Journal of The Korea Society of Computer and Information
Vol. 19, No. 10, October 2014
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2014.19.10.013

집합 커버링 문제를 위한 정수계획법 기반 지역 탐색

황 준 하*

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

Jun-Ha Hwang*

요 약

집합 커버링 문제는 대표적인 조합 최적화 문제들 중 하나로서 n개의 열로부터 일부를 선택하여 m개의 행을 커버하되 비용을 최소화하는 문제로 정의된다. 본 논문에서는 집합 커버링 문제를 해결하기 위한 정수 계획법 기반 지역 탐색의 적용 방안을 제시하고 있다. 정수계획법 기반 지역 탐색은 이웃해를 탐색하여 현재해를 반복적으로 개선하는 지역 탐색 기법의 일종으로서 이웃해를 생성하기 위한 알고리즘으로 정수계획법을 사용한다. 본 논문에서 제시한 기법의 효과를 검증하기 위해 OR-Library의 테스트 데이터를 대상으로 실험을 수행하였다. 실험 결과, 모든 테스트 데이터에 있어서 정수계획법 기반 지역 탐색을 통해 지금까지 알려진 가장 좋은 해를 탐색할 수 있었다. 특히 4개의 테스트 데이터에 대해서는 지금까지 알려진 가장 좋은 해보다 더 좋은 해를 도출할 수 있음을 확인할 수 있었다.

▸ Keywords : 정수계획법 기반 지역 탐색, 집합 커버링 문제, 정수계획법, 지역 탐색

Abstract

The set covering problem (SCP) is one of representative combinatorial optimization problems, which is defined as the problem of covering the m-rows by a subset of the n-columns at minimal cost. This paper proposes a method utilizing Integer Programming-based Local Search (IPbLS) to solve the set covering problem. IPbLS is a kind of local search technique in which the current solution is improved by searching neighborhood solutions. Integer programming is used to generate neighborhood solution in IPbLS. The effectiveness of the proposed algorithm has been tested on OR-Library test instances. The experimental results showed that IPbLS could search for the best known solutions in all the test instances. Especially, I confirmed that IPbLS could search for better solutions than the best known solutions in four test instances.

•제1저자 : 황준하 •교신저자 : 황준하
•투고일 : 2014. 7. 22, 심사일 : 2014. 8. 11, 게재확정일 : 2014. 9. 11.
* 금오공과대학교 컴퓨터공학과(Dept. of Computer Engineering, Kumoh National Institute of Technology)
※ 본 연구는 금오공과대학교 학술연구비에 의하여 연구된 논문임

www.kci.go.kr

▸ Keywords : Integer Programming-based Local Search, Set Covering Problem, Integer Programming, Local Search

I. 서 론

집합 커버링 문제는 m행 n열의 행렬로부터 열의 일부를 선택하여 모든 행을 커버하되 비용을 최소화하는 문제로 정의된다. 행을 의미하는 집합을 $M = \{1, 2, \ldots, m\}$, 열을 의미하는 집합을 $N = \{1, 2, \ldots, n\}$이라고 할 때 집합 커버링 문제는 식 (1)~(3)과 같이 표현될 수 있다.

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

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

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

식 (1)에서 x_j는 0 또는 1의 값을 갖는 결정 변수로서 x_j 값이 1이라는 것은 열 j가 선택되었음을 의미하며, c_j는 0보다 큰 값으로 열 j의 비용을 의미한다. 따라서 목적함수인 식 (1)에 의해 선택된 열들의 총 비용을 최소화하게 된다. 식 (2)에서 a_{ij}는 m행 n열의 행렬 A의 원소로서 0 또는 1의 값을 가지며, a_{ij} 값이 1인 경우 열 j가 행 i를 커버함을 의미한다. 따라서 식 (2)로 인해 각 행은 적어도 1개 이상의 열에 의해 커버되어진다.

승무 일정 계획, 자원 배치 문제 등 실세계의 많은 문제들이 집합 커버링 문제로 표현될 수 있는데[1, 2], 집합 커버링 문제는 NP-hard 문제로 알려져 있어 지금까지 이를 해결하기 위한 많은 기법들이 제시되어 왔다. [3]에서는 분지 한계법을 핵심 알고리즘으로 활용하는 정수계획법을 적용하였다. 정수계획법은 최적해의 도출을 보장하지만 데이터의 크기가 커짐에 따라 최적해가 도출될 때까지 과도한 시간을 요구한다는 단점이 있다. 이에 따라 비교적 짧은 시간 내에 준최적해를 도출하기 위한 다양한 휴리스틱 알고리즘들이 개발되어 왔다. [4]에서는 지역 탐색 시 이웃해 생성을 위해 3개까지의 열을 교체하는 방법을 사용하였으며, [5]에서는 라그랑지안 휴리스틱을 사용하였다. [6]에서는 유전 알고리즘을 적용하였고 [7]에서는 타부 탐색을 사용하였으며 [8]에서는 개미 시스템을 활용하였다. 그리고 [9]에서는 벌 군집 알고리즘을 사용하였으며 [10]에서는 정수계획법을 단독으로 적용하되 정수계획법 자체의 성능을 향상시키기 위해 휴리스틱 방법을 적용하였다. 대부분의 기존 연구들이 실험 결과를 비교하기 위해 OR-Library[11]의 테스트 데이터를 활용하였는데 데이터에 따라 차이는 있지만 평균적으로는 [4]의 연구 결과가 현재까지도 가장 좋은 결과를 도출한 것으로 인정받고 있다.

본 논문에서는 집합 커버링 문제를 해결하기 위해 정수계획법 기반 지역 탐색의 적용 방안을 제시하고 있다. 정수계획법 기반 지역 탐색은 이웃해를 탐색하여 현재해를 반복적으로 개선시켜 나가는 지역 탐색의 일종이지만 기존의 지역 탐색과는 달리 이웃해를 생성하기 위해 정수계획법을 사용한다[12]. 정수계획법은 앞서 설명한 바와 같이 최적해의 도출을 보장하는 알고리즘으로 내부적으로는 분지 한계법을 탐색 알고리즘으로 사용하고 있다[13]. 정수계획법을 적용하기 위해서는 대상 문제가 결정 변수에 대한 1차식, 즉 선형식으로 표현되어야만 한다. 집합 커버링 문제 역시 식 (1)~(3)에서 본 바와 같이 선형식으로 표현이 가능하므로 정수계획법의 적용이 가능하다. 정수계획법은 선형 최적화 문제에 있어서 최적해를 매우 효율적으로 탐색할 수 있다. 따라서 정수계획법 기반 지역 탐색 내에서 이웃해 생성을 위해 정수계획법을 활용함으로써 더 많은 열들에 대한 교체를 고려해 볼 수 있으며, 결국 전체적인 탐색 성능도 향상될 것으로 기대된다. 실험 결과, 본 논문에서 제시한 기법을 통해 모든 데이터에 있어서 기존의 가장 좋은 해를 탐색할 수 있으며 나아가 4개의 데이터에 있어서는 기존의 가장 좋은 해보다 더 좋은 해를 도출할 수 있음을 확인할 수 있었다.

본 논문의 구성은 다음과 같다. II장에서는 먼저 일반적인 정수계획법 기반 지역 탐색에 대해 설명하며, III장에서는 집합 커버링 문제를 해결하기 위한 정수계획법 기반 지역 탐색의 적용 방안을 제시한다. IV장에서는 실험 결과를 제시하고 마지막으로 V장에서 결론 및 향후 과제에 대해 설명한다.

II. 정수계획법 기반 지역 탐색

지금까지 지역 탐색 또는 정수계획법 각각의 성능을 향상시키기 위해 지역 탐색과 정수계획법을 결합하는 방안이 많이 제안되어 왔다. 그러나 본 논문에서 적용하고자 하는 정수계

www.kci.go.kr

획법 기반 지역 탐색은 기존 연구 [12]에서 처음 제시되었다. [12]에서 제시한 일반적인 정수계획법 기반 지역 탐색 알고리즘은 그림 1과 같다.

```
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
```

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

먼저 휴리스틱 방법 또는 무작위 방법으로 초기해 X를 생성하고 이 해를 현재해로 설정한다. 여기서 X는 결정 변수 x_j의 값들로 이루어진 벡터를 의미한다. 그러고 나서 종료 조건을 만족할 때까지 정수계획법(IP)을 사용하여 이웃해를 생성하고 현재해로 설정하는 과정을 반복 수행하게 된다. IP를 수행하기 위해서는 먼저 현재해 X로부터 선택된 열들, 즉 x_j 값이 1인 열들을 대상으로 k개의 변수를 선택하게 되는데 k개의 변수를 선택하는 방법 또한 문제에 맞는 휴리스틱을 사용하거나 무작위 방법을 사용할 수 있다. k개의 변수를 선택한 후에는 x_j 값이 1인 열들 중 선택된 k개 이외의 변수들의 값을 현재값과 동일한 값, 즉 1로 고정한 후 IP를 실행하게 된다.

그림 1의 알고리즘은 일반적인 지역 탐색 기법들 중 단순 언덕 오르기 탐색과 가장 유사하다. 단순 언덕 오르기 탐색의 경우 선택된 k개의 변수들과 현재해에 포함되지 않은 열들, 즉 x_j 값이 0인 열들 중 일부를 교체함으로써 이웃해 하나를 생성하여 이동 여부를 결정하게 된다. 그러나 정수계획법 기반 지역 탐색에서는 이와는 달리 정수계획법을 사용하여 k개의 변수들과 현재해에 포함되지 않은 변수들이 가질 수 있는 모든 경우를 고려하게 된다. 예를 들어 k 값이 10이고 현재해에 포함되지 않은 열, 즉 x_j 값이 0인 변수가 10개라고 가정하자. 이때 각 변수가 가질 수 있는 값이 0 또는 1이라면 20개의 변수가 가질 수 있는 2^{20}개의 모든 경우를 고려하여 그 중에서 가장 좋은 해를 이웃해로 결정하게 된다. 물론 고려 대상에는 현재해도 포함되기 때문에 항상 현재해와 같거나 더 좋은 해가 생성된다. 따라서 현재해와 동일한 해가 반복적으로 생성되는 상황을 피하기 위해서는 별도의 조치가 필요할 수도 있다. 정수계획법은 선형식으로 표현되는 문제에 대해 최적해를 매우 효율적으로 탐색하기 때문에 k 값이 비교적 큰 경우에도 빠른 시간 내에 최적해를 찾을 수 있다는 장점이 있다.

정수계획법 기반 지역 탐색은 일종의 메타 휴리스틱 탐색 알고리즘으로서 선형적으로 표현될 수 있는 문제라면 쉽게 적용이 가능하다. [12]에서는 자체적으로 만든 문제인 N-Queens 최대화 문제를 대상으로 정수계획법 기반 지역 탐색을 비롯한 기존의 탐색 알고리즘들을 적용해 봄으로써 정수계획법 기반 지역 탐색의 효과를 검증하였다. [14]에서는 실세계 문제인 최대 커버링 문제를 해결하기 위해 정수계획법 기반 지역 탐색을 적용하였으며, [15]에서는 다차원 배낭 문제를 해결하기 위해 정수계획법 기반 지역 탐색을 적용하였다. 특히 [15]에서는 OR-Library의 테스트 데이터에 대한 검증 결과, 정수계획법 기반 지역 탐색을 통해 기존의 가장 뛰어난 연구 결과들과 비슷하거나 더 좋은 결과를 도출할 수 있는 것으로 나타났다.

III. 집합 커버링 문제를 위한 정수계획법 기반 지역 탐색

1. 제안 기법의 전체적인 구조

본 논문에서 제안하는 집합 커버링 문제를 위한 정수계획법 기반 지역 탐색의 전체적인 구조는 그림 2와 같다. 먼저 그리디 휴리스틱을 사용하여 초기해를 생성한 후 현재해로 설정한다. 그리고 현재해에 포함된 열들 중 다음 IP 수행 시 제외될 열들의 개수인 k의 값을 1로 설정한다. 그러고 나서 현재해로부터 k개의 열을 선택하고 선택된 열들의 정보를 활용하여 문제를 축소하며 이 문제를 대상으로 정수계획법을 적용함으로써 이웃해를 생성하는 과정을 반복적으로 수행하게 된다. 참고로 초기해 생성 방법과 k개의 변수를 선택하는 방법, 그리고 문제 축소 방법에 대해서는 각각 본 장의 2, 3, 4절을 통해 보다 자세히 설명한다.

정수계획법 기반 지역 탐색에 있어서 k의 값은 지역 탐색 시 교체될 열들의 개수를 의미하는 것으로서 이 값이 작으면 지역 최적해에 빠질 가능성이 높아지며 이 값이 크면 IP 실행 시 최적해가 생성될 때까지의 수행 시간이 길어지게 된다. 본 논문에서는 IP의 실행 시간 t를 고정하고 이 시간 내에 최적해가 도출되었다면 k의 값을 증가시키고 그렇지 않다면 k의

www.kci.go.kr

값을 감소시킴으로써 실행 시간에 적절한 k의 값을 찾도록 하였다. 따라서 IP의 실행 시간 t가 본 기법의 성능을 좌우하는 유일한 파라미터라 할 수 있다.

```
Algorithm IPbLS for SCP
    X1 : The column set { j | xj = 1, j ∈ N }.
    t : Execution time for IP.
    tcur : Current real execution time of IP.
Begin
    X = Make an initial solution using a greedy heuristic
    k = 1
    While stopping condition is not met Do
        Select k variables stochastically from X1
        Reduce problem with unselected variables of X1
        Add Obj and all constraints to IP
            • Fix values of unselected variables of X1 to 1
            • Set maximum execution time to (2 × t)
            • Add the constraint that the new solution
              X is not equal to the current solution X
        X = Make a neighbor solution with IP
        If tcur ≤ t Then
            k = k + 1
        Else
            k = k - 1
        End If
    End While
    Return X
End Begin
```

그림 2. 집합 커버링 문제를 위한 정수계획법 기반 지역 탐색
Fig. 2. Integer Programming-based Local Search for the Set Covering Problem

집합 커버링 문제를 위해 IP 수행 시 새로 추가된 제약조건은 다음과 같이 두 가지가 있다. 첫 번째는 IP의 최대 수행 시간을 $(2 \times t)$로 설정하는 것이다. k 값이 커질수록 IP의 실행 시간 또한 늘어나게 되는데, 경우에 따라서는 k 값이 1 커짐에 따라 IP가 최적해를 생성하기까지의 실행 시간이 과도하게 늘어날 수도 있다. 이 경우를 대비하여 IP의 최대 실행 시간을 $(2 \times t)$로 설정하였다. 하지만 여전히 최적해를 도출하지 못한 채 IP의 실행이 종료될 수 있으며 이에 따라 현재해보다 좋지 않은 해가 도출될 수도 있다. 그렇다 하더라도 본 연구에서는 좋지 않은 해로의 이동을 허용함으로써 탐색이 국소 최적해에 빠지는 경향을 방지하였다. 두 번째는 국소 최적해를 방지하는 또 한 가지 방법으로 IP를 통해 현재해와 동일한 해가 생성되지 않도록 하는 제약조건을 추가하였다.

2. 초기해 생성 방법

그림 2의 알고리즘에서 초기해 생성 방법으로는 모든 행이 커버될 때까지 열을 하나씩 추가해 나가는 그리디 휴리스틱을 사용하였으며, 구체적인 알고리즘은 그림 3과 같다.

```
Algorithm GenerateInitialSolution
    X : Variable vector (Current solution).
    hj : The number of newly covered rows by column j.
    N : Column set { 1, 2, ..., n }.
Begin
    Reset all variables in X to 0
    While all rows are not covered by X Do
        Choose j ∈ N minimizing the value (cj / hj)
        Set xj to 1
        N = N - { j }
    End While
    While there exists a redundant column in X Do
        Choose a redundant column j maximizing cj
        Reset xj to 0
    End While
    return X
End Begin
```

그림 3. 초기해 생성 알고리즘
Fig. 3. Initial Solution Generation Algorithm

첫 번째 while 루프를 통해 모든 행이 커버될 때까지 열을 하나씩 추가하게 되는데 열을 선택하는 기준은 (c_j / h_j)의 값이 가장 작은 열 j를 선택하는 것이다. 여기서 c_j는 열 j의 비용을 의미하며 h_j는 현재까지 커버되지 않는 행들 중 열 j에 의해 커버되는 행들의 개수를 의미한다. 즉, 비용이 적고 커버되지 않은 행들을 많이 커버하는 열이 우선적으로 선택되어진다. 만약 (c_j / h_j)의 값이 동일한 열들이 존재할 경우에는 무작위로 하나를 선택한다.

첫 번째 while 루프가 종료되면 모든 행이 커버되는 열의 집합이 생성된다. 그런데 경우에 따라서는 특정 열 j에 의해 커버되는 모든 행들이 다른 열들에 의해 중복으로 커버되는 상황이 발생할 수 있다. 이때 열 j는 모든 행들을 커버하는 데 불필요한 열이므로 현재해 X로부터 제거함으로써 목적함수 값을 향상시킬 수 있다. 두 번째 while 루프를 통해 이와 같은 열들을 모두 제거할 때까지 하나씩 제거하고 있는데, 목적함수 값을 보다 최소화하기 위해 비용 c_j의 값이 큰 것부터 하나씩 차례로 제거하게 된다.

3. 확률적 방법에 의한 k개의 변수 선택 방법

초기해가 생성되면 이 해를 현재해로 변경한다. 다음으로는 현재해에 포함된 열, 즉 결정 변수 x_j 값이 1인 열들을 대상으로 k개의 열들을 선택한 후 이 열들의 값을 무효화하고 다시 정수계획법을 호출한다. 다시 말하면 현재해에서 결정변수 x_j 값이 1인 열들 중 선택되지 않은 열들의 값을 1로 고

www.kci.go.kr

정한 후 정수계획법을 다시 수행하는 것이다. 중요한 것은 k개의 열들을 선택하는 기준으로서 다양한 방법을 고려해 볼 수 있다. 예를 들면, k개의 열들을 무작위로 선택하는 방법 또한 한 가지 방법이 될 수 있다. 그러나 해의 개선에 대한 방향성을 제시해 주지 못하기 때문에 상대적으로 해의 개선 속도가 느릴 수밖에 없다.

본 연구에서는 k개의 열을 선택하는 방법으로 확률적인 방법을 사용하였으며, x_j 값이 1인 열 j가 선택될 확률 $p(j)$는 식 (4)와 같다. 여기서 n_1은 현재해에서 x_j 값이 1인 열들의 개수이고, h_i는 열 i를 현재해에서 제거함으로써 발생하는 공백행, 즉 커버되지 못하는 행들의 개수를 의미한다. $p(j)$의 확률은 $c_j/(h_j+1)$에 비례하게 되는데 해당 열 j의 비용이 클수록, 그리고 열 j로 인해 발생하는 공백행이 적을수록 선택될 확률은 커지게 된다. c_j 값이 클수록 현재해의 목적 함수에 도움이 되지 못하며 h_j 값이 작을수록 전체 행을 커버하는 데 도움이 되지 못하기 때문이다. h_j 값에 1을 더한 이유는 h_j 값이 0이 되는 경우를 대비하기 위함이다. h_j 값이 0이란 말은 사실상 열 j가 현재해 내에서 불필요한 열임을 의미하기 때문에 바로 열 j를 제거해도 무방하다. 그러나 여기서는 식 (4)의 확률에 의해 자연스럽게 선택되어 제거될 수 있도록 하였다.

$$p(j) = \frac{c_j/(h_j+1)}{\sum_{i=1}^{n_1}(c_i/(h_i+1))} \quad (4)$$

k개의 열을 선택하는 방법으로 $c_j/(h_j+1)$의 값이 큰 열부터 차례로 선택하는 그리디 방법을 사용할 수도 있다. 그러나 이 경우 선택된 열들이 또다시 반복적으로 선택되는 경향이 있어 탐색이 다양화되지 못하고 국소 최적해에 머무는 현상이 발생하게 된다.

4. *IP* 실행을 위한 문제 축소 방법

주어진 집합 커버링 문제는 m개의 행과 n개의 열로 이루어진다. 그러나 현재해로부터 k개의 변수가 선택되면 정수계획법 수행 시 x_j 값이 1인 열들 중 선택되지 못한 열들은 다음 현재해에서도 x_j 값이 1로 고정되기 때문에 이 정보를 이용하면 정수계획법 수행을 위한 대상 문제를 대폭 축소할 수 있게 된다. 문제를 축소함으로써 정수계획법을 통해 보다 빠르게 최적해를 찾을 수 있도록 하는 것이다.

문제를 축소하는 과정은 그림 4의 예와 같으며 이는 기존 연구 [14]에서의 문제 축소 과정과 유사하다. 예를 들어 6개의 행과 10개의 열로 이루어진 집합 커버링 문제가 있다고 가정하자. 현재해로부터 선택되지 못한 열이 그림 4(a)와 같이 열 1과 열 6이라고 하면 행 1, 2, 4는 열 1과 6에 의해 커버되기 때문에 더 이상 해당 행들에 대한 커버를 고려할 필요가 없다. 따라서 그림 4(b)와 같이 행의 개수를 축소할 수 있다. 그리고 축소된 행들을 대상으로 열 집합으로부터 *IP*에 참여할 열들을 선택하게 되는데 열 1, 6을 제외한 8개의 열들로부터 선택하게 된다. 이때 역시 행 3, 5, 6만을 대상으로 이 행들을 커버하는 열들을 선택하면 된다. 만약 열 4, 5, 8이 행 3, 5, 6 중 1개 이상의 행을 커버하고 있다면 최종적으로 *IP*의 대상 문제는 그림 4(c)와 같이 3행 3열로 이루어진 집합 커버링 문제로 축소된다.

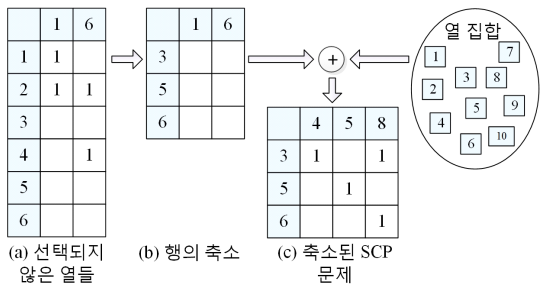

그림 4. 문제 축소 과정
Fig. 4. Problem Reduction Steps

IV. 실험 결과

1. 실험 환경

본 연구에서는 정수계획법 개발 도구로 IBM ILOG CPLEX 12.6을 사용하였다[16]. CPLEX는 정수계획법 개발을 위한 상용 라이브러리로서 현재 상업적 또는 학술적 목적으로 가장 많이 사용되고 있으며 그 성능 또한 우수한 것으로 인정받고 있다. 본 논문의 모든 실험은 Intel i7-4770 3.4GHz, 8GB RAM PC 및 Windows 7 64비트 운영체제 상에서 수행되었다.

실험 데이터로는 OR-Library에 포함된 77개의 테스트 문제들을 대상으로 하였다. 그런데 이 문제들 중에는 정수계획법만으로 최적해의 도출이 가능한 문제들이 포함되어 있다. 먼저 77개의 테스트 문제들을 대상으로 18,000초(5시간) 내에 정수계획법만으로 최적해의 도출이 가능한지를 확인하였다. 실험 결과, 9개를 제외한 모든 문제들에 대해 최적해의 도출이 가능하였다. 본 논문에서는 정수계획법만으로는 최적해의 도출이 어려운 9개의 테스트 문제와 RAIL2536 문제를

www.kci.go.kr

포함하여 총 10개의 테스트 문제를 대상으로 하였다. RAIL2536의 경우 종전까지 정수계획법만으로는 최적해의 도출이 어려웠던 문제로서 기존 연구와의 비교를 위해 포함시켰다. 표 1에서 Instance는 각 문제의 이름, m은 행의 개수, n의 열의 개수, BKS(Best Known Solution)는 지금까지 알려진 가장 좋은 해의 목적함수 값, IP의 z는 정수계획법만을 통해 도출한 가장 좋은 해의 목적함수 값을 의미하며 IP의 time은 z 값이 도출될 때까지의 초 단위 수행 시간을 의미한다. BKS와 같거나 더 좋은 결과에 대해서는 볼드체 및 음영으로 표시하였다. 상대적으로 규모가 작은 데이터인 SCPNRG5~SCPNRH5의 경우 정수계획법만으로도 BKS에 근접한 해를 찾을 수 있음을 알 수 있지만, 대규모 문제인 RAIL2586, RAIL4284, RAIL4872에 대해서는 정수계획법만으로는 BKS에 근접한 해를 찾기 어려움을 알 수 있다.

표 1. 실험 데이터
Table 1. Experimental Data

Instance	m	n	BKS	IP	
				z	time(sec)
SCPNRG5	1000	10000	168	**168**	77
SCPNRH1	1000	10000	63	64	981
SCPNRH2	1000	10000	63	64	80
SCPNRH3	1000	10000	59	**59**	8377
SCPNRH4	1000	10000	58	**58**	5601
SCPNRH5	1000	10000	55	**55**	13
RAIL2536	2536	1081841	691	**+689**	317
RAIL2586	2586	920683	945	957	13923
RAIL4284	4284	1092610	1063	1079	17259
RAIL4872	4872	968672	1528	1557	17075

+최적해를 의미함

2. 중규모 데이터에 대한 실험 결과

먼저 중규모 데이터라 할 수 있는 SCPNRG5~SCPNRH5를 대상으로 그림 2의 알고리즘에서 *IP*의 실행 시간인 t의 값에 따른 성능의 변화를 살펴보았다. 표 2는 t 값이 0.5초, 1초, 5초, 10초, 15초, 20초일 때의 실험 결과이다. 각각의 실험 데이터와 t 값에 대해 총 5회의 실험을 수행하였으며, 각 실험 당 수행 시간은 1,800초로 제한하였다. $\bar{z}$는 각 실험을 통해 도출된 해의 목적함수 값의 평균을 의미한다.

표 2에 의하면 SCPNRH1을 제외한 모든 데이터에 있어서 t의 값에 상관없이 항상 지금까지 알려진 가장 좋은 결과인 BKS를 탐색할 수 있음을 알 수 있다. time은 BKS의 결과를 도출할 때까지 소요된 평균 시간(초)을 나타낸 것으로 BKS를 도출하지 못한 경우에는 생략하였다. BKS를 찾기까지 소요 시간이 가장 짧은 t 값의 결과에 대해서는 볼드체 및 음영으로 표시하였다. SCPNRG5, SCPNRH2, SCPNRH3, SCPNRH4, SCPNRH5의 경우 t 값이 각각 15, 1, 0.5, 1, 0.5일 때 가장 빨리 BKS의 결과를 도출할 수 있었다. SCPNRG5를 제외한 모든 데이터에 있어서 비교적 t 값이 작을 때 BKS의 결과를 보다 빨리 찾을 수 있었다. 그러나 t 값이 0.5라 하더라도 *IP*를 통한 이웃해를 생성하기 위해 선택되는 변수의 개수, 즉 k의 값은 일반적인 지역 탐색에서의 이웃해 생성 시 변경되는 변수의 개수보다 훨씬 큰 값을 갖게 된다. 예를 들면 SCPNRG5의 경우 준최적해에 포함된 변수의 개수가 약 105개 정도인데 *IP*를 수행하기 위해 약 60개까지의 변수가 선택되어지며, SCPNRH2의 경우 약 55개의 변수들 중 약 28개까지의 변수가 선택되어진다. SCPNRH1의 경우 t 값이 1 또는 10일 때 모든 실험에 있어서 BKS의 결과를 도출할 수 있었으며 그 외의 값일 때는 BKS의 값을 도출하지 못하는 경우도 있었다. 본 실험을 통해 t의 값에 따라 정수계획법 기반 지역 탐색의 성능에 차이가 날 수 있음을 알 수 있다.

표 3은 정수계획법 기반 지역 탐색(IPbLS)과 지금까지 가장 좋은 결과로 인정되는 기존 연구 [4]의 실험 결과를 비교한 것이며, 정수계획법만을 사용했을 때의 결과(IP)를 함께 표시하였다. IPbLS의 결과는 SCPNRG5, SCPNRH1, SCPNRH2, SCPNRH3, SCPNRH4, SCPNRH5에 대한 t의 값으로 각각 15, 1, 1, 0.5, 1, 0.5를 적용한 것이며, 표 2의 실험 결과를 포함하여 총 10회 실험을 수행한 후의 평균값을 나타낸 것이다. [4]의 실험 결과에서 #BKS는 총 10회의 실험 중 BKS의 결과를 도출한 실험 횟수를 나타낸 것이다. [4]와 IPbLS 모두 SCPNRH1과 SCPNRH2에 있어서 IP보다 더 좋은 해를 보다 짧은 시간 내에 찾을 수 있었다. 그런데 [4]에서는 SCPNRH2, SCPNRH3, SCPNRH4에 있어서 BKS의 결과를 도출하지 못한 경우도 있으나 IPbLS의 경우 모든 데이터들에 있어서 매 실험마다 BKS의 결과를 도출할 수 있었으며 BKS의 값을 도출하기까지 수행 시간도 비교적 짧음을 알 수 있다. 사실은 [4]에서는 수행 시간을 180초로 제한하였고 실험 환경 또한 다르기 때문에 어떤 기법이 우수한지를 절대적으로 판단하기는 힘든 상황이다. 그렇다 하더라도 IPbLS의 경우 일반적인 지역 탐색의 단점인 지역 최적화 현상을 극복함으로써 결국 더 좋은 해를 찾을 수 있음을 알 수 있다.

www.kci.go.kr

표 2. 중규모 데이터에 대한 실험 결과
Table 2. Experimental Result on the Mid-sized Data

Instance	BKS	t = 0.5		t = 1		t = 5		t = 10		t = 15		t = 20	
		$\bar{z}$	time	$\bar{z}$	time	$\bar{z}$	time	$\bar{z}$	time	$\bar{z}$	time	$\bar{z}$	time
SCPNRG5	168	168	43.8	168	52	168	56.4	168	67.2	168	**27.6**	168	29.8
SCPNRH1	63	63.2	-	63	**673.2**	63.6	-	63	807.2	63.6	-	64	-
SCPNRH2	63	63	144.2	63	**73**	63	200.8	63	514	63	436.4	63	398.4
SCPNRH3	59	59	**176**	59	478	59	491.6	59	209.2	59	754.4	59	603.8
SCPNRH4	58	58	84.6	58	**69.8**	58	96.2	58	132	58	276.2	58	87.4
SCPNRH5	55	55	**10.4**	55	19.4	55	19.6	55	26.8	55	21.4	55	70.2

표 3. 중규모 데이터에 대한 다른 알고리즘과의 비교
Table 3. Comparison with Other Algorithms on the Mid-sized Data

Instance	BKS	IP		[4]		IPbLS	
		z	time	$\bar{z}$	#BKS	$\bar{z}$	time
SCPNRG5	168	**168**	77	**168**	10	**168**	29.4
SCPNRH1	63	64	981	**63**	10	**63**	782.4
SCPNRH2	63	64	80	63.3	7	**63**	68.3
SCPNRH3	59	**59**	8377	59.9	1	**59**	135.8
SCPNRH4	58	**58**	5601	**58**	10	**58**	45.2
SCPNRH5	55	**55**	13	55.4	6	**55**	15.2

3. 대규모 데이터에 대한 실험 결과

표 4는 대규모 데이터라 할 수 있는 RAIL 데이터를 대상으로 *IP*의 실행 시간인 t 값이 5초, 10초, 20초, 30초, 40초, 50초일 때의 실험 결과를 나타낸 것이다. 각각의 실험 데이터와 t의 값에 대해 총 3회의 실험을 수행하였으며, 각 실험 당 수행 시간은 18,000초로 제한하였다. 중규모 데이터와는 달리 매 실험마다 BKS의 결과를 도출하는 것이 어려웠으므로 표 4에는 평균값($\bar{z}$)과 최소값(Min)을 표시하였으며, 평균값과 최소값 중 가장 좋은 결과를 볼드체 및 음영으로 표시하였다.

표 4는 과도한 실험 소요 시간을 감안하여 단 3회의 실험을 수행한 결과이지만 t 값에 따른 성능의 변화에 대한 특성을 비교적 잘 보여주고 있다. RAIL2536의 경우 t 값이 30, 40, 50일 때 매 실험 시마다 최적해를 도출할 수 있었다. 정수계획법만으로도 최적해를 도출할 수 있었으므로 t 값이 충분히 큰 경우, 즉 k 값이 충분히 큰 경우 최적해의 도출이 가능한 것으로 판단된다. RAIL2586의 경우 t 값이 20일 때 가장 좋은 결과를 보였으며 이 값을 기준으로 t 값이 커지거나 작아질수록 결과가 더 좋지 않았다. RAIL4284의 경우 t 값이 클수록 더 좋은 결과를 보였다. 그러나 이 사실만으로 t 값이 50보다 큰 경우 더 좋은 결과가 도출될 수 있다고 단정하기는 힘들다. t 값이 가장 큰 경우는 순수하게 정수계획법만을 적용한 경우라 할 수 있다. 그런데 표 1에서 RAIL4284에 대해 정수계획법만을 적용 결과가 1079임을 볼 때 t 값이 커짐에 따라 주어진 시간 내에 더 좋지 않은 결과를 보이는 순간이 있을 것으로 예상할 수 있다. RAIL4872의 경우 t 값이 40일 때 가장 좋은 결과를 보였으며 이 값을 기준으로 t 값이 작아지거나 커질수록 더 좋지 않은 결과를 보이고 있다. 주목할 만한 사실은 RAIL2586을 제외한 데이터에 있어서 BKS보다 더 좋은 해를 찾을 수 있었다는 것이다. RAIL2536의 경우 정수계획법 자체의 발전으로 인해 정수계획법만으로도 최적해를 찾을 수 있었으므로 이를 제외한다 하더라도 RAIL4284와 RAIL4872의 경우 기존의 어떤 탐색 기법을 통해서도 찾지 못한 해를 도

표 4. 대규모 데이터에 대한 실험 결과
Table 4. Experimental Result on the Large-sized Data

Instance	BKS	t = 5		t = 10		t = 20		t = 30		t = 40		t = 50	
		Min	$\bar{z}$	Min	$\bar{z}$	Min	$\bar{z}$	Min	$\bar{z}$	Min	$\bar{z}$	Min	$\bar{z}$
RAIL2536	691	690	690.7	690	690.0	689	689.7	**689**	**689.0**	**689**	**689.0**	**689**	**689.0**
RAIL2586	945	947	948.3	946	947.0	**945**	**946.7**	946	947.3	**945**	947.3	947	948.3
RAIL4284	1063	1068	1068.7	1066	1067.3	1065	1065.0	1063	1064.0	1063	1063.7	**1062**	**1063.0**
RAIL4872	1528	1533	1533.3	1529	1530.3	**1527**	1528.7	1528	1528.3	**1527**	**1528.0**	1529	1530.0

출할 수 있었다. 이는 본 논문에서 제안하는 정수계획법 기반 지역 탐색이 기존 지역 탐색 기법의 가장 큰 문제점인 지역 최적화 문제를 효과적으로 극복할 수 있음을 보여주는 것이다.

표 5는 정수계획법 기반 지역 탐색과 지금까지 가장 좋은 결과로 인정되는 기존 연구 [4]의 실험 결과를 비교한 것이며, 정수계획법만을 사용했을 때의 결과(IP)를 함께 표시하였다. IPbLS의 결과는 RAIL2536, RAIL2586, RAIL4284, RAIL4872에 대한 t 값으로 각각 40, 20, 50, 40을 적용한 것이다. RAIL2536의 경우 t 값이 40일 때 가장 빨리 최적해를 도출할 수 있었다. 표 5의 결과는 표 4의 실험 외에 7회의 실험을 추가하여 총 10회의 실험을 수행한 결과를 나타낸 것이다.

표 5. 대규모 데이터에 대한 다른 알고리즘과의 비교
Table 5. Comparison with Other Algorithms on the Large-sized Data

Instance	BKS	IP	[4]		IPbLS	
		z	Min	$\bar{z}$	Min	$\bar{z}$
RAIL2536	691	**689**	691	691.1	**689**	**689**
RAIL2586	945	957	945	946.9	**944**	**946.5**
RAIL4284	1063	1079	1064	1065.7	**1062**	**1063.3**
RAIL4872	1528	1557	1528	1531.8	**1027**	**1528.5**

[4]에서의 실험 또한 총 10회의 실험을 수행한 결과이며 각 실험 당 수행 시간은 본 논문의 실험과 마찬가지로 18,000초로 제한한 것이다. IPbLS와 [4]의 평균값을 비교해 보면 모든 데이터에 있어서 IPbLS의 성능이 훨씬 뛰어남을 알 수 있다. 또한 최소값에 있어서도 IPbLS는 [4]보다 더 좋은 결과를 도출할 수 있었다. 더군다나 IPbLS는 모든 데이터에 있어서 BKS보다 더 좋은 결과를 도출할 수 있었음을 알 수 있다. 본 실험을 통해 본 논문에서 제안하는 정수계획법 기반 지역 탐색의 우수성을 보다 확실하게 확인할 수 있다.

본 논문에서 제안하는 정수계획법 기반 지역 탐색은 선형적으로 모델링이 가능한 문제, 즉 정수계획법의 적용이 가능한 문제를 대상으로 하고 있다. 실험 결과에 의하면 정수계획법만으로 어느 정도의 성능을 발휘한다면 정수계획법 기반 지역 탐색 또한 탁월한 성능을 발휘할 가능성이 높을 것으로 예상된다. 물론 대상 문제 또는 데이터에 따라서는 기본적으로 정수계획법만 적용할 수 있다면 정수계획법 기반 지역 탐색의 적용을 통한 우수한 해의 도출을 기대할 수 있을 것으로 판단된다.

V. 결 론

본 논문에서는 정수계획법 기반 지역 탐색을 활용하여 집합 커버링 문제를 해결하는 방안을 제시하였다. 실험 결과, 모든 실험 데이터에 있어서 기존의 가장 좋은 해를 도출할 수 있었으며 4개의 데이터에 있어서는 기존의 가장 좋은 해보다 더 좋은 해를 도출할 수 있었다. 정수계획법 기반 지역 탐색은 지역 탐색 수행 시 이웃해를 생성하기 위해 정수계획법을 활용하는 방법이다. 본 논문에서는 이와 같이 지역 탐색과 정수계획법의 단순한 결합만으로도 기존의 지역 탐색 기법들의 문제점인 지역 최적화 현상을 효과적으로 극복할 수 있음을 확인하였다.

대부분의 탐색 기법들의 경우 탐색 성능에 영향을 미칠 수 있는 다수의 파라미터들이 존재할 수 있으며 해당 파라미터의 값을 설정하는 것 또한 매우 어렵다. 본 논문에서 제시하는 정수계획법 기반 지역 탐색 알고리즘의 파라미터는 단 하나로서 이웃해 생성을 위한 정수계획법의 실행 시간이 그것이다. 결국 정수계획법 기반 지역 탐색의 실행 도중에 정수계획법의 실행 시간이 변경될 수 있어야 하는데, 예를 들면 현재해보다 더 좋은 해의 도출 여부에 따라 정수계획법의 실행 시간을 증가 또는 감소시키는 방법을 고려해 볼 수 있다. 향후 이 방법을 포함하여 파라미터의 값을 자동으로 설정하기 위한 추가 연구가 필요할 것으로 판단된다. 아울러 보다 다양한 조합 최적화 문제로의 적용을 통해 정수계획법 기반 지역 탐색의 우수성 및 실용성을 재검증할 필요가 있다.

참고문헌

[1] B. Gopalakrishnan, and E. Johnson, "Airline Crew Scheduling: State-of-the-Art", Anals of Operations Research, Vol. 140, No. 1, pp.305-337, Nov. 2005.

[2] R.Z. Farahani, N. Asgari, N. Heidari, M. Hosseininia, and M. Goh, "Covering Problems in Facility Location: A Review", Computers & Industrial Engineering, Vol. 62, No. 1, pp.368-407, Feb. 2012.

[3] E. Balas, and M.C. Carrera, "A Dynamic Subgradient-based Branch-and-bound Procedure for Set Covering", Operations Research, Vol. 44,

www.kci.go.kr

No. 6, pp.875-890, Dec. 1996.
〔4〕 M. Yagiura, M. Kishida, and T. Ibaraki, "A 3-flip Neighborhood Local Search for the Set Covering Problem", European Journal of Operational Research, Vol. 172, No. 2, pp.472-499, July 2006.
〔5〕 A. Caprara, M. Fischetti, and P. Toth, "A Heuristic Method for the Set Covering Problem", Operations Research, Vol. 47, No. 5, pp.730-743, Oct. 1999.
〔6〕 J.E. Beasley, and P.C. Chu, "A Genetic Algorithm for the Set Covering Problem", European Journal of Operational Research, Vol. 94, No. 2, pp.392-404, Oct. 1996.
〔7〕 M. Caserta, "Tabu Search-Based Metaheuristic Algorithm for Large-scale Set Covering Problems", Metaheuristics: Progress in Complex Systems Optimization, Springer, pp.43-63, 2007.
〔8〕 Z. Ren, Z. Feng, L. Ke, and Z. Zhang, "New Ideas for Applying Ant Colony Optimization to the Set Covering Problem", Computers & Industrial Engineering, Vol. 58, No. 4, pp.774-784, May 2010.
〔9〕 B. Crawford, R. Soto, R. Cuesta, and F. Paredes, "Application of the Artificial Bee Colony Algorithm for Solving the Set Covering Problem", The Scientific World Journal, 2014.
〔10〕 M. Yaghini, M.R. Sarmadi, and M. Momeni, "A Local Branching Approach for the Set Covering Problem", International Journal of Industrial Engineering & Production Research, Vol. 25, No. 2, pp.95-102, June 2014.
〔11〕 J.E. Beasley, "OR-Library: Distributing Test Problems by Electronic Mail", Journal of the Operational Research Society, Vol. 41, No. 11, pp.1069-1072, Nov. 1990.
〔12〕 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.
〔13〕 L.A. Wolsey, "Integer Programming", Wiley, pp. 91-111, 1998.
〔14〕 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.
〔15〕 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.
〔16〕 "IBM ILOG CPLEX Optimization Studio CPLEX User's Manual, Version 12 Release 6", International Business Machines Corporation, 2013.

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

www.kci.go.kr