Journal of The Korea Society of Computer and Information
Vol. 20 No. 9, pp. 21-29, September 2015

www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2015.20.9.021

An Integer Programming-based Local Search for the Set Partitioning Problem

Junha Hwang*

Abstract

The set partitioning problem is a well-known NP-hard combinatorial optimization problem, and it is formulated as an integer programming model. This paper proposes an Integer Programming-based Local Search for solving the set partitioning problem. The key point is to solve the set partitioning problem as the set covering problem. First, an initial solution is generated by a simple heuristic for the set covering problem, and then the solution is set as the current solution. Next, the following process is repeated. The original set covering problem is reduced based on the current solution, and the reduced problem is solved by Integer Programming which includes a specific element in the objective function to derive the solution for the set partitioning problem. Experimental results on a set of OR-Library instances show that the proposed algorithm outperforms pure integer programming as well as the existing heuristic algorithms both in solution quality and time.

▸ Keyword : Set partitioning problem, Set covering problem, Integer programming-based local search

I. Introduction

집합 분할 문제(Set Partitioning Problem, SPP)는 0 또는 1의 값을 가진 m행 n열의 행렬로부터 각 행이 단 하나의 열에 의해 커버되도록 일부의 열들을 선택하되 선택된 열들의 비용의 합을 최소화하는 문제로 정의된다. 행을 의미하는 집합을 $M = \{ 1, 2, ..., m \}$이라 하고, 열을 의미하는 집합을 $N = \{ 1, 2, ..., n \}$이라고 할 때 집합 분할 문제는 식 (1)~(3)과 같이 표현된다.

$$\text{Minimize} \quad z = \sum_{j \in N} c_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j \in N} a_{ij} x_j = 1, \quad \forall\, i \in M \tag{2}$$

$$x_j \in \{0, 1\}, \quad \forall\, j \in N \tag{3}$$

식 (1)에서 x_j는 0 또는 1의 값을 갖는 결정 변수로서 x_j의 값이 1이면 열 j가 선택되었음을 의미하며, c_j는 0보다 큰 값으로 열 j의 비용을 의미한다. 식 (1)의 목적함수에 의해 선택된 열들의 비용의 합을 최소화하게 된다. 식 (2)에서 a_{ij}는 m행 n열인 행렬의 원소로서 0 또는 1의 값을 가지며, 이 값이 1인 경우 행 i가 열 j에 의해 커버됨을 의미한다. 따라서 식 (2)로 인해 각 행은 단 하나의 열에 의해 커버되어진다.

예를 들어 그림 1과 같이 6행 10열로 이루어진 집합 분할 문제가 있을 때, 제약조건을 만족하는 열의 조합은 (1, 3, 4, 6), (1, 5, 6, 8), (2, 4, 10) 등 여러 개가 존재하지만 그중에서 비용의 합을 최소화하는 열의 조합은 (1, 5, 6, 8)이며 그때의 목적함수 값은 10이 된다.

	1	2	3	4	5	6	7	8	9	10
1	1						1			1
2		1				1				
3				1				1		
4	1	1							1	
5				1	1		1			
6			1					1		1
c_j	2	5	4	6	4	1	3	3	7	5

Fig. 1. An Example of Set Partitioning Problem

• First Author: Junha Hwang, Corresponding Author: Junha Hwang
*Junha Hwang (jhhwang@kumoh.ac.kr), Dept. of Computer Engineering, Kumoh National Institute of Technology
• Received: 2015. 07. 14, Revised: 2015. 08. 09, Accepted: 2015. 08. 26.
• This work was supported by Research Fund, Kumoh National Institute of Technology.

승무일정계획 문제를 비롯하여 차량경로 문제, 자원배치 문제 등 실세계의 많은 문제들이 집합 분할 문제로 표현될 수 있다[1]. 집합 분할 문제는 NP-hard 문제로 알려져 있으며, 지금까지 이 문제를 해결하기 위해 많은 탐색 기법들이 제시되어 왔는데 크게 두 가지로 구분할 수 있다. 첫 번째는 정수계획법(Integer Programming, IP)의 적용에 관한 연구이다[2, 3]. 정수계획법은 식 (1)~(3)과 같이 대상 문제가 선형적으로 표현될 때 적용이 가능하다. 정수계획법의 경우 문제의 규모가 비교적 작을 때에는 빠른 시간 내에 최적해의 도출이 가능하지만, 문제의 규모가 커짐에 따라 최적해를 도출하기까지 과도한 시간이 소요되거나 메모리 문제로 인해 적용이 불가능한 경우도 발생한다. 두 번째는 유전 알고리즘, 개미 시스템 등 휴리스틱 탐색 기법들을 적용한 연구이다[4, 5, 6]. 휴리스틱 탐색 기법들의 경우 최적해의 도출을 보장하지는 못하지만 규모가 큰 데이터에 대해서도 비교적 짧은 시간 내에 준최적해의 도출이 가능하다는 장점이 있다.

본 논문에서는 집합 분할 문제를 해결하기 위한 정수계획법 기반 지역 탐색(Integer Programming-based Local Search, IPbLS)의 활용 방안을 제시한다. 정수계획법 기반 지역 탐색은 지역 탐색의 틀 내에서 이웃해를 생성하기 위해 정수계획법을 적용하는 탐색 기법으로 선형적으로 표현되는 조합 최적화 문제에 효과적인 것으로 알려져 있다[7, 8, 9]. 본 논문에서는 집합 분할 문제를 이와 유사한 문제인 집합 커버링 문제로 모델링하여 풀게 된다. 먼저 간단한 휴리스틱을 사용하여 집합 커버링 문제의 초기해를 생성한 후 이 해를 현재해로 설정한다. 다음으로는 현재해를 기반으로 원래의 집합 커버링 문제를 축소시킨 후 축소된 집합 커버링 문제의 최적해를 도출함으로써 현재해를 개선하는 과정을 반복적으로 수행하게 되는데, 축소된 문제를 해결하기 위해 정수계획법을 사용한다. 이때 정수계획법의 목적함수로 특정 요소를 추가함으로써 자연스럽게 집합 분할 문제의 해가 도출될 수 있도록 유도한다. OR-Library의 집합 분할 문제 데이터를 대상으로 실험을 수행한 결과, 본 논문에서 제안한 기법을 통해 순수한 정수계획법이나 기존의 휴리스틱 탐색 기법들보다 더 빠른 시간 내에 보다 좋은 해를 도출할 수 있음을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 집합 커버링 문제와 정수계획법 기반 지역 탐색에 대해 설명하며, 3장에서는 집합 분할 문제의 해결을 위한 정수계획법 기반 지역 탐색의 적용 방안에 대해 기술한다. 4장에서는 실험 결과를 제시하고 분석하며, 마지막으로 5장에서 결론 및 향후 과제에 대해 기술한다.

II. Related Work

1. Set Covering Problem

집합 커버링 문제(Set Covering Problem, SCP)는 0 또는 1의 값을 가진 m행 n열의 행렬로부터 일부의 열들을 선택하여 모든 행을 커버하되 선택된 열들의 비용의 합을 최소화하는 문제로 정의된다. 집합 커버링 문제는 식 (4)~(6)과 같이 표현될 수 있다[8].

$$\text{Minimize} \quad z = \sum_{j \in N} c_j x_j \tag{4}$$

$$\text{subject to} \quad \sum_{j \in N} a_{ij} x_j \geq 1, \quad \forall i \in M \tag{5}$$

$$x_j \in \{0, 1\}, \quad \forall j \in N \tag{6}$$

집합 분할 문제인 식 (1)~(3)과 비교해 볼 때 식 (5)의 부등호를 제외하면 모두 동일하다. 즉, 집합 분할 문제의 경우 각 행이 단 하나의 열에 의해 커버되어야 하는 반면에 집합 커버링 문제의 경우에는 각 행이 열의 개수에 관계없이 1개 이상의 열에 의해 커버되면 된다. 예를 들어 그림 1의 집합 분할 문제를 집합 커버링 문제로 변환할 경우 열 5 대신 열 7을 선택함으로써 목적함수 값이 9인 조합을 도출할 수 있다.

승무일정계획 문제, 자원배치 문제 등 실세계의 많은 문제들이 집합 커버링 문제로도 표현될 수 있는데, 집합 커버링 문제 또한 NP-hard 문제로 알려져 있다. 하지만 집합 커버링 문제의 경우 식 (5)의 특성에 의해 비교적 쉽게 유효한 해들을 생성할 수 있다. 예를 들면, 모든 행들을 커버할 때까지 무작위로 열을 하나씩 추가해 나가는 것이다. 그러나 집합 분할 문제의 경우 각 행이 단 하나의 열에 의해 커버되어야 한다는 식 (2)의 보다 강화된 제약조건으로 인해 유효한 해 하나를 만드는 것조차 쉬운 일이 아니다[4]. 이와 같은 이유로 본 논문에서는 기본적으로 집합 분할 문제를 집합 커버링 문제로 모델링하여 해를 도출하고 있다.

2. Integer Programming-based Local Search

조합 최적화 문제가 선형식으로 표현되는 경우 이를 정수계획법 모델이라고 부르며, 이에 대한 최적해를 도출하기 위한 한 가지 기법이 정수계획법이다. 정수계획법은 기본 탐색 기법으로 분지 한계법(Branch and Bound)을 사용하고 있으며, 이외에도 정수계획법 자체의 성능을 향상시키기 위한 연구가 활발히 진행되고 있다[10]. 정수계획법의 장점은 중소 규모 문제의 경우 매우 빠른 시간 내에 최적해의 도출이 가능하다는 것이다. 반면에 문제의 규모가 커질 경우 메모리 문제로 인해 정수계획법의 적용 자체가 불가능할 수 있으며, 적용이 가능하다 하더라도 최적해를 도출하기까지 매우 많은 시간이 소요될 수 있다는 단점이 있다.

정수계획법 기반 지역 탐색은 지역 탐색의 틀 내에서 이웃해 생성을 위해 정수계획법을 활용하는 탐색 기법으로 기본적인 알고리즘은 그림 2와 같으며[8], 그림 3은 각 단계에 대한 예를 나타낸 것이다. 하나의 해는 그림 3(a)의 $x_1, x_2, \ldots, x_{10}$과 같이 0 또는 1의 값을 가진 일련의 결정 변수들로 표현될 수 있으며, 이 해를 X로 표현한다. 먼저 휴리스틱 기법 등을 통해

초기해를 생성하고 이 해를 현재해로 설정한다. 그러고 나서 현재해를 개선하기 위해 문제를 축소하고 정수계획법을 적용하는 과정을 반복적으로 수행하게 된다. 여기서 문제 축소 과정은 다음 정수계획법 수행에 참여할 열들을 선택하는 과정이라 할 수 있다. 이를 위해 현재해 X로부터 x_j의 값이 1인 열들 중 k개를 선택하게 된다. 그림 3(b)에서는 열 1, 3, 4, 5 중 열 3, 4가 선택되었음을 보여주고 있다. 이 변수들과 현재해 X에서 x_j의 값이 0인 열들이 다음 정수계획법 수행에 참여할 열들이 된다. 즉, 그림 3(c)와 같이 열 1과 6의 값을 1로 고정한 상태에서 정수계획법을 다시 실행하게 된다. 이와 같이 정수계획법에서 고려해야 될 변수들의 수를 줄임으로써 정수계획법 자체의 탐색 공간을 줄이게 된다. k의 값을 크게 설정할수록 정수계획법에서 고려해야 될 경우의 수가 증가하게 되어 최적의 이웃해를 도출하기까지 더 많은 시간이 소요되지만 더 좋은 이웃해가 나올 가능성은 커지게 된다. 반면에 k의 값을 작게 설정할수록 더 빨리 최적의 이웃해를 도출할 수 있지만 이웃해의 질은 떨어질 가능성이 커지게 된다.

```
Algorithm IPbLS
    X : Variable vector (Current solution).
    Obj : Objective function.
    k : The number of variables to be selected.
    IP : An integer programming solver.
Begin
    X = Make an initial solution
    While stopping condition is not met Do
        Select k variables among variables with value 1 from X
        Add Obj and all constraints to IP
            • Fix values of unselected variables among
              variables with value 1 from X
        X = Make a neighbor solution with IP
    End While
    return X
End Begin
```

Fig. 2. General Integer Programming-based Local Search

Fig. 3. An Example of IPbLS Steps

일반적인 지역 탐색에서는 이웃해 하나를 만들기 위해 1개 또는 2개 정도의 변수들에 대한 값을 변경해 보는 것이 보통이다. 왜냐하면 변경할 변수들의 개수가 늘어날수록 고려해야 될 이웃해의 개수는 기하급수적으로 늘어나기 때문이다. 이로 인해 대부분의 지역 탐색은 지역 최적해에서 빠져나오지 못하는 지역 최적화 문제를 갖고 있다. 하지만 정수계획법 기반 지역 탐색에서는 정수계획법의 특성을 활용하여 한 번에 더 많은 변수들의 값을 변경해 봄으로써 지역 최적화 문제를 개선하고 있다.

정수계획법 기반 지역 탐색은 지금까지 다차원 배낭 문제, 집합 커버링 문제, 서비스 네트워크 설계 문제 등 다양한 조합 최적화 문제를 해결하기 위해 적용되어 왔으며[7, 8, 9], 그 성능 또한 다른 휴리스틱 탐색 기법들에 비해 우수한 것으로 확인되었다. 본 논문에서는 집합 분할 문제를 풀기 위한 정수계획법 기반 지역 탐색의 활용 방안을 제시한다.

III. An IPbLS for the SPP

1. Overall Structure

집합 분할 문제 해결을 위한 정수계획법 기반 지역 탐색의 전체 구조는 그림 4와 같이 그림 2의 일반적인 정수계획법 기반 지역 탐색의 구조와 유사하다.

```
Algorithm IPbLS for SPP
    X1 : The column set { j | xj = 1, j ∈ N } from X.
    Obj' : Revised objective function.
Begin
    X = Make an initial solution for SCP
    While stopping condition is not met Do
        Select k variables from X1
        Reduce problem with unselected variables of X1
        Add Obj' and all constraints for SCP to IP
        X = Make a neighbor solution with IP
    End While
    Return X
End Begin
```

Fig. 4. Integer Programming-based Local Search for the Set Partitioning Problem

여기서 가장 중요한 점은 집합 분할 문제를 집합 커버링 문제로 모델링하여 해결한다는 것이다. 먼저 집합 커버링 문제의 초기해를 생성하여 현재해로 지정한다. 그리고 수행 시간 등 종료 조건을 만족할 때까지 문제 축소 및 정수계획법 적용 단계를 반복적으로 수행하게 된다. 문제 축소를 위해 x_j의 값이 1인 열들로부터 k개를 선택하고 그 결과를 기반으로 열과 행을 선별함으로써 정수계획법 대상 문제를 축소하게 된다. 그리고 정수계획법 적용 시에는 기본적으로 집합 커버링 문제의 목적함수와 제약조건을 사용하되 수정된 목적함수(Obj')를 사용함으로써 자연스럽게 집합 분할 문제의 해가 도출되도록 하였다.

초기해 생성, k개의 열 선택, 문제 축소, 정수계획법 적용에 관한 구체적인 방법에 대해서는 다음 각 절에서 설명한다.

2. Initial Solution Generation Method for SCP

집합 커버링 문제의 초기해 생성 방법으로는 확률적 방법을 사용하였으며, 이는 기존 연구 [8]에서의 초기해 생성 방법과 동일하다. 이에 대해 간략히 설명하면 다음과 같다.

모든 행들이 커버될 때까지 열을 하나씩 선택하되, 비용이 적으면서 현재 커버되지 않은 행들을 많이 커버하는 열을 우선적으로 선택한다. 즉, 현재 커버되지 않은 행들 중 열 j가 커버하는 개수를 h_j라고 한다면 (c_j / h_j)의 값이 가장 작은 열을 선택하는 것이다. 만약 (c_j / h_j)의 값이 동일한 열들이 있을 경우에는 그 열들 중 무작위로 하나를 선택한다. 모든 행들이 커버된 후에는 후처리 과정을 통해 특정 열이 커버하는 행들이 함께 선택된 다른 열들에 의해 모두 커버된다면 해당 열을 제외하게 된다. 이와 같은 열들이 여러 개라면 목적함수 값을 최소화하기 위해 비용 c_j의 값이 큰 것부터 하나씩 차례로 제거하게 된다.

3. Column Selection Method

다음은 현재해에서 x_j의 값이 1인 열들을 대상으로 k개의 열들을 선택하는데, 그 방법으로는 무작위 방법과 확률적 방법을 사용할 수 있다. 무작위 방법은 말 그대로 열 k개를 무작위로 선택하는 방법이다. 확률적 방법은 열 j에 대한 선택 확률을 식 (7)에 따라 계산하고 그 값에 따라 확률적으로 하나씩 선택하는 것이다[8]. 여기서 n_1은 현재해에서 x_j 값이 1인 열들의 개수이고, h_j는 열 j를 현재해에서 제거함으로써 발생하는 공백행, 즉 커버되지 못하는 행들의 개수를 의미한다. 이에 따라 열 j의 비용이 클수록, 그리고 열 j로 인해 발생하는 공백행이 적을수록 선택될 확률은 커지게 된다. 비용이 클수록 목적함수에 도움이 되지 못하며 발생하는 공백행의 개수가 적을수록 전체 행을 커버하는 데 도움이 되지 못하기 때문이다.

$$p(j)=\frac{c_j/(h_j+1)}{\sum_{i=1}^{n_1}(c_i/(h_i+1))} \tag{7}$$

k개의 열 선택 방법으로 무작위 방법과 확률적 방법 중 어느 것이 더 좋은지는 알 수 없다. 무작위 방법의 경우 더 다양한 영역으로의 탐색이 가능한 반면 확률적 방법의 경우 현재해의 특성을 잘 반영하여 해당 영역에서의 집중적인 탐색이 가능하다. 결국 데이터의 특성에 따라 각 방법의 장점이 효과를 발휘할 수도 있고 역효과를 초래할 수도 있다.

4. Problem Reduction Method

현재해에서 x_j의 값이 1인 열들 중 선택되지 않은 열들의 경우 다음 정수계획법 적용 시 미리 선택된 열, 즉 1로 고정된다. 그렇다면 이 열들에 의해 커버되는 행들 또한 정수계획법 적용 시 고려하지 않아도 된다. 이와 같은 특성을 활용하여 정수계획법을 적용할 문제를 축소하게 된다. 이 과정 또한 기존 연구[8]에서의 적용 방법과 유사하지만 집합 분할 문제의 특성에 의해 다소 차이가 나게 된다.

그림 5는 문제 축소 과정의 예를 나타낸 것이다. 이 예에서 행의 개수는 6개이고 열의 개수는 10개이다. 그리고 현재해에 포함된 열은 1, 3, 4, 6이며 여기서 앞서 k개의 열 선택 과정을 통해 열 3과 4가 선택되었다고 가정한다.

먼저 그림 5(a)와 같이 선택되지 않은 열, 즉 열 1과 6만을 고려하여 열 1과 6에 의해 커버되지 않는 행들만 표현하면 그림 5(b)와 같다. 즉, 다른 행들은 열 1과 6에 의해 커버되므로 행 3, 5, 6만을 대상으로 이 행들을 커버하는 열들을 선택하는 문제가 된다. 이를 위해 이번에는 정수계획법에 참여할 열들을 선택할 차례이다. 전체 열 집합에서 행 3, 5, 6만을 커버하는 열들만 선택하되, 이미 변수의 값이 1로 고정될 열 1과 6은 제외한다. 그림 5(c)는 이 과정의 결과로 열 4, 5, 8이 선택된 예를 보인 것이다. 여기서 주의할 사항은 행 3, 5, 6만 커버하는 열들을 선택했기 때문에 경우에 따라서는 특정 행이 어떤 열에 의해서도 커버되지 않는 상황이 발생할 수 있다는 것이다. 예를 들면 열 5가 행 1과 행 5를 커버하고 있었다면 그림 5(c)에서 열 5는 제외될 것이고 결국 행 5를 커버하는 열이 존재하지 않게 된다. 그러나 보통 k의 값을 크게 설정하고 이에 따라 많은 열들이 다음 정수계획법의 대상이 되므로 대부분의 경우에는 모든 행들이 1개 이상의 열들에 의해 커버되어진다. 그렇다 하더라도 이와 같은 상황을 고려하여 다음에 설명할 정수계획법 적용 시에는 어떤 열에 의해서도 커버되지 않는 행에 대해서는 식 (5)의 제약조건을 생략하게 된다.

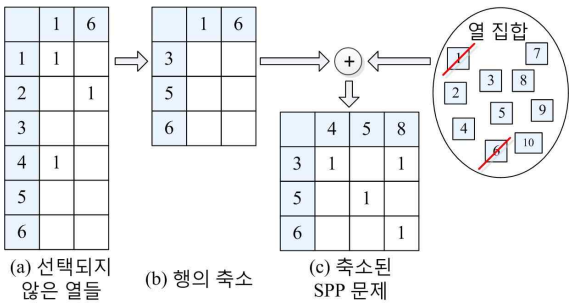

Fig. 5. Problem Reduction Steps

5. IP for the Reduced Problem

이제 축소된 집합 분할 문제를 풀면 된다. 그런데 축소된 문제의 경우 집합 분할 문제의 해가 존재하지 않을 수도 있기 때문에 집합 커버링 문제를 풀게 된다. 단, 집합 분할 문제의 해를 유도하기 위해 집합 커버링 문제의 목적함수로는 식 (4) 대신 식 (8)을 사용하며, 식 (5)와 (6)은 그대로 사용한다. 식 (8)에서 M′와 N′는 각각 축소된 문제에서의 행의 집합과 열의 집합을 의미한다.

$$\text{Minimize}\quad z=\sum_{j\in N'}c_jx_j+\alpha\sum_{i\in M'}\Big(\sum_{j\in N'}a_{ij}x_j-1\Big) \tag{8}$$

식 (8)에서 특정 행 i에 대한 $(\sum_{j\in N'}a_{ij}x_j-1)$의 값은 식 (5)에 의해 0 이상의 값을 갖게 된다. 이 값이 0이란 말은 단 하나의 열에 의해 커버된다는 의미이며, 이 값이 1 이상이란 말은 2개

이상의 열에 의해 커버된다는 의미이다. 따라서 이 수식을 목적 함수에 추가하여 최소화함으로써 집합 분할 문제의 해가 유도될 수 있도록 하였다.

이때 상수값 α의 크기에 따라 비용 최소화를 우선적으로 고려할 것인지, 아니면 집합 분할 문제 해의 유도를 우선적으로 고려할 것인지가 결정된다. α의 값이 작아지면 비용 최소화를 우선하며 이 값이 커지면 집합 분할 문제 해의 유도를 우선하게 된다. 본 연구에서는 α의 값으로 다음 두 가지 값을 사용하고 있다. 여기서 *MAX_COST*는 주어진 집합 분할 문제의 열들에 대한 비용값들 중 가장 큰 비용값을 의미하며 *MIN_COST*는 가장 작은 비용값을 의미한다.

① α = *MAX_COST* * 10
② α = *MIN_COST*

①의 값은 정상 모드에서 적용되며 가능한 집합 분할 문제의 해를 유도하게 된다. 그런데 정수계획법 기반 지역 탐색이라 하더라도 지역 탐색의 일종이기 때문에 지역 최적화 현상이 발생할 수 있다. 따라서 정수계획법이 총 5회가 실행되는 동안 해의 개선이 없을 경우 탐색 다각화를 모색하게 된다. 이때 ②의 α 값을 사용하게 된다. ②의 경우 상대적으로 작은 값이기 때문에 집합 분할 문제를 해결하기 보다는 집합 커버링 문제의 틀 내에서 비용을 최소화하려고 노력하게 된다. 이 과정에서 정상 모드에서 도출된 해와는 다른 해의 도출이 가능할 것으로 기대된다. 다각화 모드는 총 2회 동안 실행된 후 다시 정상 모드로 변경된다.

IV. Experimental Results

1. Experimental Environment

본 연구에서는 OR-Library에 있는 55개의 집합 분할 문제 데이터를 사용하였다. OR-Library는 집합 분할 문제뿐만 아니라 다양한 조합 최적화 문제에 대한 테스트 데이터들을 포함하고 있으며, 관련 연구들의 비교를 위해 많이 사용되고 있다 [11]. 본 연구에서 사용한 집합 분할 문제 데이터는 4개의 항공 회사로부터 제공된 실세계 승무일정계획 데이터로서 각 데이터는 다양한 행의 개수, 열의 개수, 밀도로 구성되어 있다 [2]. 또한 모든 데이터에 대한 최적해가 알려져 있기 때문에 [2], 지금까지 정수계획법 또는 휴리스틱 기법들에 대한 비교 연구를 위해 많이 사용되어 왔다.

본 연구의 모든 실험은 Intel Core 2 Duo E8400 3GHz, 3GB RAM PC 및 Windows 7 32비트 운영체제 상에서 수행되었다. 그리고 정수계획법 기반 지역 탐색을 구현하기 위한 정수계획법 개발 도구로는 IBM ILOG CPLEX 12.6.1을 사용하였다. CPLEX는 정수계획법 개발을 위한 상용 라이브러리로서 현재 상업적 또는 학술적 목적으로 가장 많이 사용되고 있으며 그 성능 또한 가장 우수한 것으로 인정받고 있다.

2. Experimental Results for IPbLS

표 1은 총 7개의 실험 데이터에 대해 정수계획법 기반 지역 탐색 적용 시 다양한 파라미터의 설정에 따른 실험 결과를 나타낸 것이다. US01은 55개의 데이터 중 규모가 가장 큰 데이터이며 AA01 또한 비교적 큰 규모의 데이터에 포함된다.

고려해야 될 파라미터로는 현재해로부터 k개의 열을 선택할 때 k의 값과 열의 선택 방법이 있으며, 그 외에 정수계획법 수행 시 다각화 사용 여부가 있다. 표 1에서 Name은 데이터의 이름, Rows는 행의 개수, Columns는 열의 개수를 의미한다. k-ratio는 k의 값과 관련이 있는데, 좋은 k 값이란 데이터마다 그리고 현재해의 상황에 따라 달라질 수 있다. 따라서 k의 값을 현재해에서 x_i의 값이 1인 변수들의 개수에 대한 일정 비율로 나타내었으며 이를 k-ratio라 표현하였다. 예를 들어 k-ratio가 0.6이란 말은 현재해에서 x_i의 값이 1인 변수들 중 60%를 제외하고 나머지 40%의 변수들의 값을 1로 고정하겠다는 의미이다. 따라서 이 값이 클수록 전역 탐색에 가까워진다. 대부분의 데이터에 있어서 k-ratio의 경우 0.6 이상과 같이 어느 정도 큰 경우에 정수계획법 기반 지역 탐색의 성능이 우수한 것으로 나타났다. 따라서 실험 결과에는 k-ratio의 값이 0.6, 0.7, 0.8, 0.9인 경우에 대해 기술하였다.

열 선택 방법에 있어서 Random은 무작위 선택 방법을 의미하고, Stochastic은 확률적 선택 방법을 의미한다. 'with Diversification'은 반복적인 정수계획법 수행 시 다각화를 사용한다는 의미이며 'without Diversification'은 다각화를 사용하지 않는다는 의미이다. 각 데이터의 각 파라미터에 대해 총 10회의 실험을 실시하였으며 한 번의 실험 당 수행 시간을 1,200초로 제한하였다. #Success는 10회의 실험 중 최적해를 도출한 횟수를 의미하며 time(sec)은 최적해를 도출하기까지 소요된 초 단위 평균 시간을 의미한다. 각 데이터에 있어서 가장 좋은 결과에 대해서는 음영 배경 및 빨간색 볼드체로 표시하였으며, 각 데이터 내에서 다각화 적용 여부에 따른 결과 중 가장 좋은 결과는 음영 배경 및 파란색 볼드체로 표시하였다.

전체적인 실험 결과를 살펴보면 규모가 큰 데이터인 AA01과 US01을 제외하면 대부분의 경우에 있어서 10회 모두 최적해를 도출하였으며, AA01과 US01의 경우에도 다각화 적용 여부 등의 파라미터 설정에 따라 10회 모두 최적해의 도출이 가능한 경우도 있다.

먼저 다각화의 효과를 살펴보면 NW01, NW14, NW21, US01, KL02에서는 다각화를 하는 경우 더 좋은 결과를 보였고 NW11과 AA01에서는 다각화를 하지 않은 경우 약간 더 좋은 결과를 보였다. 전반적으로는 다각화를 적용하는 것이 더 좋은 것으로 판단되며 특히 규모가 가장 큰 데이터인 US01의 경우에는 다각화의 효과가 매우 큰 것으로 나타났다.

Table 1. Experimental Results for Integer Programming-based Local Search

Data			k-ratio	with Diversification				without Diversification			
				Random		Stochastic		Random		Stochastic	
Name	Rows	Columns		#Sucess	time(sec)	#Sucess	time(sec)	#Sucess	time(sec)	#Sucess	time(sec)
NW01	135	51975	0.6	**10**	15.318	**10**	44.389	**10**	41.497	8	301.900
			0.7	**10**	8.354	**10**	10.765	**10**	29.660	**10**	245.246
			0.8	**10**	5.099	**10**	14.965	**10**	9.329	**10**	17.891
			0.9	**10**	6.504	**10**	10.160	**10**	5.514	**10**	9.353
NW11	39	8820	0.6	**10**	0.337	**10**	3.708	**10**	0.391	**10**	2.348
			0.7	**10**	0.333	**10**	0.680	**10**	0.281	**10**	0.514
			0.8	**10**	0.382	**10**	0.686	**10**	0.371	**10**	0.542
			0.9	**10**	0.399	**10**	0.452	**10**	0.456	**10**	0.410
NW14	73	123409	0.6	**10**	17.079	9	476.609	**10**	17.305	4	473.154
			0.7	**10**	10.029	**10**	164.928	**10**	17.004	9	343.123
			0.8	**10**	12.935	**10**	36.833	**10**	13.597	**10**	20.451
			0.9	**10**	9.622	**10**	24.891	**10**	12.080	**10**	20.499
NW21	25	577	0.6	**10**	0.046	**10**	0.086	**10**	0.071	**10**	0.056
			0.7	**10**	0.037	**10**	0.026	**10**	0.048	**10**	0.059
			0.8	**10**	0.028	**10**	0.021	**10**	0.060	**10**	0.031
			0.9	**10**	0.026	**10**	0.018	**10**	0.047	**10**	0.020
AA01	823	8904	0.6	0	–	0	–	0	–	2	749.890
			0.7	1	61.620	2	704.614	0	–	2	110.359
			0.8	8	344.919	4	611.467	7	439.189	3	780.288
			0.9	**10**	386.756	**10**	460.707	**10**	299.679	**10**	554.169
US01	145	1053137	0.6	8	411.466	**10**	188.395	6	122.080	5	125.806
			0.7	8	489.196	9	321.087	0	–	4	179.905
			0.8	0	–	3	75.238	1	38.173	4	61.660
			0.9	0	–	1	42.400	0	–	0	–
KL02	71	36699	0.6	**10**	2.164	**10**	1.775	**10**	3.405	**10**	1.624
			0.7	**10**	1.145	**10**	1.132	**10**	2.028	**10**	1.744
			0.8	**10**	1.763	**10**	1.243	**10**	2.201	**10**	1.381
			0.9	**10**	1.775	**10**	1.248	**10**	2.210	**10**	1.895

k-ratio에 따른 결과를 보면 NW01, NW14, NW21, AA01에 있어서는 0.9와 같이 큰 값일 때 더 좋은 결과를 보였고 NW11에서는 0.7, US01에서는 0.6일 때 가장 좋은 결과를 보였다. 중소 규모의 문제에 있어서는 k의 값이 클수록 더 좋은 결과를 보였는데, 이는 결국 정수계획법의 우수성을 보여주는 것이라 할 수 있다. 그러나 US01과 같이 문제의 규모가 큰 경우에는 k의 값이 커짐에 따라 정수계획법의 적용 자체가 불가능해질 수 있고 때로는 축소된 문제의 최적해를 도출하기까지 과도한 시간이 소요될 수 있기 때문에 0.6과 같이 상대적으로 작은 값일 때 더 좋은 성능을 발휘하게 된다. 그 외에도 NW11의 결과에 따르면 데이터의 특성에 따라 최적의 k-ratio의 값이 달라질 수 있는 것으로 보인다.

k개의 열 선택 시 선택 방법으로 무작위 방법과 확률적 방법 중 어떤 것이 더 좋은지는 표 1의 결과만으로 판단하기 어렵다. NW01, NW11, NW14, AA01은 무작위 방법이 더 좋은 것으로 판단되며 NW21, US01, KL02는 확률적 방법이 우세한 것으로 판단된다. 결과적으로 데이터의 특성에 따라 무작위 방법이 적합한 경우도 있고 확률적 방법이 적합한 경우도 있는 것으로 보인다.

3. Comparison with Other Algorithms

표 2는 본 연구에서 활용한 모든 데이터에 대한 정수계획법 기반 지역 탐색의 실험 결과를 포함하고 있으며, 아울러 집합분할 문제와 관련된 기존의 주요 연구들에 대한 실험 결과를 함께 제시하였다. 그런데 각각의 연구들마다 주장하는 바가 다르기 때문에 그 성격에 맞게 최종 도출해나 최적해 도출 시간 또는 최적해 도출 횟수를 표기하였다.

정수계획법 기반 지역 탐색과의 비교를 위해 제시한 기존 연구는 크게 정수계획법(IP)과 휴리스틱 탐색 기법(Heuristic)으로 구분된다. 정수계획법으로는 기존 연구 [2]와 [3]의 결과를 제시하였으며, 그 외에 정수계획법 개발 도구인 CPLEX를 단순하게 적용한 실험 결과를 함께 제시하였다. 기존 연구 [2]는 분지 절단법(Branch-and-Cut)을 기반으로 정수계획법의 성능을 향상시켰으며 그 결과 모든 테스트 데이터에 대한 최적해를 도출할 수 있었다. 각 데이터에 대한 최적해의 목적함수는 표 2의 Optimal 열에 표기하였으며, 기존 연구 [2]에 대한 실험 결과로는 최적해가 도출될 때까지의 초 단위 수행시간을 time 열에 표기하였다. 참고로 [2]에서는 대부분의 데이터에 대한 실험을 위해 IBM RISC6000 워크스테이션을 사용하였고, US01 등 몇몇 대규모 데이터를 위해 CONVEX C-220 미니슈퍼컴퓨터를 사용하였다. 기존 연구 [3]은 열생성(Column Generation) 기법을 사용하였다. 이 연구에서는 NW, AA, US, KL 데이터 그룹에 대해 최대 수행 시간을 각각 100초, 200초, 100초, 100초로 제한하고 최적해의 도출이 가

Table 2. Comparison with Other Algorithms

Data				IP			Heuristic				
Name	Rows	Columns	Optimal	[2]	[3]	CPLEX	[4]	[5]	IPbLS		
				time	*Obj*	time	#Success	*Obj*	*k*-ratio	#Success	time
NW01	135	51975	114852	19.250	–	4.477	X	–	0.9	10	10.160
NW02	145	87879	105444	37.350	–	6.754	0	–	0.9	10	9.416
NW03	59	43749	24492	24.000	24492	2.901	2	–	0.6	10	4.258
NW04	36	87482	16862	2642.000	16956	7.332	4	–	0.8	10	7.432
NW05	71	288507	132878	192.500	–	14.632	0	–	0.9	10	31.725
NW06	50	6774	7810	10.410	7810	0.608	10	8038	0.9	10	0.591
NW07	36	5172	5476	0.740	–	0.218	10	–	0.9	10	0.167
NW08	24	434	35894	0.080	–	0.046	10	35894	0.9	10	0.021
NW09	40	3103	67760	0.530	–	0.156	10	69332	0.9	10	0.106
NW10	24	853	68271	0.130	–	0.062	10	X	0.9	10	0.042
NW11	39	8820	116256	2.050	116265	0.390	9	–	0.9	10	0.452
NW12	27	626	14118	0.090	–	0.046	10	14252	0.9	10	0.073
NW13	51	16043	50146	4.290	50146	0.811	3	–	0.9	10	1.065
NW14	73	123409	61844	87.600	–	6.037	0	–	0.9	10	24.891
NW15	31	467	67743	0.100	–	0.078	10	67743	0.9	10	0.034
NW16	139	148633	1181590	174.400	–	16.474	10	–	0.7	10	23.269
NW17	61	118607	11115	87.530	11115	9.079	5	–	0.7	10	5.460
NW18	124	10757	340160	62.490	340160	0.998	0	345130	0.9	10	6.704
NW19	40	2879	10898	0.500	–	0.171	10	11060	0.8	10	0.081
NW20	22	685	16812	0.620	16812	0.062	10	–	0.9	10	0.057
NW21	25	577	7408	0.300	7408	0.062	10	–	0.9	10	0.018
NW22	23	619	6984	0.340	6984	0.093	10	–	0.7	10	0.023
NW23	19	711	12534	0.340	12534	0.093	10	12880	0.9	10	0.073
NW24	19	1366	6314	0.560	6314	0.093	10	–	0.9	10	0.046
NW25	20	1217	5960	0.620	5960	0.078	10	–	0.9	10	0.064
NW26	23	771	6796	0.340	6796	0.093	10	6880	0.8	10	0.078
NW27	22	1355	9933	0.280	9933	0.093	10	–	0.8	10	0.059
NW28	18	1210	8298	0.400	8298	0.140	10	–	0.9	10	0.146
NW29	18	2540	4274	0.990	4274	0.218	10	–	0.7	10	0.115
NW30	26	2653	3942	0.750	3942	0.156	10	–	0.9	10	0.164
NW31	26	2662	8038	1.430	8038	0.686	10	–	0.8	10	0.252
NW32	19	294	14877	0.170	14877	0.031	10	14877	0.9	10	0.028
NW33	23	3068	6678	1.450	6724	0.982	9	–	0.7	10	0.124
NW34	20	899	10488	0.300	10488	0.156	10	10713	0.9	10	0.134
NW35	23	1709	7216	0.480	7216	0.374	10	–	0.9	10	0.129
NW36	20	1783	7314	3.680	7314	0.452	9	–	0.9	10	0.173
NW37	19	770	10068	0.190	10524	0.093	10	–	0.8	10	0.145
NW38	23	1220	5558	1.350	5558	0.234	10	–	0.9	10	0.124
NW39	25	677	10080	0.190	10080	0.078	10	10545	0.7	10	0.028
NW40	19	404	10809	0.210	10809	0.031	10	–	0.9	10	0.098
NW41	17	197	11307	0.060	11307	0.015	10	11307	0.9	10	0.017
NW42	23	1079	7656	0.990	7666	0.202	10	–	0.9	10	0.393
NW43	18	1072	8904	0.380	8904	0.078	10	–	0.9	10	0.114
AA01	823	8904	56137	14441.000	56213	8.439	X	60246	0.9	10	460.710
AA02	531	5198	30494	10.150	–	0.733	0	37452	0.9	10	8.227
AA03	825	8627	49649	48.420	49663	2.059	X	55082	0.8	10	84.294
AA04	426	7195	26374	139337.000	26374	18.111	0	–	0.9	10	216.920
AA05	801	8308	53839	215.300	53839	2.246	X	58158	0.9	10	69.528
AA06	646	7292	27040	37.300	27063	2.418	0	33524	0.9	10	22.859
US01	145	1053137	10036	1410.600	10036	X	0	–	0.6	10	188.400
US02	100	13635	5965	4.780	–	8.814	10	–	0.9	10	1.065
US03	77	85552	5338	20.270	–	17.830	10	–	0.6	10	2.399
US04	163	28016	17854	11.190	17854	3.666	10	–	0.6	10	2.491
KL01	55	7479	1086	35.400	1086	0.624	7	–	0.8	10	0.830
KL02	71	36699	219	134.480	219	1.466	4	–	0.7	10	1.132

능한지를 실험하였다. 따라서 해당 열 *Obj*는 최종적으로 도출된 해의 목적함수값을 의미하며, 해당 연구에 사용되지 않은 데이터들은 '-' 문자로 표기하였다. CPLEX의 경우 집합 분할 문제의 모델링을 그대로 적용하였고 정수계획법 실행을 위한 설정값 또한 디폴트 값을 사용하였다. CPLEX의 time 열은 최적해가 도출될 때까지의 소요 시간을 의미한다. 단, 최적해를 도출하지 못한 경우 'X'로 표시하였다.

휴리스틱 탐색 기법으로는 기존 연구 [4]와 [5]의 실험 결과를 제시하였다. [4]는 유전 알고리즘을 적용하였으며, #Success 열은 10회 실험 중 최적해를 도출한 횟수를 의미한다. 기존 연구 [5]에서는 개미 시스템을 적용하였으며, *Obj*는 최종적으로 도출한 해의 목적함수값을 의미한다.

본 논문에서 제안한 정수계획법 기반 지역 탐색의 실험 결과는 IPbLS 열에 제시하였다. 제시된 결과는 k개의 열 선택 시 확률적 방법을 사용하고 정수계획법 적용 시 다각화를 사용하여 실험을 수행한 결과이며, 각 데이터 별로 k-ratio 값들 중 가장 좋은 값을 제시하였다. #Success 열은 10회 실험 중 최적해를 도출한 횟수를 의미하며 time 열은 최적해를 도출하기까지의 평균 시간을 나타낸 것이다.

휴리스틱 탐색 기법들 중 유전 알고리즘을 적용한 [4]의 경우 많은 데이터에 있어서 10회 모두 최적해를 도출하였다. 그러나 NW02, NW05 등 총 8개 데이터에 대해서는 단 한 번도 최적해를 도출하지 못했고, 더군다나 NW01과 AA01 등 총 4개의 데이터에 대해서는 제약조건을 만족하는 해 자체도 발견하지 못했다. 수행 시간 또한 NW02 데이터의 경우 최종해 도출 시까지 평균 15,132초가 소요되는 등 전반적으로 많은 시간이 소요되었다. 기존 연구 [6]에서는 [4]와 같이 유전 알고리즘을 사용하여 NW14, AA01, AA03, AA05에 있어서 보다 개선된 결과를 도출하였으나 NW14를 제외하면 여전히 최적해의 도출은 불가능하였다. 이에 반해 정수계획법 기반 지역 탐색의 경우 모든 데이터에 있어서 훨씬 짧은 시간 내에 10회 모두 최적해를 도출할 수 있었다. 개미 시스템을 활용한 기존 연구 [5]의 경우 NW08 등 4개 데이터에 대한 최적해를 도출하였지만 나머지 데이터에 대해서는 최적해 도출에 실패하였으며 NW10에 대해서는 제약조건을 만족하는 해조차 도출하지 못하였다.

휴리스틱 탐색 기법들에 비해 정수계획법 기반 기법들의 성능은 매우 뛰어난 편이다. [2]의 경우 모든 데이터에 대해 최적해를 도출할 수 있었다. 다만 실행 환경의 차이로 절대적인 비교는 힘들지만 정수계획법 기반 지역 탐색이나 다른 정수계획법 기반 기법들에 비해 AA04 등 몇몇 데이터들에 있어서 실행 시간이 과도하게 소요되는 경향이 있다. 기존 연구 [3]의 경우 많은 데이터에 있어서 보다 짧은 시간 내에 최적해를 도출할 수 있었다. 그러나 NW04 등 6개 데이터에 대한 최적해 도출에 실패하였다. CPLEX의 경우 정수계획법 기반 기법들 중 가장 좋은 성능을 발휘하였다. CPLEX는 1개를 제외한 모든 실험 데이터에 대한 최적해를 도출할 수 있었으며 최적해 도출 시까지의 소요 시간도 가장 짧은 편이다. 그러나 문제의 규모가 가장 큰 US01에 대해서는 메모리 문제로 인해 정수계획법의 적용이 불가능하였다. 이와 같이 순수한 정수계획법은 문제의 규모가 커짐에 따라 적용 자체가 불가능하거나 최적해 도출 시까지 과도한 시간이 소요된다는 단점이 있다.

CPLEX와 본 논문에서 제안하는 정수계획법 기반 지역 탐색의 실험 결과를 비교해 보면 정수계획법 기반 지역 탐색이 정수계획법의 이와 같은 단점을 극복하면서도 정수계획법을 능가하는 성능을 발휘하고 있음을 확인할 수 있다. 정수계획법 기반 지역 탐색의 경우 US01 데이터에 대한 최적해를 비교적 짧은 시간 내에 10회 모두 도출할 수 있었다. 그 외의 데이터들에 대해서도 CPLEX보다 더 많은 데이터에 있어서 최적해를 보다 빨리 도출할 수 있는 것으로 확인되었다. 다만 AA01과 AA04 등 몇몇 데이터에 있어서는 최적해 도출 시까지 CPLEX보다 훨씬 많은 시간이 소요된 것으로 확인됨에 따라 이에 대한 원인 분석 및 개선이 필요할 것으로 판단된다.

V. Conclusions

본 논문에서는 집합 분할 문제를 해결하기 위한 정수계획법 기반 지역 탐색의 적용 방안을 제시하였다. 집합 분할 문제의 제약 조건을 완화하여 집합 커버링 문제로 모델링함으로써 정수계획법 기반 지역 탐색을 쉽게 적용할 수 있도록 하였으며, 이웃해 탐색을 위한 정수계획법 적용 시 목적함수를 집합 분할 문제에 적합하도록 수정함으로써 자연스럽게 집합 분할 문제의 해가 도출될 수 있도록 하였다. 부가적으로 다각화 전략 등을 통해 정수계획법 기반 지역 탐색의 성능을 향상시켰다. 실험 결과, 본 논문에서 제시한 정수계획법 기반 지역 탐색을 통해 기존의 휴리스틱 탐색 기법들뿐만 아니라 정수계획법에 기반한 기법들보다 빠른 시간 내에 최적해를 안정적으로 도출할 수 있음을 확인할 수 있었다.

본 논문에서 제시한 정수계획법 기반 지역 탐색의 경우 다른 휴리스틱 탐색 기법들과 마찬가지로 k의 값과 열 선택 방법 등 성능에 영향을 미칠 수 있는 다양한 파라미터를 가지고 있다. 비록 각각의 파라미터가 데이터의 특성에 따라 탐색 성능에 어떤 영향을 미치는지 분석하는 것은 쉬운 일이 아니지만, 향후 파라미터와 탐색 성능의 상관관계를 분석해 봄으로써 최적의 파라미터 설정을 통해 정수계획법 기반 지역 탐색의 성능을 향상시킬 필요가 있다.

REFERENCES

[1] R.E. Marsten, "An Algorithm for Large Set Partitioning Problem," Management Science, Vol. 20, No. 5, pp.774-787, Jan. 1974.

[2] K.L. Hoffman, and M. Padberg, "Solving Airline Crew Scheduling Problems by Branch-and-cut," Management Science, Vol. 39, No. 6, pp.657-682, June 1993.

[3] D. Bredström, K. Jörnsten, M. Rönnqvist, and M. Bouchard, "Searching for Optimal Integer Solutions to Set Partitioning Problems using Column Generation," International Transactions in Operational Research, Vol. 21, No. 2, pp.117-197, March 2014.

[4] P.C. Chu, and J.E. Beasley, "Constraint Handling in Genetic Algorithms: The Set Partitioning Problem," Journal of Heuristics, Vol. 4, No. 4, pp.323-357, Dec. 1998.

[5] B. Crawford, R. Soto, E. Monfroy, et al., "A Hybrid Soft Computing Approach for Subset Problems," Mathematical Problems in Engineering, Vol. 2013, Article ID 716069, 12 pages, June 2013.

[6] C.A. Lin, "Generational Model Genetic Algorithm for Real World Set Partitioning Problems," International Journal of Electronic Commerce Studies, Vol. 4, No. 1, pp.33-46, June 2013.

[7] 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.

[8] 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.

[9] A. Erera, M. Hewitt, M. Savelsbergh, and Y. Zhang, "Improved Load Plan Design Through Integer Programming Based Local Search," Transportation Science, Vol. 47, No. 3, pp.412-427, Nov. 2012.

[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. 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.

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.