韓國 컴퓨터情報學會 論文誌
第11卷 第1號, 2006. 3.

2006-11-1-1-14

이웃해 탐색 기법을 이용한 Maximal Covering 문제의 해결

황 준 하*

Neighborhood Search Algorithms for the Maximal Covering Problem

Junha Hwang*

요 약

지금까지 maximal covering 문제를 해결하기 위해 다양한 기법들이 적용되어 왔다. 타부 탐색 역시 그 중의 하나이다. 그러나 기존 연구에서는 타부 탐색을 비롯한 언덕오르기 탐색이나 시뮬레이티드 어닐링과 같은 이웃해 탐색 기법들에 대한 종합적인 분석과 성능 향상을 위한 노력이 부족하였다. 본 논문에서는 다양한 실험과 분석을 통해 이웃해 탐색 기법들의 성능을 향상시키기 위한 방안을 소개한다. 기본적으로 모든 이웃해 탐색 기법들은 k-exchange 이웃해 생성 방법을 사용하고 있으며 다양한 파라미터 설정에 따라 각 기법의 성능이 어떻게 달라지는가를 분석하였다. 실험 결과 단순 언덕오르기 탐색과 시뮬레이티드 어닐링이 다른 기법들에 비해 훨씬 우수한 탐색 성능을 보였으며, 일반적인 경우와는 달리 단순 언덕오르기 탐색이 시뮬레이티드 어닐링과 비슷한 성능을 보임을 확인하였다.

Abstract

Various techniques have been applied to solve the maximal covering problem. Tabu search is also one of them. But, existing researches were lacking of the synthetic analysis and the effort for performance improvement about neighborhood search techniques such as hill-climbing search and simulated annealing including tabu search. In this paper, I introduce the way to improve performance of neighborhood search techniques through various experiments and analyses. Basically, all neighborhood search algorithms use the k-exchange neighborhood generation method. And I analyzed how the performance of each algorithm changes according to various parameter settings. Experimental results have shown that simple hill-climbing search and simulated annealing can produce better results than any other techniques. And I confirmed that simple hill-climbing search can produce similar results as simulated annealing unlike general case.

▸ Keyword : neighborhood search(이웃해 탐색), hill-climbing search(언덕오르기 탐색), simulated annealing(시뮬레이티드 어닐링), maximal covering problem(maximal covering 문제)

• 제1저자 : 황준하
• 접수일 : 2006.02.23, 심사완료일 : 2006.03.20
* 금오공과대학교 컴퓨터공학부 조교수
※ 본 논문은 금오공과대학교 학술연구비에 의하여 연구된 논문임.

Ⅰ. 서 론

Maximal covering 문제(MCP)는 0과 1로 이루어진 $m \times n$ 행렬이 주어진 경우 n개의 열로부터 주어진 p개의 열을 선택하되 가장 많은 행을 cover하는 문제로 정의된다[1]. (그림 1)은 5행 4열로 이루어진 maximal covering 문제의 예로서 각 열마다 서로 다른 행을 cover하고 있다. 여기서 p의 값이 2로 주어진다면 즉, 2개의 열을 선택하여 가장 많은 행을 cover하는 경우는 열 1과 4를 선택하는 것이다.

	1	2	3	4
1	1	1		
2		1	1	1
3				
4	1		1	
5				1

그림 1. maximal covering 문제의 예
Fig 1. An example of maximal covering problem

실세계의 많은 문제들이 maximal covering 문제로 표현될 수 있는데 대표적인 응용 문제로는 승무일정계획 문제와 위치선정 문제를 들 수 있다. 승무일정계획은 특정 기간 동안 운행해야 할 차량들을 대상으로 각 차량마다 필요로 하는 승무원을 배정하는 계획으로서, 실제 응용 예로는 항공을 비롯한 철도, 지하철, 버스 등의 승무일정계획이 있다[2, 3]. 열차 운행을 위한 승무일정계획 문제를 maximal covering 문제로 모델링하면 승무원별 근무표가 열이 되고 계획기간 내의 열차운행 구간(근무 교대가 가능한 역 사이의 구간)들이 행이 된다. 근무표는 한 승무원이 실제로 근무할 수 있는 소정 개수의 열차운행 구간들의 집합으로서 근무와 관련한 여러 제약조건들을 만족하는 것이다. 문제는 이러한 개별 근무표 즉 생성 가능한 열들의 종류가 대단히 많고 동원 가능한 승무원의 수가 제한되어 있을 경우, 근무표들 중 어떤 조합으로 주어진 개수만큼의 열들을 뽑아야 가장 많은 열들이 cover될 수 있느냐 하는 것이다. 이와 같이 방대한 열들의 풀(pool)로부터 가장 많은 행들을 cover하는 열들의 최적 조합을 찾는다는 것은 복잡도가 매우 높은 문제이다.

위치선정 문제는 고정된 자원을 사용하여 최대한 많은 서비스를 제공해 줄 수 있는 위치들을 선정하는 문제로서 생산 현장에 장비를 설치하거나 병원과 같은 복지 시설을 설치하는 등의 실세계 문제에 적용되어 왔다[4]. 이와 같은 maximal covering 문제는 NP-hard 문제로서[5], 행과 열의 수가 방대하고 다양한 제약조건들을 만족해야 하는 대규모의 실세계 문제가 주어질 경우 최적해를 찾는 것이 사실상 불가능하다.

Maximal covering 문제를 해결하기 위한 종래의 연구에서는 정수계획법(Integer Programming)이나 라그랑지안 완화법(Lagrangian Relaxation)이 많이 적용되었다. 이들 중 정수계획법은 최적해를 보장해 주는 대신 문제의 규모가 조금만 커져도 해를 구하는 것 자체가 어려워지기 때문에 현실적 규모의 문제에는 적용이 불가능하다. 따라서 라그랑지안 완화법과 같은 휴리스틱 기법이 그 대안으로 제시되기는 했지만 이 또한 대규모 문제가 주어질 경우 그다지 만족스러운 성능을 보이지 못하고 있다. 더구나 선형적 모델을 기반으로 하고 있어 비선형적 목적함수 또는 제약조건을 요구하는 문제에 있어서는 적용 자체가 불가능하다는 단점이 있다.

최근 들어서는 대안으로 이웃해 탐색 기법이나 유전알고리즘을 활용한 연구 결과가 보고된 바 있다. 기존 연구 [3]에서는 이웃해 탐색 기법과 정수계획법을 하이브리드 형태로 결합하여 적용하는 방안을 제시하였다. 또한 기존 연구[6]에서는 비발현 유전자의 개념을 활용하여 염색체와 염색체 사이의 교배를 통해 소규모 부분집합 탐색을 반복적으로 수행해 나감으로써 MCP를 효율적으로 풀 수 있는 방안을 제시하였다. 그러나 이들 연구에서는 이웃해 탐색 기법들 중 주로 타부 탐색(Tabu Search)만을 대상으로 비교 실험을 수행하였을 뿐만 아니라, 타부 탐색 자체에 대해서도 개선을 위한 심도 있는 연구가 부족하였다. 또한 일부 연구에서는 하나의 데이터만을 대상으로 실험한 결과를 분석함으로써 정확한 결론을 도출해내기 힘든 측면이 있었다.

본 논문에서는 타부 탐색을 포함한 여러 가지 이웃해 탐색 기법들을 대상으로 각 기법들의 성능을 향상시키기 위해 다양한 실험을 수행하였으며, 여러 가지 데이터를 사용함으로써 실험 결과의 신뢰성을 높였다.

이웃해 탐색 기법의 가장 간단한 형태로는 언덕오르기 탐색(Hill-Climbing Search)과 단순 언덕오르기 탐색(Simple Hill-Climbing Search)이 있으며, 언덕오르기 탐색의 한계를

극복하기 위한 탐색 기법으로 타부 탐색과 시뮬레이티드 어닐링(Simulated Annealing)이 가장 널리 사용되고 있다. 다른 탐색 기법들과 마찬가지로 이웃해 탐색 기법들 역시 적용하고자 하는 대상 문제의 특성에 따라 적합한 방법을 개발하여 적용하여야 하며, 다양한 파라미터 값을 문제에 맞게 조정하는 과정이 필요하다. 본 연구에서는 모든 이웃해 탐색 기법들에 있어서 이웃해를 생성하는 방법으로 k개 이하의 열을 삭제하고 삭제한 만큼의 열을 또 다시 추가하는 방법(k-exchange)을 사용하였으며[6] 이 방법을 기반으로 각 기법 별로 성능 개선을 위해 고려해야 할 다양한 실험을 수행하였다.

실험 결과 시뮬레이티드 어닐링의 특성이 타부 탐색의 특성보다 MCP를 해결하기 위해 보다 효과적인 것으로 나타났으며, 한 가지 주목할 만한 사실은 단순 언덕오르기 탐색이 시뮬레이티드 어닐링과 비슷한 성능을 발휘하였다는 것이다. 이는 MCP 자체의 특성과 이웃해 생성을 위한 k-exchange 방법의 특성에 의한 것으로서 k 값을 적절히 설정할 경우 단기적으로는 현재해보다 좋은 이웃해가 나오지 않는다 하더라도 현재해와 평가값이 같은 이웃해로의 이동을 통해 결국 더 좋은 해로의 이동이 가능하기 때문인 것으로 판단된다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 이웃해 탐색 기법들의 기본적인 알고리즘을 소개한다. Ⅲ장에서는 MCP의 해결을 위한 적용 방법을 설명하고 실험을 통해 확인해야 할 여러 가지 고려 사항들을 나열한다. 아울러 MCP와 k-exchange의 특성을 분석함으로써 각 탐색 방법들의 성능을 예측한다. Ⅳ장에서는 다양한 실험 결과를 제시하고 분석하며 마지막으로 Ⅴ장에서 결론 및 향후 과제를 설명한다.

Ⅱ. 이웃해 탐색 기법의 개요

이웃해 탐색 기법은 하나의 완전한 해로부터 출발하여 이웃해를 반복적으로 탐색함으로써 해를 개선시켜 나가는 방식으로서 언덕오르기 탐색, 시뮬레이티드 어닐링, 타부 탐색과 같은 이웃해 탐색 기법들이 최적화 문제 해결을 위해 널리 사용되어 왔다[7, 8, 9, 10].

2.1 언덕오르기 탐색과 단순 언덕오르기 탐색

언덕오르기 탐색은 가장 단순한 형태의 이웃해 탐색 기법으로서 이웃해들 중 다음 해를 선택하는 전략에 따라 언덕오르기 탐색과 단순 언덕오르기 탐색으로 나뉘어진다[11]. 언덕오르기 탐색을 최소화 문제에 적용할 경우 알고리즘은 (그림 2)와 같다. MCP에서 *Cost*는 cover되지 못한 행의 개수를 의미하며, 이와 같은 행을 공백행이라 부른다. 따라서 공백행의 개수를 최소화하는 문제로 정의된다. 언덕오르기 탐색은 모든 이웃해들 중 가장 좋은 해를 다음 해로 선택하는 steepest ascent 전략을 사용하는 반면에 단순 언덕오르기 탐색은 현재해로부터 무작위적으로 하나의 이웃해를 생성한 후 현재해보다 좋으면 이동하는 first improvement 전략을 사용한다. 두 알고리즘 모두 현재해보다 더 좋은 이웃해가 존재하지 않으면 종료하게 된다. 일반적으로 알려진 언덕오르기 탐색의 가장 큰 문제점은 국소 최적화로서 국소 최적해에 도달할 경우 전역적으로는 더 좋은 해가 존재함에도 불구하고 이웃해들 중에서는 더 좋은 해가 존재하지 않기 때문에 알고리즘을 종료하게 된다.

```
Algorithm Hill_Climbing_Search
  Ω : Set of feasible solutions.
  x : current solution.
  Cost : Objective function.
  ℵ(x) : Neighborhood of x ⊂ Ω.
Begin
  Start with an initial solution x ∈ Ω
  Loop
    Find the best solution x* ∈ ℵ(x)
    If Cost(x*) ≥ Cost(x) Then return x
    Else x = x*
  End Loop
End Begin
```

그림 2. 언덕오르기 탐색 알고리즘
Fig 2. Hill-climbing search algorithm

본 연구에서는 전통적인 언덕오르기 탐색과는 달리 이웃해의 *Cost*가 현재해와 같은 경우에도 이동을 허용하도록 하였다. 이와 같은 전략은 언덕오르기 탐색, 특히 단순 언덕오르기 탐색의 성능을 향상시키는 데 큰 역할을 하고 있다.

2.2 시뮬레이티드 어닐링

시뮬레이티드 어닐링은 언덕오르기 탐색의 국소최적화 문제를 개선하기 위한 탐색 기법으로서 단순 언덕오르기 탐

색 알고리즘을 기반으로 하고 있다[8]. 기본 개념은 (그림 3)과 같이 생성한 이웃해가 현재해보다 좋지 않더라도 좋지 않은 정도에 따라 확률적으로 이동할 기회를 부여하는 것이다. (그림 3)에서 Ω, x, *Cost*, $\aleph(x)$의 정의는 (그림 2)에서와 동일하다.

```
Algorithm Simulated_Annealing_Search
  Bestx: Best solution.
  t: Current temperature.
  Schedule(t): Annealing schedule function.
Begin
  Initialize t
  Start with an initial solution x ∈ Ω
  Bestx = x
  While stopping condition is not met Do
    Generate a neighbor solution x* ∈ ℵ(x)
    ΔE=Cost(x*)-Cost(x)
    If ΔE < 0 Then
      x = x*
      If Cost(x*) < Cost(Bestx) Then
        Bestx = x*
    Else
      If (Rand < e^(-ΔE/t)) Then x = x*
    t = Schedule(t)
  End While
  return Bestx
End Begin
```

그림 3. 시뮬레이티드 어닐링 알고리즘
Fig 3. Simulated annealing algorithm

이 알고리즘은 금속의 냉각 과정을 모방한 알고리즘으로서 일반적으로 이동 확률 $p(\Delta E)$는 $e^{-\Delta E/t}$와 같이 결정되는데 ΔE는 해의 개선 정도를 의미한다. 온도 역할을 하는 t는 탐색 초기에는 높은 값을 가지다 탐색이 진행될수록 점점 감소하게 된다. 이로 인해 탐색 초기에는 좋지 않은 해로의 이동 확률이 높다가 탐색이 진행될수록 점점 낮아지게 되며 t 값이 0이 되면 단순 언덕오르기 탐색과 같아진다. *Schedule*(t)은 t 값을 갱신하기 위한 함수를 의미하며 일반적으로 t 값을 어떻게 낮추느냐에 따라 탐색 성능에 영향을 미치게 된다.

2.3 타부 탐색

타부 탐색은 시뮬레이티드 어닐링과 마찬가지로 언덕오르기 탐색의 국소 최적화 문제를 극복하기 위해 개발된 탐색 기법으로서 언덕오르기 탐색 알고리즘을 기반으로 하고 있다[7]. 기본 개념은 (그림 4)와 같이 이웃해들 모두가 현재해보다 좋지 않다 하더라도 그 중에서 가장 좋은 이웃해로의 이동을 허용하는 것이다. 이 때 특별한 조치를 취하지 않는다면 바로 이전해로 다시 돌아가 버리는 현상이 발생할 수 있다. 따라서 Tabu list라는 메모리를 통해 해의 이동(Move)을 규정하는 특성을 저장한 후 일정 기간 동안 그 방향으로의 이동을 금지하게 된다.

```
Algorithm Tabu_Search
  Bestx : Best solution.
  T : Tabu list.
Begin
  Start with an initial solution x ∈ Ω
  Initialize tabu list
  While stopping condition is not met Do
    Find the best solution x* ∈ ℵ(x)
      (move x to x* is not in T)
    x = x*
    Update T
    If Cost(x*) < Cost(Bestx) Then
      Bestx = x*
  End While
  return Bestx
End Begin
```

그림 4. 타부 탐색 알고리즘
Fig 4. Tabu search algorithm

III. 이웃해 탐색 기법의 적용

3.1 초기해 생성 방법

초기해 생성 방법으로는 greedy adding 휴리스틱[1]을 사용하였다. 초기해 x를 만들기 위해 지정한 개수만큼의 열을 하나씩 선택하되 현재 가장 많은 공백행을 cover할 수 있는 열을 선택하였으며 이와 같은 열이 여러 개 존재할 경우에는 그 중에서 임의로 하나를 선택하였다.

3.2 이웃해 생성 방법

이웃해 생성 방법은 이웃해 탐색 기법에 있어서 가장 중요한 요소이다. MCP에서 이웃해를 만드는 가장 단순한 방

법은 현재해 x에 포함된 열 중 하나를 제거하고 새로운 열 하나를 추가하는 것이다. 이와 같이 열 하나를 교체하는 방법을 1-exchange로 정의한다. 그런데 1-exchange의 경우 지역 최적해의 질이 좋지 않고 쉽게 지역 최적해에 빠질 수 있으므로 이웃해의 범위를 확장할 필요가 있다. 따라서 k개 이하의 열을 교체하는 k-exchange 이웃해 생성 방법을 사용할 수 있으며 이와 관련하여 다음과 같은 세 가지 사항을 고려해야 한다.

첫 번째는 k 값을 어떻게 설정하느냐의 문제이다. 1-exchange와 같이 k 값이 작으면 빠르게 지역 최적해에 도달할 수 있지만 지역 최적해의 질이 나빠지게 된다. 반면에 k 값이 크면 이웃해의 범위가 넓어져 보다 좋은 해가 이웃해로 포함될 수 있으나 그만큼 좋지 않은 해까지도 포함될 수 있어 좋은 해를 효율적으로 찾기가 힘들 수도 있다. 극단적으로 k 값이 MCP에서 지정한 전체 열의 수와 같다고 가정하면 완전한 무작위 탐색(random search)이 되어 버린다. 따라서 k 값을 적절히 설정하는 것이 탐색의 효율을 높이는 데 중요한 역할을 할 것으로 추정할 수 있다. 실험 결과, k 값은 2 이상 5 이하의 비교적 작은 값으로 설정할 때 좋은 결과가 나옴을 확인하였다.

두 번째로는 언덕오르기 탐색과 타부 탐색의 경우 한 번의 Move 시 고려해야 할 이웃해의 개수이다. k 값이 커짐에 따라 이웃해의 규모와 관련된 문제가 발생하기 때문이다. 예를 들어 행의 개수가 100개이고 선택해야 할 열의 개수가 50개이며 열 풀에 존재하는 열의 개수가 100,000개인 문제가 있다고 가정하자. 1-exchange의 경우 이웃해의 개수는 $50 \times 100{,}000 = 5{,}000{,}000$개로서 비교적 큰 수이지만 모든 이웃해를 대상으로 한 탐색이 불가능하지는 않다. 그러나 2-exchange만 하더라도 이웃해의 개수는 (1개를 교환하는 경우) + (2개를 교환하는 경우) = 5,000,000개 + ${}_{50}C_2 \times {}_{100000}C_2 \fallingdotseq 6.13 \times 10^{12}$개가 된다. 단순 언덕오르기 탐색과 시뮬레이티드 어닐링의 경우에는 한 번에 하나의 이웃해만을 대상으로 이동 여부를 판단하기 때문에 이웃해의 규모가 커지더라도 큰 영향을 미치지 않지만, 언덕오르기 탐색과 타부 탐색의 경우에는 일반적으로 모든 후보해들 중 가장 좋은 해를 다음해로 선택하기 때문에 2-exchange만 하더라도 후보해의 규모가 너무 커져 모든 후보해를 대상으로 할 수 없게 된다. 따라서 k-exchange의 경우 1개 교환, 2개 교환, ..., k개 교환을 통해 만드는 이웃해의 개수를 일정 개수로 제한할 필요가 있다. 실험 결과에 의하면 언덕오르기 탐색과 타부 탐색 모두 각각의 k개를 교환하여 만드는 이웃해의 개수를 1로 최소화할 때 가장 좋은 결과가 나오는 것으로 확인되었다. 이는 한 번의 Move 시 여러 개의 이웃해를 고려함으로써 소요시간이 증가하는 것보다 단위 시간 동안 많은 Move를 행하는 것이 보다 효과적임을 나타내는 것이다. 즉, 단순 언덕오르기 탐색 또는 시뮬레이티드 어닐링의 특성이 보다 더 효과적인 것으로 판단된다.

마지막으로 고려해야 할 사항은 하나의 이웃해를 생성하기 위해 k개의 열을 교체하는 방법이다. 하나의 이웃해를 생성하는 것은 현재해로부터 열을 제거하는 과정과 새로운 열을 추가하는 과정을 순차적으로 수행함으로써 이루어진다. 열을 제거하고 추가하는 방법 중 고려해 볼만한 조합은 〈표 1〉과 같다.

표 1. 제거 및 추가 방법
Table 1. Removing & Adding methods

	제거 방법	추가 방법
GG	Greedy 방법	Greedy 방법
PG	확률적 방법	Greedy 방법
RG	무작위 방법	Greedy 방법

제거할 열을 선택하는 방법으로는 Greedy 방법, 확률적 방법, 무작위 방법을 생각해 볼 수 있다. Greedy 방법은 제거 시 공백열을 최소화하는 열을 우선적으로 선택하는 방법이며 확률적 방법은 제거 시 발생하는 공백열의 개수에 반비례하여 선택될 확률을 높이는 것이다. 즉, 공백열이 많이 발생하는 열은 선택될 확률이 낮아지게 된다. 무작위 방법은 발생하는 공백열의 개수와 관계없이 임의로 하나의 열을 선택하는 것이다. 직관적으로는 Greedy 방법 또는 확률적 방법이 공백열을 최소화하는 데 크게 기여할 것으로 생각되나 오히려 탐색의 다양성을 저해하는 측면이 강한 것으로 보인다. 실험 결과에 의하면 Greedy 방법이 가장 좋지 않은 결과를 보였으며 확률적 방법과 무작위 방법은 거의 비슷한 성능을 발휘하였다. 열을 추가하는 방법 역시 Greedy 방법, 확률적 방법, 무작위 방법을 생각해 볼 수 있으나, 기본적으로 대상 풀의 열의 개수가 수만 개 이상으로 대규모이기 때문에 확률적 방법과 무작위 방법을 적용할 경우에는 무작위 탐색과 유사해져 해의 개선이 매우 어려운 것으로 나타났다. 따라서 추가 방법으로는 Greedy 방법만을 고려 대상으로 하였으며, 이 방법은 초기해 생성 시 사용한 greedy adding 휴리스틱과 동일하다. 제거 및 추가 시 한 가지 주의할 사항은 제거된 열들 모두가 또 다시 추가됨으로써 이전해와 동일한 해가 되지 않도록 해야 한다는 것이다. 이것은 특히 1-exchange의 경우 성능에 큰 영향을 미치는 것으로 보인다.

3.3 k-exchange 이웃해 탐색 알고리즘

(그림 5)는 k-exchange 이웃해 생성 방법을 사용하여 MCP를 해결하기 위한 알고리즘을 정리한 것으로서 언덕오르기 탐색을 기준으로 작성하였다. 단순 언덕오르기 탐색, 시뮬레이티드 어닐링, 타부 탐색 역시 해당 알고리즘의 특성에 따라 쉽게 변경이 가능하다.

```
Algorithm k-exchange_Hill_Climbing_Search
Begin
   Start with an initial solution x ∈ Ω
   While stopping condition is not met Do
      Generate neighborhood sets of x :
         S1, S2, ..., Sk ⊂ ℵ(x)
      Find the best solution x* ∈ S1 ∪ S2 ∪ ... ∪ Sk
      If Cost(x*) ≤ Cost(x) Then
         x = x*
   End While
   return x
End Begin
```

그림 5. MCP를 위한 언덕오르기 탐색 알고리즘
Fig 5. Hill-climbing search algorithm for MCP

우선 k개를 교환하여 만들 수 있는 이웃해의 집합을 S_k로 정의하였으며 각각의 S_k마다 생성되는 이웃해의 개수를 일정 개수 이하로 제한하였다. 그리고 전통적인 언덕오르기 탐색과는 달리 현재해의 평가값과 같을 경우에도 알고리즘은 종료되지 않고 While 루프를 반복하여 수행하게 된다. 이는 앞에서도 밝힌 바와 같이 k 값이 2만 되더라도 모든 이웃해들을 나열하는 것이 불가능해지므로 실제로 k-exchange에 의해 생성될 수 있는 이웃해들 중 현재해의 평가값과 같거나 더 좋은 해가 존재하지 않는다는 결론을 내릴 수 없기 때문이다. 따라서 본 연구에서의 모든 실험은 수행시간을 종료 조건으로 사용하였다. 이와 같은 특성으로 인해 언덕오르기 탐색이나 타부 탐색은 오히려 단순 언덕오르기 탐색을 확대한 개념으로 해석될 수도 있다.

3.4 MCP와 k-exchange 이웃해 생성

실험 결과에 의하면 MCP를 위해 k-exchange를 적용할 경우 단순 언덕오르기 탐색과 시뮬레이티드 어닐링이 언덕오르기 탐색과 타부 탐색보다 좋은 결과를 보였으며, 단순 언덕오르기 탐색 기법의 성능이 시뮬레이티드 어닐링과 유사함을 확인할 수 있었는데 그 이유를 분석해 볼 필요가 있다. 일반적으로는 시뮬레이티드 어닐링이나 타부 탐색 기법들이 언덕오르기 탐색의 국소 최적화 현상을 개선하기 위해 사용되고 있으며, 성능 역시 언덕오르기 탐색보다 훨씬 좋은 것으로 알려져 있기 때문이다.

먼저 단순 언덕오르기 탐색의 성능이 시뮬레이티드 어닐링의 성능과 유사한 이유는 k-exchange라는 이웃해 생성 방법과 관련하여 분석될 수 있다. 사실상 1-exchange의 경우에는 일반적인 경우와 마찬가지로 시뮬레이티드 어닐링이 훨씬 더 좋은 결과를 보인다. 예를 들어 (그림 6)과 같이 4개의 열로부터 2개의 열을 선택하는 MCP를 생각해 보자. 여기서 평가값은 cover하는 행의 개수를 의미하는 것으로서 최대화 문제로 정의된다.

일단 초기해 생성 과정은 무시하고 탐색 도중 (1, 2)를 갖는 해에 도달했다고 할 때 단순 언덕오르기 탐색에서 1-exchange를 사용하여 전역 최적해인 (3, 4)에 도달할 방법은 존재하지 않는다. 왜냐하면 (1, 2)로부터 하나씩 교체했을 때 (1, 2)에 비해 같거나 좋은 해가 존재하지 않기 때문이다. 시뮬레이티드 어닐링의 경우에는 좋지 않은 이웃해일 경우에도 이동이 가능하므로 전역 최적해인 (3, 4)에 도달할 가능성이 있다. 그런데 2-exchange를 사용한다면 단순 언덕오르기 탐색의 경우 한번에 2개의 열을 교체할 수 있으므로 좋지 않은 해로의 이동을 허용하지 않더라도 바로 전역 최적해로의 이동이 가능하게 된다. 즉, k 값을 크게 설정하게 되면 이웃해의 범위가 넓어지게 되어 (3, 4)까지도 이웃해의 범위 내에 들어오게 되는 것이다. 그러나 대개는 좋지 않은 해까지도 이웃해의 범위에 포함될 수 있으므로 무작정 k 값을 크게 설정한다고 좋은 것은 아니다. 극단적으로 k 값을 선택 가능한 모든 열의 크기로 설정하면 지역 최적해가 바로 전역 최적해가 되어 단순 언덕오르기 탐색만으로도 전역 최적해의 탐색이 가능해지지만, 결국 무작위 탐색이 되어 초기해를 반복해서 생성하는 것과 같아지게 된다. 따라서 적절한 k 값을 찾는 것이 매우 중요하다.

	1	2	3	4
1			1	
2	1		1	
3		1	1	
4		1		1
5		1		1
6	1			1
7	1			1

(a) MCP의 예

후보해	평가값
(1, 2)	6
(3, 2)	5
(1, 3)	5
(4, 2)	5
(1, 4)	5
(3, 4)	7

(b) 후보해의 평가값

그림 6. k-exchange 탐색의 예
Fig 6. An example of k-exchange search

한편 단순 언덕오르기 탐색과 시뮬레이티드 어닐링이 언덕오르기 탐색과 타부 탐색보다 좋은 성능을 발휘하는 이유는 문제 자체의 탐색 공간의 형태와 단위 시간당 Move 횟수의 연관성으로 설명되어질 수 있다. MCP의 경우 현재해의 평가값과 같은 이웃해가 매우 많이 존재하게 되는데 이와 같은 상황 하에서는 이웃해들 중 어떤 이웃해로 이동해야 더 좋은 해로 이동할 가능성이 높은지를 예측하기가 매우 어렵다. 따라서 임의로 현재해와는 다른 후보해로 이동하게 되는데 이 과정에서 다수의 이웃해들을 대상으로 하는 것은 오히려 시간을 낭비하게 되는 경향이 높은 것으로 추정된다. 오히려 단위 시간당 Move 수를 늘려 가능하면 많은 후보해로 이동해 보는 것이 보다 효과적인 것으로 판단된다. 이와 같은 이유로 타부 탐색에서 다수개의 이웃해로부터 다음해로 이동할 경우에는 5-exchange가 1-exchange에 비해 성능의 개선이 거의 없음을 확인할 수 있었다.

Ⅳ. 실험 결과

실험을 위해 사용된 데이터는 〈표 2〉와 같다. MCP814는 국내 모 도시의 지하철 일일 승무일정계획 데이터로서 MCP로 모델링할 경우 각 행은 승무원이 운행할 최소 단위의 열차 운행 구간을 의미하며, 각 열은 한 승무원이 하루 동안 담당할 열차 운행 구간들의 집합으로 일일 근무계획에 해당된다. MCP814의 경우 m = 814, n = 179,514, p = 83으로서 약 18만개의 열들 중 83개를 선택하여 814개의 행을 최대한 많이 cover하는 문제이다. MCP634를 비롯한 다른 데이터들은 실험을 위해 다양한 규모로 인공적으로 생성한 데이터이다. 모든 실험은 Pentium IV 3GHz, 1G RAM PC 상에서 수행되었다. 각 실험 당 20분의 시간 제한을 두고 수행하였으며 모든 실험 결과 수치는 각각 5회 실험 후 평균값을 나타낸 것이다. 각 수치는 공백행의 개수로서 값이 작을수록 더 좋은 결과임을 의미한다.

표 2. 실험 데이터
Fig 2. Experimental data

특징 / data	행의 수(m)	열의 수(n)	선택 열의 수(p)	각 열이 cover하는 행의 수
MCP814	814	179,514	83	10
MCP634	634	142,265	65	10
MCP520	520	111,578	53	9, 10
MCP453	453	72,094	45	9, 10
MCP384	384	40,544	40	8, 9
MCP312	312	55,807	39	7, 8

첫 번째 실험은 이웃해 생성 방법 중 1-exchange에 대한 각 기법의 효과를 알아보기 위한 실험으로서 실험 결과는 〈표 3〉과 같다. HC-A는 언덕오르기 탐색 시 하나의 열을 교환하는 모든 경우를 고려한 후 다음해로 이동하는 방법을 의미한다. HC와 TABU는 각각 언덕오르기 탐색과 타부 탐색을 의미하는데 수행 시 현재해로부터 10개의 이웃해를 대상으로 하여 다음해로 이동하는 방법을 사용하였다. SHC는 단순 언덕오르기 탐색을 의미하며 SA는 시뮬레이티드 어닐링을 의미한다. 표에서 음영 처리된 수치는 각 데이터 별로 가장 좋은 결과를 나타낸 것이다. 실험 결과에 의하면 HC-A는 가장 좋지 않은 결과를 보이고 있는데 이것은 한번의 Move 당 소요되는 시간이 과다하기 때문으로 HC에 비해 총 Move 수가 20분의 1에도 미치지 못하였다. 그리고 타부 탐색과 시뮬레이티드 어닐링이 언덕오르기 탐색이나 단순 언덕오르기 탐색보다 훨씬 좋은 결과를 보였다. 이것은 일반적인 탐색의 경향에 부합하는 것으로서 1-exchange의 경우에는 언덕오르기 탐색 기법에 의한 지역 최적화 현상이 발생하며, 타부 탐색이나 시뮬레이티드 어닐링 기법을 통해 지역 최적화 현상의 극복이 가능한 것으로 판단된다.

표 3. 1-exchange 실험 결과
Table 3. Experimental results of 1-exchange

기법 / data	HC-A	HC	SHC	TABU	SA
MCP814	25.4	18	17	12.8	12.2
MCP634	22.2	16.8	14	10.4	9.2
MCP520	17.4	12	14	9	8.2
MCP453	22.8	21	20.4	14.6	13.6
MCP384	17	16	13.2	7.2	7.4
MCP312	8.8	8	6.4	3.8	3.4

두 번째 실험은 3.2절의 〈표 1〉에서 제시한 바와 같이 하나의 이웃해를 생성하기 위한 제거 및 추가 전략의 효과를 알아보기 위한 것으로서 실험 결과는 〈표 4〉와 같다. 이웃해 탐색 기법으로는 언덕오르기 탐색과 단순 언덕오르기 탐색을 사용하였으며 이웃해 생성을 위해 5-exchange를 사용하였다. GG는 Greedy 제거, Greedy 추가를 의미하고 RG는 무작위 제거, Greedy 추가를 의미하며 PG는 확률적 제거, Greedy 추가를 의미한다. 실험 결과를 통해 제거 시 Greedy한 제거 방법보다는 확률적 방법이나 무작위 방법이 효과적임을 알 수 있으나 확률적 방법과 무작위 방법은 거의 비슷한 성능을 발휘함을 알 수 있다. 이후의 실험 결과에서는 무작위 제거, Greedy 추가(RG)를 사용한 결과를 나타내었다.

표 4. 제거 및 추가 실험 결과
Table 4. Experimental results of removing & adding

방법 / data	HC			SHC		
	GG	RG	PG	GG	RG	PG
MCP814	33.8	13.8	12.2	30	9	8.8
MCP634	28.2	9.8	10	27.6	6.2	5.4
MCP520	25.2	7.6	10	27.8	5	5.8
MCP453	31	14.2	17	28	12.8	15.2
MCP384	23.4	8	8.4	26.4	6.4	6
MCP312	7.6	3.8	3.6	7	1.6	2

세 번째 실험에서는 언덕오르기 탐색을 대상으로 한번의 Move 시 고려해야 할 이웃해의 규모를 달리하여 성능을 비교하였으며 결과는 (그림 7)과 같다. 이웃해 생성 방법으로는 5-exchange를 사용하였다. (그림 7)에서 이웃해의 개수가 10이라는 것은 1개, 2개, 3개, 4개, 5개를 각각 교환하여 만든 이웃해의 개수 10개로 총 50개의 이웃해로부터 다음 Move를 결정하는 것을 의미한다. 실험 결과에서 보는 바와 같이 이웃해의 개수가 많아질수록 공백행의 개수가 늘어남을 알 수 있는데 이는 단위 시간 당 Move 수가 줄어들기 때문이다. 다시 말하면 이웃해의 개수가 적을수록 성능이 향상되는데, 이웃해의 개수가 적어질수록 단순 언덕오르기 탐색과 유사해진다는 사실은 언덕오르기 탐색이나 타부 탐색보다 단순 언덕오르기 탐색이나 시뮬레이티드 어닐링의 탐색 특성이 보다 효과적일 수 있음을 시사하고 있다.

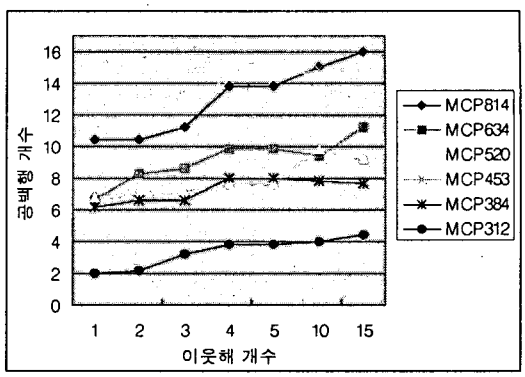

그림 7. 이웃해 규모에 대한 실험 결과
Fig 7. Experimental results of neighborhood size

네 번째 실험에서는 k-exchange 적용 시 k 값의 변화에 따른 성능의 변화를 살펴보았다. (그림 8)은 단순 언덕오르기 탐색에 대해 k-exchange의 k 값을 1, 2, 3, 5, 7, 9로 각각 변경하여 적용한 결과이다. 실험 결과에서 보는 바와 같이 어떤 k 값이 가장 좋은지를 정확하게 판단하기는 어려운 상황이며, 언덕오르기 탐색의 경우도 마찬가지였다. 그러나 k 값이 1인 경우와 9와 같이 너무 클 경우에는 다른 k 값에 비해 결과가 현저히 좋지 않음을 알 수 있다. k 값이 1인 경우에는 지역 최적해로부터 벗어나기가 어려워지는 경향이 있다. 반대로 k 값이 너무 크면 이웃해 생성 시 교환하는 열의 개수가 커지는 경우가 자주 발생하게 되고 이에 따라 현재해보다 나쁜 해가 나올 가능성이 커지게 되며 결국 다음 해로의 이동을 못하게 된다. 따라서 불필요한 이웃해의 생성으로 인해 Move 당 소요시간이 길어짐으로써 단위 시간 당 Move 수가 줄어들게 된다. 결론적으로 적절한 k 값을 설정해야 되는데 실험적으로는 3 내지 5 정도의 값이 좋을 것으로 판단된다.

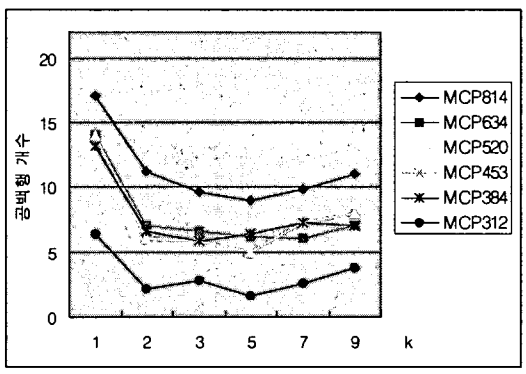

그림 8. k-exchange 실험 결과
Fig 8. Experimental results of k-exchange

다섯 번째로는 시뮬레이티드 어닐링과 타부 탐색 각각에 대해 탐색 성능에 영향을 미칠 수 있는 파라미터 설정에 관한 실험을 수행하였다. 시뮬레이티드 어닐링의 경우 어닐링 스케쥴을 어떻게 하느냐에 따라 성능이 달라질 수 있다. 따라서 $T_{k+1} = T_k - \Delta T$와 같은 선형적 형태와 $T_{k+1} = a T_k$와 같은 지수적 형태를 고려해 봤으며 다양한 ΔT, a값을 적용해 보았다. 또한 온도 T가 최종적으로 0에 근접한 아주 작은 값을 가지는 경우와 탐색이 종료될 때까지 어느 정도 큰 값을 유지하도록 하는 방안도 고려해 보았다. 타부 탐색의 경우에는 Tabu list의 길이를 조정하는 방안과 다각화를 위해 다각화 기간 동안은 현재까지 많이 추가되지 못했던 열을 우선적으로 추가하는 방안을 고려하였다. 그러나 이 모든 경우에 있어서 특별히 차별될만한 실험 결과를 얻지는 못하였다.

마지막으로 〈표 5〉는 언덕오르기 탐색, 단순 언덕오르기 탐색, 시뮬레이티드 어닐링, 타부 탐색 기법의 성능을 종합적으로 분석한 것이다. 이웃해 생성 방법으로는 5-exchange를 사용하였으며 언덕오르기 탐색과 타부 탐색에 대해서는 각 교환 당 10개의 이웃해 즉, 총 50개의 이웃해로부터 다음 해를 선택하도록 하였다. 〈표 3〉의 1-exchange 결과와 비교해 볼 때 대부분의 기법에 있어서 훨씬 좋은 결과를 보임을 알 수 있다. 그러나 타부 탐색만은 개선이 거의 없는 것으로 나타났는데 이것은 k 값이 커짐에 따라 Move 당 소요시간이 대폭 늘어나기 때문인 것으로 해석될 수 있다. 탐색 기법들 사이의 성능은 단순 언덕오르기 탐색과 시뮬레이티드 어닐링이 다른 기법들에 비해 월등히 좋은 것으로 나타났으며 단순 언덕오르기 탐색과 시뮬레이티드 어닐링은 비슷한 것으로 나타났다. 이것은 3.4절에서 설명한 바와 같이 k 값이 커짐에 따라 단순 언덕오르기 탐색만으로도 효과적인 탐색이 가능해지기 때문이다.

표 5. 이웃해 탐색 기법들의 실험 결과
Table 5. Experimental results of neighborhood search algorithms

기법 / data	HC	SHC	TABU	SA
MCP814	13.8	9	13.4	9
MCP634	9.8	6.2	9.6	5.8
MCP520	7.6	5	8.4	5.8
MCP453	14.2	12.8	14	12.4
MCP384	8	6.4	7.6	5
MCP312	3.8	1.6	4	1.8

V. 결론 및 향후 과제

본 논문에서는 MCP를 해결하기 위한 이웃해 탐색 기법들의 다양한 적용 방법에 대해 설명하였다. 기본적으로 k-exchange 이웃해 생성 방법을 기반으로 하고 있으며 이 방법을 이웃해 탐색 기법들에 적용할 경우 고려해야 할 여러 가지 사항들에 대한 실험을 수행해 봄으로써 이웃해 탐색 기법에 대한 결론을 도출할 수 있었다. 실험 결과 단순 언덕오르기 탐색이나 시뮬레이티드 어닐링과 같이 하나의 이웃해만을 대상으로 이동 여부를 판단하는 기법이 언덕오르기 탐색이나 타부 탐색과 같이 여러 개의 이웃해들로부터 다음해를 선택하는 방법보다 더 좋은 결과를 보임을 확인하였다. 또한 k-exchange의 k 값을 적절히 설정함에 따라 모든 기법들의 성능이 향상됨을 확인하였으며, 단순 언덕오르기 탐색이 시뮬레이티드 어닐링과 비슷한 성능을 발휘함을 확인하였다.

향후 과제로는 다음 세 가지 사항을 고려해 볼 수 있다. 첫 번째는 단순 언덕오르기 탐색이나 시뮬레이티드 어닐링의 성능을 개선하는 것이다. 사실상 본 연구를 진행하면서 단순 언덕오르기 탐색과 시뮬레이티드 어닐링에 타부 탐색의 요소를 추가하는 등 여러 가지 방법을 사용해 봤으나 아직 뚜렷한 개선점을 찾지는 못하였다. 두 번째로는 본 연구를 통해 MCP를 위한 이웃해 탐색 기법들에 대한 결론을 도출할 수 있었으므로 향후로는 이웃해 탐색 기법 이외에 기존에 적용되었던 기법들과의 비교 실험이 필요할 것으로 사료된다. 마지막으로는 k-exchange 이웃해 생성 기법을 MCP가 아닌 다른 최적화 문제에도 적용해 봄으로써 본 논문에서 도출한 결과에 대한 검증 및 분석이 필요하다.

참고문헌

[1] R.L. Church, C.S. ReVelle, "The Maximal Covering Location Problem", Regional Science, 32, 101-118, 1974.

[2] E. Andersson, E. Housos, N. Kohl, D. Wedelin, "Crew Pairing Optimization, OR in Airline Industry", Kluwer Academic Press, 1997.

[3] 황준하, 류광렬, "승무일정계획의 최적화를 위한 이웃해 탐색 기법과 정수계획법의 결합", 한국정보과학회 논문지, 31권, 6호, 829-839, 2004.

[4] R.D. Galvao, C. ReVelle, "A Lagrangean Heuristic for the Maximal Covering Location Problem", European Journal of Operational Research, 88, 114-123, 1996.

[5] M.R. Garey, D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman, San Francisco, 1979.

[6] 박태진, 황준하, 류광렬, "대규모 Maximal Covering 문제 해결을 위한 유전 알고리즘", 한국정보과학회 논문지, 31권, 5호, 570-576, 2002.

[7] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, 1997.

[8] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, "Optimization by Simulated Annealing", Science, 220, 671-680, 1983.

[9] 강명주, "Rural Postman Problem 해법을 위한 향상된 Simulated Annealing 알고리즘", 한국컴퓨터정보학회 논문지, 6권, 1호, 25-30, 2001.

[10] 강명주, "시뮬레이티드 어닐링 알고리즘을 이용한 클러스터 기반의 멀티캐스트 라우팅 문제 해법", 한국컴퓨터정보학회 논문지, 9권, 3호, 1-6, 2004.

[11] S. Russel and P. Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, 2nd edition. 2003.

저 자 소 개

황 준 하

1995 부산대학교 컴퓨터공학과 졸업(공학사)

1997 부산대학교 컴퓨터공학과 졸업(공학석사)

2002 부산대학교 컴퓨터공학과 졸업(공학박사)

2002 ~ 현재 : 금오공과대학교 컴퓨터공학부 교수

〈관심분야〉 인공지능, 최적화, 기계학습