법과 마찬가지로 2라인, 현재해에서 선택된 아이템들을 포함하는 동시에 3라인, 원문제에 대한 선형 계획법 적용 결과 0을 초과하는 값을 가진 아이템들을 포함한다. 그리고 4라인, 나머지 아이템들은 값/무게 비에 따라 확률적으로 선택한다.

③ 무작위 선택 방법(RANDOM)

마지막으로 무작위로 선택하는 방법을 생각해 볼 수 있다. 그런데 실험 결과에 의하면 현재해에서 선택된 아이템들과 무작위로 선택된 아이템들만 대상으로 하는 경우 대부분의 데이터에 있어서 초기해 생성 이후 개선이 전혀 되지 않을 정도로 성능이 좋지 않음을 확인하였다. 따라서 무작위 선택 방법의 경우에도 첫 번째와 두 번째 선택 방법과 마찬가지로 현재해에서 선택된 아이템들과 원문제에 대한 선형 계획법 적용 결과 0을 초과하는 값을 가진 아이템들을 포함한다.

Algorithm SelectPW
Input
k : The number of items to be selected.
X : Current solution vector.
L : Linear programming value vector for the original problem.
PW : Price/∑weight ratio vector.
Output
J : Selected item vector.
Begin
1: $J[i]$ = 0
2: $J[i]$ = 1 if $X[i]$ == 1
3: $J[i]$ = 1 if $L[i]$ > 0
4: $J[i]$ = 1 if i is included in (k − Sum(J)) selected items
stochastically with the probability $PW[i]$/Sum(PW)
5: **return** J
End Begin

Fig. 3. Selection Method using Profit/Weight Ratio

IV. Experimental Results

1. Comparison of Selection Methods

본 연구의 모든 실험은 Intel Core i7-4790 CPU 3.6GHz, 4GB RAM PC 및 Windows 7 64비트 운영체제 상에서 수행되었다. 그리고 정수 계획법 기반 지역 탐색 프로그램은 정수 계획법 개발 도구인 IBM ILOG CPLEX 12.7.1 및 DOcplex를 활용하여 Python 언어로 구현하였다[13].

[표 3]은 각 데이터 및 각 아이템 선택 방법 별로 선택 비율에 따른 최종 결과를 나타낸 것이다. 예를 들어, I07의 경우 모든 아이템의 개수가 1000개이므로 선택 비율이 0.2이면 다음 정수 계획법 수행 시 대상 아이템으로 200개가 선택되며, 선택 비율이 0.9이면 900개가 선택된다. 각 수치는 5회 실험 결과의 평균값이며, 각 실험 당 수행 시간은 1시간으로 설정하였다. 각 데이터 별로 가장 좋은 결과에 대해서는 빨간색 및 볼드체로 표시하였다.

INST01과 INST02의 경우 모든 경우에 있어서 최적해의 도출이 가능하다. 이외의 35개 데이터 중 선형 계획법을 활용한 방법(LP)에 의해 가장 좋은 해가 도출된 데이터는 총 29개이며, 값/무게 비를 활용하는 방법(P/W)에 의해 가장 좋은 해가 도출된 데이터는 7개이다. 그리고 무작위 선택 방법에 의해 가장 좋은 해가 도출된 데이터는 3개이다. 이를 통해 선형 계획법을 활용한 아이템 선택 방법의 성능이 가장 좋음을 확인할 수 있다.

선형 계획법을 활용한 방법만 고려해 볼 때 성능이 가장 좋은 선택 비율은 데이터마다 다르게 나타나고 있다. 참고로 해당 결과에 대해서는 이탤릭체로 표시하였다. INST01과 INST02를 제외하고 선택 비율이 0.5일 때 가장 많은 데이터인 10개의 데이터에서 가장 좋은 성능을 발휘하고 있으며, 다음으로는 선택 비율이 0.2와 0.4인 경우 9개, 0.3인 경우 8개의 데이터에서 가장 좋은 성능을 발휘하고 있다. 이는 비록 데이터마다 다르긴 하지만 50% 이하의 비교적 적은 아이템들만 포함하더라도 매우 좋은 조합의 아이템들을 포함할 가능성이 높은 것으로 해석된다. 본 실험 결과를 통해 좋은 조합의 아이템들을 선택하는데 선형 계획법의 적용 결과가 중요한 역할을 담당하고 있음을 알 수 있다.

[그림 4]는 각 선택 비율 별로 모든 데이터에 대한 합계를 그래프로 나타낸 것이다. 즉, [표 3]의 “Sum”에 대한 그래프를 의미한다. 이를 통해 각 선택 방법 별로 선택 비율에 대한 성능의 전반적인 변화를 보다 명확히 확인할 수 있다. 먼저 선택 비율이 0.6 이하인 경우 LP의 성능이 다른 방법에 비해 월등히 우수하며 RANDOM의 성능이 가장 좋지 않음을 확인할 수 있다. 선택 비율이 0.7 이상인 경우 3가지 방법에 있어서 큰 차이가 없는데, 이는 어떤 방법을 사용하든 매우 많은 아이템들이 선택되므로 결국 최적 조합을 포함할 가능성이 높아지기 때문인 것으로 분석된다. LP의 경우 선택 비율이 0.5인 경우 최적 조합의 아이템들을 충분히 선택할 수 있는 것으로 보이며, 0.6 이후로는 최적 조합의 아이템들이 선택된다 하더라도 정수 계획법의 대상 아이템 자체가 많아지므로 전반적인 성능은 저하되는 것으로 판단된다. [그림 4]를 통해 아이템 선택 시 선형 계획법의 유용성을 다시 한 번 확인할 수 있다.

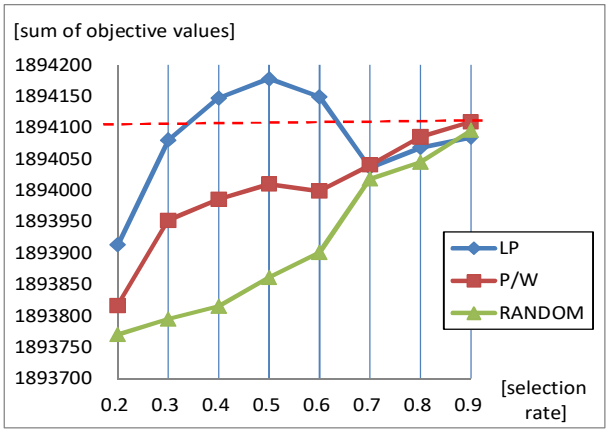

Fig. 4. Sum Graph of Objective Values for Selection Rate

Table 3. Experimental Results for Selection Methods

Data	Method	Selection Rate							
		0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
I07	**LP**	24591.6	24592.2	24593.4	***24593.6***	24592.0	24593.0	24592.0	24592.8
	P/W	24592.0	24592.0	24592.0	24592.8	24592.4	24592.0	24593.0	24593.0
	RANDOM	24590.4	24590.8	24590.6	24592.0	24591.0	24591.4	24592.0	24592.8
I08	**LP**	36890.8	***36892.4***	36890.8	36890.4	36890.6	***36892.4***	36890.0	36890.6
	P/W	36888.0	36887.6	36887.8	36887.2	36889.6	36890.0	36891.2	36890.2
	RANDOM	36886.0	36886.2	36887.8	36887.2	36886.8	36888.0	36887.8	36890.8

Data	Method	Selection Rate							
		0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
I9	LP	49179.6	*49183.0*	*49183.0*	49182.6	*49183.0*	49181.8	49182.2	49182.2
	P/W	49175.4	49175.8	49178.8	49180.2	49179.6	49182.8	49181.0	**49183.2**
	RANDOM	49175.0	49174.8	49176.0	49178.8	49177.4	49178.8	49179.8	49180.0
I10	**LP**	61476.2	61478.4	61477.4	61478.0	***61479.0***	61478.0	61478.6	61478.4
	P/W	61476.8	61475.4	61476.8	61477.6	61475.2	61478.8	61477.4	61478.6
	RANDOM	61475.2	61475.2	61475.2	61476.2	61476.2	61475.4	61476.2	61476.8
I11	**LP**	73786.4	***73789.6***	73788.6	73789.0	73788.8	73788.2	73787.6	73785.6
	P/W	73785.6	73785.8	73788.0	73786.6	73786.2	73787.0	73787.4	73786.8
	RANDOM	73784.8	73784.8	73786.2	73785.8	73787.4	73785.4	73787.0	73787.4
I12	**LP**	86086.2	86091.6	86091.2	***86093.6***	86091.8	86091.4	86090.0	86090.8
	P/W	86087.2	86086.2	86087.6	86087.2	86090.2	86092.0	86092.0	86090.0
	RANDOM	86083.6	86082.4	86083.4	86084.6	86085.2	86089.2	86092.6	86089.6
I13	**LP**	98437.2	***98441.0***	98440.2	98440.8	98437.6	98440.4	98440.2	98436.8
	P/W	98434.6	98436.2	98438.4	98436.2	98439.8	98439.4	98439.4	98437.4
	RANDOM	98431.8	98432.2	98433.0	98433.4	98436.0	98439.0	98439.4	98438.8
INST01	**LP**	**10738.0**	**10738.0**	**10738.0**	**10738.0**	**10738.0**	**10738.0**	**10738.0**	**10738.0**
	P/W	**10738.0**	**10738.0**	**10738.0**	**10738.0**	**10738.0**	**10738.0**	**10738.0**	**10738.0**
	RANDOM	**10738.0**	**10738.0**	**10738.0**	**10738.0**	**10738.0**	**10738.0**	**10738.0**	**10738.0**
INST02	**LP**	**13598.0**	**13598.0**	**13598.0**	**13598.0**	**13598.0**	**13598.0**	**13598.0**	**13598.0**
	P/W	**13598.0**	**13598.0**	**13598.0**	**13598.0**	**13598.0**	**13598.0**	**13598.0**	**13598.0**
	RANDOM	**13598.0**	**13598.0**	**13598.0**	**13598.0**	**13598.0**	**13598.0**	**13598.0**	**13598.0**
INST03	**LP**	10942.6	10948.8	10950.4	10949.2	10949.0	10949.2	10948.6	***10952.6***
	P/W	10944.2	10943.8	10948.2	10946.2	10945.0	10945.0	10952.4	10949.2
	RANDOM	10942.6	10942.6	10943.2	10945.6	10942.6	10947.0	10945.8	10949.8
INST04	**LP**	14443.4	14447.4	14450.4	***14453.6***	14450.4	14452.8	14448.2	14449.4
	P/W	14443.4	14443.4	14444.2	14443.8	14447.2	14451.2	14451.4	14449.0
	RANDOM	14443.4	14443.4	14444.0	14444.0	14445.0	14446.6	14446.6	14450.2
INST05	**LP**	17055.4	17059.2	***17060.4***	17059.2	17059.0	17058.8	17058.4	17057.2
	P/W	17056.0	17055.0	17057.0	17056.6	17058.6	17058.0	17057.8	17057.6
	RANDOM	17055.0	17055.0	17056.2	17056.6	17058.2	17057.8	17058.2	17058.0
INST06	**LP**	16829.6	***16838.4***	***16838.4***	16836.6	16837.4	16836.2	16835.4	16833.8
	P/W	16827.8	16829.4	16827.4	16831.2	16834.4	16833.4	16833.2	16833.6
	RANDOM	16824.6	16826.2	16827.4	16825.6	16827.4	16832.8	16835.8	16832.6
INST07	LP	16440.0	16440.4	*16440.8*	16440.0	16440.8	16440.0	16440.0	16440.2
	P/W	16440.0	16440.0	16440.0	16440.6	16440.2	16440.4	16440.0	**16441.2**
	RANDOM	16440.0	16440.0	16440.4	16440.2	16440.0	16440.0	16440.0	16440.0
INST08	**LP**	17508.2	17515.8	17513.6	17511.8	17515.4	17510.2	***17517.0***	17510.2
	P/W	17506.2	17507.4	17508.4	17508.6	17512.2	17513.8	17512.4	17511.6
	RANDOM	17505.8	17505.8	17507.6	17508.4	17508.6	17510.6	17510.0	17510.8
INST09	**LP**	17758.2	17762.2	***17763.2***	17762.2	17762.0	17762.8	17761.8	17762.0
	P/W	17759.0	17756.8	17761.2	17762.6	17762.2	17761.8	17760.6	17762.4
	RANDOM	17757.2	17754.2	17757.8	17758.8	17755.8	17756.4	17757.4	17762.0
INST10	**LP**	19310.0	19319.2	19317.8	***19319.6***	19317.2	19316.0	19316.0	19316.0
	P/W	19311.0	19309.2	19312.0	19314.0	19316.0	19315.2	19314.4	19316.0
	RANDOM	19304.4	19307.2	19305.0	19307.6	19307.4	19313.4	19312.2	19313.0
INST11	LP	19434.0	19444.4	19443.4	*19444.8*	19442.2	19442.2	19440.0	19442.2
	P/W	19432.4	19434.6	19433.2	19436.2	19437.2	19442.0	19439.8	***19446.0***
	RANDOM	19434.2	19429.4	19429.8	19431.0	19433.0	19436.0	19439.6	19439.0
INST12	**LP**	21728.6	21739.4	***21740.0***	21739.6	21739.0	21739.6	21737.6	21739.2
	P/W	21732.0	21730.0	21733.6	21735.6	21736.8	21736.4	21737.8	21735.2
	RANDOM	21726.2	21728.2	21728.8	21727.8	21730.4	21738.2	21736.6	21739.0
INST13	**LP**	***21577.4***	21577.2	***21577.4***	21577.0	21577.0	21577.2	21577.0	21577.0
	P/W	21577.0	21577.2	21577.0	21577.2	***21577.4***	21577.2	21577.2	21577.2
	RANDOM	21576.2	21577.0	21576.2	21576.2	21576.2	21576.6	***21577.4***	21577.2
INST14	LP	*32873.2*	*32873.2*	32873.0	*32873.2*	32873.0	32873.0	*32873.2*	32873.0
	P/W	32873.4	32873.0	32873.0	32873.4	32873.6	32873.4	32873.4	32873.0
	RANDOM	32872.4	32872.8	32872.8	32872.4	32872.8	32872.6	32873.0	**32873.8**
INST15	**LP**	***39164.4***	39163.4	39163.0	39163.0	39162.2	39162.8	39161.8	39162.6
	P/W	39158.6	39159.6	39162.0	39162.6	39163.2	39163.2	39163.0	39162.4
	RANDOM	39158.2	39158.2	39158.0	39158.8	39159.0	39159.4	39160.2	39161.0
INST16	**LP**	***43365.8***	43365.6	43364.8	43365.2	43364.6	43364.8	43364.8	43365.2
	P/W	43362.4	43362.8	43363.4	43365.0	43365.2	43363.8	43364.2	43364.8
	RANDOM	43362.2	43361.8	43361.8	43362.0	43363.4	43364.6	43364.4	43364.8

Data	Method	Selection Rate							
		0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
INST17	**LP**	***54360.8***	54360.0	54360.2	54360.0	54360.6	54360.6	54360.2	54360.6
	P/W	54358.0	54358.2	54358.0	54360.0	54360.2	54359.6	54360.2	54359.4
	RANDOM	54358.0	54357.6	54357.6	54357.8	54358.4	54358.6	54359.2	54359.8
INST18	**LP**	60466.6	***60467.0***	60466.6	60466.8	60466.4	60466.0	***60467.0***	60466.8
	P/W	60463.8	60463.2	60463.4	60465.0	60465.0	60466.8	60466.0	60465.4
	RANDOM	60461.4	60461.6	60462.0	60463.8	60463.8	60465.0	60465.2	60465.6
INST19	LP	*64931.4*	*64931.4*	64930.6	64931.2	64930.4	64930.4	64930.8	*64931.4*
	P/W	64928.4	64928.4	64930.2	**64932.0**	64930.2	64931.0	64930.2	64931.0
	RANDOM	64927.0	64927.2	64927.8	64927.8	64928.6	64929.6	64930.2	64930.8
INST20	**LP**	***75617.0***	75615.0	75614.4	75615.0	75615.0	75614.0	75614.2	75614.2
	P/W	75613.8	75613.4	75613.4	75613.8	75614.4	75615.0	75614.4	75614.6
	RANDOM	75612.8	75612.8	75613.2	75612.8	75614.2	75613.8	75614.0	75615.2
INST21	**LP**	44280.0	44280.0	***44280.4***	44280.0	***44280.4***	44280.0	44280.0	44280.0
	P/W	44280.0	44280.0	44280.0	44280.0	44280.0	44280.4	44280.0	**44280.4**
	RANDOM	44280.0	44280.0	44280.0	**44280.4**	44280.0	**44280.4**	44278.4	44280.0
INST22	**LP**	41942.8	41954.8	41962.0	***41978.8***	41973.2	41960.0	41960.0	41964.4
	P/W	41936.8	41950.0	41962.0	41961.2	41950.0	41962.8	41966.8	41970.0
	RANDOM	41936.8	41942.0	41937.6	41946.0	41944.4	41962.0	41962.4	41968.8
INST23	**LP**	42510.4	42540.0	42571.2	***42588.8***	42580.0	42537.6	42540.4	42551.6
	P/W	42509.6	42542.0	42552.0	42520.8	42529.2	42544.4	42555.2	42551.6
	RANDOM	42512.8	42523.2	42519.6	42514.4	42538.8	42540.8	42554.4	42554.4
INST24	**LP**	41878.0	41915.6	***41948.4***	41923.2	41922.8	41920.4	41935.2	41937.6
	P/W	41870.0	41942.8	41927.2	41944.8	41897.2	41890.0	41916.0	41930.4
	RANDOM	41868.8	41869.6	41875.6	41883.2	41888.8	41927.2	41916.8	41923.2
INST25	**LP**	***44151.8***	44150.8	44147.6	44143.6	44144.6	44149.4	44150.0	44150.4
	P/W	44139.2	44139.6	44140.4	44148.6	44148.6	44149.6	44150.8	44151.4
	RANDOM	44138.6	44142.0	44143.0	44144.6	44142.6	44144.0	44147.4	44151.4
INST26	**LP**	***44878.0***	44870.6	44871.2	44873.8	44872.0	44869.6	44871.4	44870.8
	P/W	44859.4	44853.4	44864.6	44861.4	44870.2	44875.4	44871.6	44876.4
	RANDOM	44858.8	44859.6	44860.8	44862.4	44865.0	44874.0	44870.0	44869.8
INST27	LP	87619.2	87620.0	87620.0	87616.8	*87624.0*	87618.0	87619.6	87617.6
	P/W	87616.0	87618.8	87622.0	87618.4	87622.0	87620.4	87621.2	**87624.8**
	RANDOM	87618.8	87616.0	87617.6	87618.0	87621.6	87620.4	87624.4	87621.6
INST28	**LP**	134634.4	134639.2	134642.4	134647.2	***134648.4***	134630.4	134635.6	134638.4
	P/W	134630.4	134640.8	134629.2	134640.4	134638.4	134635.2	134638.0	134640.8
	RANDOM	134629.2	134630.8	134630.8	134632.8	134633.2	134636.4	134639.2	134645.2
INST29	**LP**	179221.6	179217.2	179218.8	***179226.4***	179223.2	179214.4	179221.2	179217.6
	P/W	179217.2	179220.8	179215.6	179221.2	179222.0	179223.2	179222.4	179223.2
	RANDOM	179214.4	179214.4	179216.4	179222.4	179216.8	179220.4	179222.0	179223.6
INST30	**LP**	214206.4	214220.0	214216.4	***214227.6***	214220.4	214198.8	214206.0	214209.2
	P/W	214195.2	214204.0	214201.6	214205.2	214213.2	214214.0	214217.6	214215.6
	RANDOM	214187.6	214194.0	214196.0	214205.6	214209.2	214210.4	214213.6	214222.8
Sum	**LP**	1893913.2	1894080.4	1894147.4	**1894178.2**	1894149.4	1894036.4	1894068.0	1894084.4
	P/W	1893816.8	1893952.6	1893985.6	1894010.2	1893998.8	1894040.6	1894085.4	1894109.4
	RANDOM	1893770.2	1893795.0	1893815.2	1893861.0	1893901.2	1894018.2	1894045.2	1894095.6

2. Comparison with Other Methods

[표 4]는 기존 연구들과의 비교 결과를 나타낸 것이다. IP는 정수 계획법만을 사용하여 1시간 동안 실행한 결과이다. 단, 대상 아이템이 많은 경우 과도한 메모리를 요구하는 정수 계획법의 특성으로 인해 다른 실험의 실행 환경과는 달리 8GB RAM PC에서 수행하였다. 참고로 4GB RAM PC에서 메모리 요구량이 적은 깊이우선 탐색 방식으로 정수 계획법을 수행한 경우 [표 4]의 IP 결과보다 전반적으로 좋지 않은 결과가 도출됨을 확인하였다. [표 4]에서 Shojaei는 기존 연구 [7]의 결과이고 Chen은 기존 연구 [8]의 결과이며, Gao는 기존 연구 [9]의 결과이다. IPbLS(LP)의 Average는 [표 3]에서 LP를 활용한 선택 방법 중 각 데이터 별로 가장 좋은 결과를 보인 선택 비율의 결과값을 나타낸 것이다. 그리고 Best는 해당 선택 비율에 대한 5회의 실험 중 가장 좋은 값을 의미한다. 참고로 몇몇 데이터에 있어서는 [표 4]와는 다른 선택 비율일 때 더 좋은 결과가 도출되기도 하였는데, 해당 데이터 및 목적함수 값은 다음과 같다. I10(61481), INST04(14457), INST06(16845), INST09(17767), INST10(19325), INST12(21745), INST16(43367), INST17(54463), INST18(60469), INST29(189232), INST30(214234). 각 데이터 별로 가장 좋은 결과에 대해서는 빨간색 및 볼드체로 표기하였다.

Shojaei, Chen, Gao의 경우 실행 환경 자체가 본 연구와 다

르기 때문에 공정한 비교가 어렵지만, 수행 시간 등에 있어서 동일한 조건으로 실험이 수행되었으므로 전반적인 성능을 비교하기에 충분할 것으로 판단된다. 다만 Shojaei의 경우 I07~INST20 데이터에 대한 수행 시간은 1200초로 설정하였다. 이 경우에도 10개의 프로세서를 포함한 병렬 컴퓨터를 활용하였기 때문에 컴퓨터 성능을 직접적으로 비교하기는 힘들다. 이외의 모든 실험에 대한 수행 시간은 본 연구와 마찬가지로 1시간으로 설정하였다.

Table 4. Comparison with Other Methods

Data	IP	Shojaei [7]	Chen [8]	Gao [9]	IPbLS(LP)	
					Average	Best
I07	24592	24592	**24595**	**24595**	24593.6	**24595**
I08	36894	36894	36894	36893	36892.4	***36895***
I09	49185	49185	49185	**49187**	49183.0	49186
I10	61474	61471	61478	61479	61479.0	***61480***
I11	73784	73784	73791	73791	73789.6	***73792***
I12	86090	86091	86095	86094	86093.6	***86097***
I13	98439	**98445**	**98445**	98443	98441.0	98444
INST01	**10738**	**10738**	**10738**	**10738**	**10738.0**	**10738**
INST02	**13598**	**13598**	**13598**	**13598**	**13598.0**	**13598**
INST03	10945	**10955**	10949	10952	10952.6	**10955**
INST04	14452	14452	**14456**	**14456**	14453.6	**14456**
INST05	17059	17059	17061	17061	17060.4	***17063***
INST06	16838	16835	16840	**16843**	16838.4	16842
INST07	16440	16440	**16444**	16442	16440.8	16442
INST08	17509	17511	17514	17521	17517.0	***17524***
INST09	17761	17761	17763	17763	17763.2	***17766***
INST10	19314	**19320**	**19320**	**19320**	19319.6	**19320**
INST11	19437	19446	**19449**	19446	19444.8	**19449**
INST12	21738	21738	21741	**21742**	21740.0	21741
INST13	21578	**21580**	**21580**	**21580**	21577.4	21578
INST14	32872	32874	**32875**	32873	32873.2	**32875**
INST15	39160	39162	39163	39163	39164.4	***39165***
INST16	43363	43366	**43367**	**43367**	43365.8	43366
INST17	54360	54361	**54363**	**54363**	54360.8	54362
INST18	60466	60467	60467	**60469**	60467.0	60467
INST19	64930	64932	64932	**64933**	64931.4	64932
INST20	75613	75615	75616	75616	75617.0	***75619***
INST21	44268	44270	44280	**44284**	44280.4	44282
INST22	41994	41976	41976	41964	41978.8	***42000***
INST23	42538	42562	42584	42536	42588.8	***42600***
INST24	41916	41918	41918	**41998**	41948.4	41972
INST25	44156	44156	**44159**	44156	44151.8	**44159**
INST26	44878	44869	44879	44869	44878.0	***44893***
INST27	87618	87616	87630	**87634**	87624.0	87630
INST28	134648	134634	134648	**134654**	134648.4	**134654**
INST29	179216	179206	179228	179222	179226.4	***179230***
INST30	214222	214198	214230	**214242**	214227.6	214232
Sum	1894083	1894077	1894251	1894287	1894178.2	**1894399**

[표 4]에서 IPbLS(LP)의 Average에 의하면 전체적으로 볼 때 Shojaei의 결과보다는 좋지만 Chen과 Gao의 결과에 비해서는 뒤떨어짐을 알 수 있다. 그러나 5회 실험 결과 중 가장 좋은 결과인 Best를 기준으로 비교하면 Shojaei, Chen, Gao의 모든 결과보다 월등히 좋은 결과가 도출됨을 확인할 수 있다. 먼저 전체 합계(Sum)를 보면 IPbLS(LP-Best)가 1894399로 Gao의 1894287, Chen의 1894251, Shojaei의 1894077, IP의 1894083에 비해 훨씬 좋음을 알 수 있다. 또한 가장 좋은 결과를 도출한 데이터 개수 측면에서도 IPbLS(LP-Best)의 우수함을 확인할 수 있다. 즉, 기존 연구들 중 가장 좋은 결과를 보인 Gao가 18개의 데이터에 있어서 가장 좋은 결과를 보인 반면에 IPbLS(LP-Best)는 23개의 데이터에 있어서 가장 좋은 결과를 보였으며, 그 중에서 13개 데이터에 있어서는 기존의 가장 좋은 해보다 더 좋은 해를 도출할 수 있었다. 참고로 해당 13개 데이터에 대한 결과는 이탤릭체로 표시하였다. 비록 평균적인 성능에 있어서는 기존의 우수 연구 결과에 비해 뒤떨어지지만 5회라는 비교적 적은 횟수의 반복 수행만으로 기존의 가장 좋은 결과보다 더 좋은 결과를 도출할 수 있었다.

V. Conclusions

본 논문에서는 다선택 다차원 배낭 문제를 해결하기 위한 방안으로 정수 계획법 기반 지역 탐색 내에서 다음 정수 계획법 실행을 위한 아이템을 선택할 때 선형 계획법을 활용하는 방안을 제시하였다. 이 방법을 통해 37개의 테스트 데이터 중 23개의 데이터에 있어서 가장 좋은 해를 찾을 수 있었으며, 13개의 데이터에 대해서는 기존의 가장 좋은 해보다 더 좋은 해를 찾을 수 있었다. 선형 계획법을 활용한 아이템 선택 방법은 특정 문제의 지식을 필요로 하는 것이 아니기 때문에 본 논문의 대상 문제뿐만 아니라 일반적인 조합 최적화 문제로도 쉽게 적용이 가능할 것으로 판단된다. 따라서 향후로 정수 계획법 기반 지역 탐색의 적용이 가능한 다른 최적화 문제나 실세계 문제의 발굴 및 적용을 통해 선형 계획법을 활용한 아이템 선택 방법의 유용성을 추가로 검증할 예정이다. 아울러 초기해 생성을 위한 정수 계획법의 수행 시간과 아이템 선택 시 선형 계획법을 통해 선택하는 비율 등 많은 파라미터 값들을 실험적으로 결정하였는데, 아이템 선택 비율을 포함하여 파라미터 값들에 대한 최적값을 결정하기 위한 연구도 함께 진행될 필요가 있다.

REFERENCES

[1] M. Hifi, and A. Sbihi, "Heuristic Algorithms for the Multiple-choice Multidimensional Knapsack Problem," Journal of the Operational Research Society, Vol.55, pp.1323-1332, Dec. 2004.

[2] M. Caserta, and S. Voß, "An Exact Algorithm for the Reliability Redundancy Allocation Problem," European Journal of Operational Research, Vol.244, No.1, pp.110-116, July 2015.

[3] D. Pisinger, "Budgeting with Bounded Multiple-choice Constraints," European Journal of Operational Research, Vol.129, No.3, pp.471-480, March 2001.

[4] C. Basnet, and J. Wilson, "Heuristics for Determining

the Number of Warehouses for Storing Non-compatible Products," International Transactions in Operational Research, Vol.12, No.5, pp.527-538, Sep. 2005.
[5] S. Khan, K.F. Li, E.G. Manning, and M.M. Akbar, "Solving the Knapsack Problem for Adaptive Multimedia Systems," Studia Informatica Universalis, Vol.2, No.1, pp.157-178, 2002.
[6] M. Hifi, M. Michrafy, and A. Sbihi, "A Reactive Local Search-based Algorithm for the Multiple-choice Multi-dimensional Knapsack Problem," Computational Optimization and Applications, Vol.33, No.2-3, pp.271-285, March 2006.
[7] H. Shojaei, T. Basten, M. Geilen, and A. Davoodi, "A Fast and Scalable Multidimensional Multiple-choice Knapsack Heuristic," ACM Transactions on Design Automation of Electronic Systems, Vol.18, No.4, pp.51:1-32, Oct. 2013.
[8] Y. Chen, and J.K. Hao, "A Reduce and Solve Approach for the Multiple-choice Multidimensional Knapsack Problem," European Journal of Operational Research, Vol.239, NO.2, pp.313-322, Dec. 2014.
[9] C. Gao, G. Lu, X. Yao, and J. Li, "An Iterative Pseudo-gap Enumeration Approach for the Multidimensional Multiple-choice Knapsack Problem," European Journal of Operational Research, Vol.260, No.1, pp.1-11, July 2017.
[10] K. Genova, and V. Guliashki, "Linear Integer Programming Methods and Approaches-A Survey," Cybernetics and Information Technologies, Vol.11, No.1, pp.3-25, 2011.
[11] J. Hwang, "An Integer Programming-based Local Search for the Set Covering Problem," Journal of The Korea Society of Computer and Information, Vol.19, No.10, pp.13-21, Oct. 2014.
[12] J. Hwang, "Integer Programming-based Local Search Techniques for the Multidimensional Knapsack Problem," Journal of The Korea Society of Computer and Information, Vol.17, No.6, pp.13-27, June 2012.
[13] "IBM Decision Optimization CPLEX Modeling for Python", V2.7, IBM Corporation, 2018.

Authors

Junha Hwang received the B.S., M.S. and Ph.D. degrees in Computer Engineering from Pusan National University, Korea, in 1995, 1997 and 2002, respectively. Dr. Hwang joined the faculty of the Department of Computer Engineering at Kumoh National Institute of Technology, Gumi, Korea, in 2002. He is currently a Professor in the Department of Computer Engineering, Kumoh National Institute of Technology. He is interested in artificial intelligence, combinatorial optimization, machine learning, and programming language.