韓國컴퓨터情報學會 論文誌
第15卷 第9號, 2010. 9.

2010-15-9-1-6

선형 제약 만족 최적화 문제를 위한 정수계획법 기반 지역 탐색 기법

황 준 하*, 김 성 영*

Integer Programming-based Local Search Technique for Linear Constraint Satisfaction Optimization Problem

Junha Hwang*, Sungyoung Kim*

요 약

선형 제약 만족 최적화 문제는 선형식으로 표현 가능한 목적함수 및 복잡한 제약조건을 포함하는 조합 최적화 문제를 의미한다. 정수계획법은 이와 같은 문제를 해결하는 데 매우 효과적인 기법으로 알려져 있지만 문제의 규모가 커질 경우 준최적해를 도출하기까지 매우 많은 시간과 메모리를 요구한다. 본 논문에서는 지역 탐색과 정수계획법을 결합하여 탐색 성능을 향상할 수 있는 방안을 제시한다. 기본적으로 대상 문제의 해결을 위해 지역 탐색의 가장 단순한 형태인 단순 언덕오르기 탐색을 사용하되 이웃해 생성 시 정수계획법을 적용한다. 또한 부가적으로 초기해 생성을 위해 제약 프로그래밍을 활용한다. *N*-Queens 최대화 문제를 대상으로 한 실험 결과, 본 논문에서 제시한 기법을 통해 다른 탐색 기법들보다 훨씬 더 좋은 해를 도출할 수 있음을 확인할 수 있었다.

Abstract

Linear constraint satisfaction optimization problem is a kind of combinatorial optimization problem involving linearly expressed objective function and complex constraints. Integer programming is known as a very effective technique for such problem but require very much time and memory until finding a suboptimal solution. In this paper, we propose a method to improve the search performance by integrating local search and integer programming. Basically, simple hill-climbing search, which is the simplest form of local search, is used to solve the given problem and integer programming is applied to generate a neighbor solution. In addition, constraint programming is used to generate an initial solution. Through the experimental results using *N*-Queens maximization problems, we confirmed that the proposed method can produce far better solutions than any other search methods.

▸ Keyword : 선형 제약 만족 최적화 문제(Linear Constraint Satisfaction Optimization Problem), 지역 탐색(Local Search), 정수계획법(Integer Programming)

• 제1저자 : 황준하
• 투고일 : 2010. 06. 24, 심사일 : 2010. 07. 24, 게재확정일 : 2010. 08. 18.
* 금오공과대학교 컴퓨터공학부 부교수
※ 본 논문은 금오공과대학교 학술연구비에 의하여 연구된 논문임.

Ⅰ. 서 론

조합 최적화 문제(Combinatorial Optimization Problem)는 제약조건을 동반하는 결정 변수들의 값을 결정하되 주어진 목적함수를 최대화 또는 최소화하는 문제로 정의되며, 지금까지 이를 해결하기 위한 많은 기법들이 제시되어 왔다[1]. 대상 문제에 대한 특유의 지식이 존재하는 경우 이를 활용한 휴리스틱 기법을 적용할 수 있으며, 그렇지 못한 경우라면 지역 탐색(Local Search)과 같은 메타 휴리스틱 기법들을 적용할 수 있다. 그러나 대상 문제에 대한 도메인 지식이 존재한다 하더라도 문제의 규모가 커지는 경우 해당 도메인 지식을 활용한 휴리스틱만으로는 좋은 해를 도출하기 어려우므로 메타 휴리스틱의 틀 내에서 도메인 지식을 활용하는 경우가 대부분이다.

지역 탐색은 완전한 하나의 해로부터 출발하여 이웃해를 기반으로 현재해를 반복적으로 개선하는 방식을 취한다. 대표적인 지역 탐색 기법으로는 언덕오르기 탐색(Hill-Climbing Search), 시뮬레이티드 어닐링(Simulated Annealing), 타부 탐색(Tabu Search) 등이 있다[2, 3]. 지역 탐색을 수행하기 위해서는 이웃해를 정의해야만 하는데 일반적으로 하나의 변수값을 임의의 다른 값으로 변경하는 방식을 사용한다. 이웃해 생성 시 여러 개의 변수값을 동시에 변경할 수도 있지만 이 경우 이웃해의 규모가 기하급수적으로 커지기 때문에 언덕오르기 탐색이나 타부 탐색과 같이 모든 이웃해들 중 가장 좋은 해로 이동하는 기법의 적용이 부적합하다. 물론 단순 언덕오르기 탐색(Simple Hill-Climbing Search)이나 시뮬레이티드 어닐링과 같이 하나의 이웃해만을 대상으로 이동 여부를 판단하는 기법에 있어서는 이와 같은 방법의 적용이 가능하다. 그러나 여러 개의 변수를 한꺼번에 변경할 경우 임의의 값으로 변경하는 단순한 이웃해 생성 방법으로는 현재해보다 더 좋은 해를 발견하기가 어려워진다. 이와 같은 현상은 탐색이 진행될수록 더욱 심해지는데, 변경하는 변수의 개수가 많아질수록 해당 변수들을 변경하여 만들 수 있는 이웃해의 규모 역시 매우 방대해지기 때문이다. 예를 들어, 각각 100개의 값을 가질 수 있는 10개의 변수값을 동시에 변경하고자 할 때 고려할 수 있는 이웃해의 개수는 100^{10}개나 된다. 따라서 변수의 값을 어떤 값으로 변경할 것인가의 문제 자체가 또 다른 탐색 문제가 된다.

본 연구는 지역 탐색의 이웃해 생성 시 많은 결정 변수들을 한꺼번에 변경하되 보다 효과적인 방법을 모색하는 것으로부터 출발하였다. 그러나 모든 조합 최적화 문제에 일반적으로 적용 가능한 효과적인 이웃해 생성 방법을 개발하는 것은 매우 어려운 일이다. 기존 연구 [4]에서는 조합 최적화 문제의 일종인 제약 만족 최적화 문제(Constraint Satisfaction Optimization Problem) 해결을 위한 효과적인 이웃해 생성 방안을 제시한 바 있다. [4]에서는 이웃해 생성을 위해 제약 만족 문제 해결에 적합한 제약 프로그래밍(Constraint Programming)을 적용하되 현재해보다 더 좋은 해의 도출을 요구하는 제약조건을 명시적으로 추가함으로써 25개 이상의 많은 결정 변수들을 한꺼번에 변경하면서도 빠른 시간 내에 현재해보다 더 좋은 이웃해를 도출할 수 있도록 하였다. 본 연구에서는 대상 문제를 선형 제약 만족 최적화 문제(Linear Constraint Satisfaction Optimization Problem)로 제한하여 이를 위한 이웃해 생성 방법을 개발한다. 선형 제약 만족 최적화 문제란 목적 함수와 모든 제약 조건들이 선형적으로 표현될 수 있는 제약 만족 최적화 문제로 정의된다.

본 논문에서 선형 제약 만족 최적화 문제의 해결을 위해 제안한 방법은 단순 언덕오르기 탐색을 기반으로 하고 있으며 이웃해 생성 방법으로 정수계획법(Integer Programming)을 적용하고 있다[5, 6]. 정수계획법은 기본적으로 탐색 공간의 해들을 완전히 열거하는 방식을 사용하기 때문에 최적해의 도출이 보장된다. 따라서 기존 연구 [4]가 단순히 현재해와 같거나 좋은 이웃해를 다음해로 도출하는 것과는 달리 본 논문에서 제안한 방법은 고려 가능한 이웃해들 중 항상 최적해를 다음해로 도출하게 된다. 이는 최적해를 매우 효율적으로 탐색하는 정수계획법의 특성에 의해 가능한 것이다. 단, 대상 문제가 선형적으로 표현될 경우에만 적용이 가능하며 선형적 특징을 바탕으로 수학적 이론을 적용함으로써 매우 효율적으로 최적해를 도출하게 된다. 그러나 문제의 규모가 커질 경우 방대한 탐색 공간을 완전하게 열거해야 하는 정수계획법의 특성상 최적해 또는 준최적해를 도출하기까지 너무 많은 시간과 메모리가 소요된다는 단점이 있다.

본 논문에서는 단순 언덕오르기 탐색의 이웃해 생성 시 전체 결정 변수들 중 일부 변수들만을 대상으로 정수계획법을 적용하여 최적해를 도출하도록 한다. 정수계획법을 적용하면 일반적인 지역 탐색에서의 이웃해 생성에 비해 훨씬 더 많은 변수들을 대상으로 최적해의 도출이 가능하기 때문에 한 번의 이웃해로의 이동 시 훨씬 더 좋은 해로의 이동이 가능하다. 물론 이를 위해 더 많은 시간이 소요될 수 있는데 이는 결국 정수계획법의 성능에 의해 좌우된다 할 수 있다. 정수계획법은 소요 시간과 메모리 요구량에 있어서 매우 부담이 큰 알고리즘이기 때문에 본 논문에서 제시한 바와 같은 형태의 결합에 관한 연구는 드문 편이다. 단지 정수계획법 자체의 성능을 향상시키기 위해 지역 탐색을 부가적으로 사용한 경우가 있었

으며[7], 하나의 문제에 포함된 부문제들을 해결하기 위해 정수계획법과 지역 탐색을 각각 적용한 경우가 있다[8]. 그러나 본 연구의 실험 결과에 의하면 본 논문에서 제안한 기법은 시뮬레이티드 어닐링이나 제약 프로그래밍을 적용했을 때보다 월등히 좋은 결과를 보임을 확인할 수 있었다. 이는 정수계획법의 현재 기술 수준이 본 논문에서 제시한 방법을 뒷받침할 수 있을 정도로 향상되었기 때문인 것으로 해석된다.

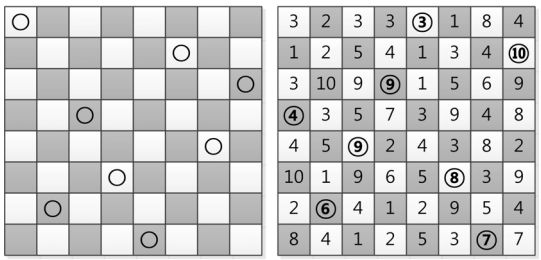
본 논문에서 제시한 방법은 최적화 문제 중 선형 제약 만족 최적화 문제로 초점을 제한하고 있다. 그러나 집합 커버링 문제(Set Covering Problem)와 순회 외판원 문제(Traveling Salesman Problem) 등 실세계의 많은 문제들이 선형 최적화 문제로 정형화될 수 있으며[6] 또한 많은 문제들이 선형 제약 만족 최적화 문제로 정형화될 수 있을 것으로 예상된다. 따라서 본 논문에서 제시한 방법의 추후 활용 가치는 매우 높을 것으로 예상된다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 대상 문제인 N-Queens 최대화 문제에 대해 설명하며, Ⅲ장에서는 본 논문에서 제시한 정수계획법 기반 지역 탐색 기법에 대해 설명한다. 그리고 Ⅳ장에서는 정수계획법 기반 지역 탐색을 비롯하여 다양한 탐색 기법에 대한 실험 결과를 제시하며 마지막으로 Ⅴ장에서 결론 및 향후 과제에 대해 설명한다.

Ⅱ. 대상 문제

본 연구에서는 제안한 기법의 성능을 검증하기 위해 N-Queens 최대화 문제를 활용한다. 원래 N-Queens 문제는 대표적인 제약 만족 문제로서 N개의 Queen을 $N\times N$의 체스 보드에 위치시키되 모든 2개의 Queen 쌍들이 수평, 수직, 대각선의 직선 방향에 위치하지 않도록 하는 문제로 정의된다. 그림 1(a)는 8-Queens 문제에 대한 하나의 해를 나타낸 것이다. 본 논문에서는 N-Queens 문제를 변형하여 제약 만족 최적화 문제로 변경하였으며 이를 N-Queens 최대화 문제라고 부른다. 각 격자에 1부터 10까지의 값을 부여하고 Queen들이 위치한 격자의 값들의 합을 최대화하는 것이 목표이다. 따라서 대상 문제는 직선 방향에 위치하지 않도록 하되 가중치의 합을 최대화하는 제약 만족 최적화 문제로 정의될 수 있다. 그림 1(b)는 가중치가 부여된 8-Queens 최대화 문제에 대한 최적해의 예를 보인 것이다.

제약 만족 문제로서의 N-Queens 문제에 대한 모델링은 다음과 같다. 변수의 개수는 N개이며 각 변수는 0부터 (N-1)까지의 값을 가질 수 있다. 이는 제약조건을 만족하기 위해서는 필수적으로 하나의 열 당 하나의 Queen이 반드시 위치해야 하기 때문이다. 그리고 첫 번째 제약조건은 N개의 변수값은 모두 다르다는 것이며 이를 통해 동일한 행에 위치하는 쌍이 존재하지 않음을 보장할 수 있다. 두 번째 제약조건은 i번째 열의 변수값이 a이고 j번째 열의 변수값이 b일 경우 $(i-j) \neq (a-b)$이고 $(i-j) \neq (b-a)$이어야 한다는 것으로서 이를 통해 대각선 방향으로 위치하는 쌍이 존재하지 않음을 보장할 수 있다.



(a) 원문제 (b) 최대화 문제 (*Obj*=56)

그림 1. 8-Queens 문제의 예
Fig. 1. An Example of 8-Queens Problem

본 논문의 실험 대상 문제인 N-Queens 최대화 문제는 기존 연구 [4]와 [9]에서 실험 대상 문제로 활용된 바 있다. [4]에서는 제약 만족 최적화 문제의 해결을 위한 지역 탐색 적용 시 이웃해 생성 방안으로 제약 프로그래밍을 활용하였다. [9]에서는 제약 만족 최적화 문제를 해결하기 위한 유전 알고리즘의 적용 방안을 제시하였는데, 실험 데이터로서 30-Queens 문제와 같이 규모가 매우 작은 문제를 대상으로 하였다. 두 연구 모두 제약 만족 최적화 문제로서의 N-Queens 최대화 문제를 해결하였다.

그런데 N-Queens 문제를 분석해 보면 식 (1)~(5)와 같이 선형 제약 만족 최적화 문제로 모델링될 수 있음을 알 수 있다. 제약 만족 문제의 모델링에서 N개의 결정 변수로 표현한 것과는 달리 선형 제약 만족 문제에 있어서는 $N\times N$의 결정 변수로 표현된다. 그림 2의 예와 같이 행과 열은 0부터 시작한다고 가정하자. i번째 행, j번째 열에 Queen이 위치하는 경우 1의 값을 갖고 위치하지 않는 경우 0의 값을 갖는 결정 변수를 x_{ij}라 하고, 해당 행과 열에 대한 가중치를 w_{ij}라 하면 목적함수는 식 (1)과 같이 Queen이 놓인 위치의 가중치를 모두 합산한 결과를 최대화하는 것으로 정의할 수 있다. 그리고 그림 2의 ①과 같이 하나의 행에 단 하나의 Queen이 위치해야만 하는 제약조건은 식 (2)에 의해 표현되며 ②와 같이 하나의 열에 단 하나의 Queen이 위치해야만 하는 제약조건은 식 (3)에 의해 표현된다. 또한 ③과 같은 대각선 방향에 대한 제약조건은 식 (4)에 의해 표현된다. 예를 들어 $i+j$의

값이 6인 경우 식 (4)에 의해 대각선 ③에는 단 하나의 Queen만 위치하게 된다. 마찬가지로 ④와 같은 대각선 방향에 대한 제약조건은 식 (5)에 의해 표현된다.

$$Max \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} w_{ij}x_{ij} \quad \cdots\cdots (1)$$

$$\sum_{j=0}^{N-1} x_{ij} = 1 \text{ for } i = 0, \ldots, N-1 \quad \cdots\cdots (2)$$

$$\sum_{i=0}^{N-1} x_{ij} = 1 \text{ for } j = 0, \ldots, N-1 \quad \cdots\cdots (3)$$

$$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{ij} \le 1 \text{ for } i+j = 1, \ldots, ((N-1)+(N-1)-1) \quad \cdots\cdots (4)$$

$$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{ij} \le 1 \text{ for } i-j = (-(N-2)), \ldots, (N-2) \quad \cdots\cdots (5)$$

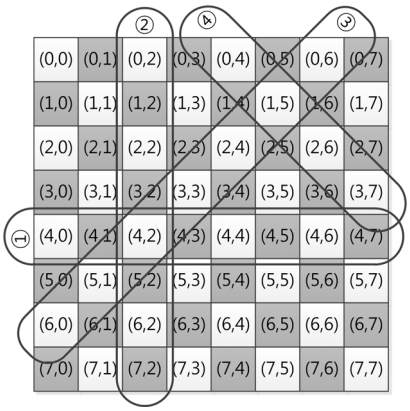


그림 2. 8-Queens 최대화 문제의 선형 모델링
Fig. 2. A Linear Modeling of 8-Queens Maximization Problem

이상과 같이 *N*-Queens 최대화 문제가 선형식으로 표현될 수 있음을 보였다. 대상 문제에 대한 선형식으로의 모델링 자체가 중요한 의미를 갖는 경우가 많다. 대상 문제를 선형식으로 표현할 수 있다면 정수계획법의 적용이 가능하여 문제의 규모가 크지 않은 경우 최적해를 효과적으로 찾을 수 있기 때문이다. [10]에서는 복잡한 일정 계획 문제를 선형식으로 모델링하였고, [11]에서는 2차원 절단 평면 문제에 대한 선형식 모델을 제시한 후 정수계획법을 적용하였다. 그러나 문제의 규모가 커질 경우 정수계획법을 적용한다 하더라도 최적해를 찾는 것이 어려워지며 비교적 좋은 해를 찾는 것조차 매우 많은 시간과 메모리를 필요로 하게 된다. 실제로 64-Queens 최대화 문제에 정수계획법을 적용해 본 결과 약 30분 정도의 시간 내에 최적해를 도출할 수 있음을 확인할 수 있었으나, 100-Queens 최대화 문제만 하더라도 80분이 경과한 후에도 최적해를 도출할 수 없었을 뿐만 아니라 결국 메모리 문제로 프로그램이 중단됨을 확인할 수 있었다. 문제의 규모가 커질수록 이와 같은 현상은 더욱 뚜렷하게 나타나며 해의 개선 속도 또한 매우 느려지게 된다. 따라서 문제의 규모가 큰 경우에는 준최적해를 보다 빠른 시간 내에 도출할 수 있는 기법이 필요하며 본 연구에서는 이를 위한 한 가지 방안으로 지역 탐색과 정수계획법을 결합하는 방안을 제시하고 있다.

III. 정수계획법 기반 지역 탐색

본 논문에서 제시하는 정수계획법 기반 지역 탐색 알고리즘은 그림 3과 같이 단순 언덕오르기 탐색을 기반으로 하고 있다. 그림 4는 그림 3의 알고리즘에서 해의 표현 및 이웃해 생성과 관련하여 8-Queens 최대화 문제로의 적용 예를 나타낸 것이다.

```
// SHC+IP : Algorithm for maximization problems
Algorithm Local_Search_with_Integer_Programming
    x : Variable vector (Current solution).
    Obj : Objective function.
    k : The number of variables to be selected.
Begin
    CP = Start a Constraint Programming
    Add all constraints to CP
    x = Make an initial solution with CP
    While stopping condition is not met Do
        Select k variables randomly from x
        IP = Start an Integer Programming
        Add Obj and all constraints to IP
            (Fix values of unselected variables of x)
        x = Make a neighbor solution with IP
    End While
    return x
End Begin
```

그림 3. 정수계획법 기반 지역 탐색 알고리즘
Fig. 3. Integer Programming-based Local Search Algorithm

N-Queens 최대화 문제의 현재해 *x*는 *N*개의 결정 변수로 구성되며 각 변수가 취할 수 있는 값은 0부터 *N*-1까지의 값이 된다. 8-Queens 최대화 문제에서의 해의 표현은 그림 4(a)와 같다. *i*번째 변수는 *i*번째 열에 위치할 Queen을 의미하고 *i*번째 변수의 값은 *i*번째 열에 위치하는 Queen이 위치하는 행을 의미한다. 이와 같은 모델링은 제약 만족 문제로서의 *N*-Queens 문제를 풀 때와 동일한 것이다.

단순 언덕오르기 탐색을 수행하기 위해 가장 먼저 초기해를 생성하는데 이를 위해 제약 프로그래밍을 사용한다. 사실

본 연구에서는 정수계획법의 역할이 가장 중요하기 때문에 이를 강조하기 위해 정수계획법 기반 지역 탐색이라고 부르고 있지만, 제약 프로그래밍 역시 초기해 생성을 위해 효과적인 역할을 담당하고 있다. 제약 프로그래밍은 대상 문제와 같이 복잡한 제약조건을 포함하는 경우 실행 가능해를 도출하는 데 탁월한 성능을 발휘하는 것으로 알려져 있다[12, 13].

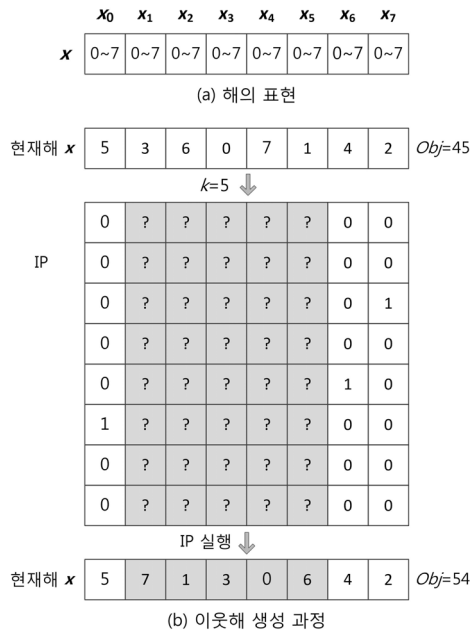


그림 4. 8-Queens 문제로의 적용 예
Fig. 4. An Application Example to an 8-Queens Problem

초기해가 생성되었다면 일반적인 단순 언덕오르기 탐색과 마찬가지로 종료 조건이 만족될 때까지 이웃해 하나를 생성하고 이를 현재해로 재설정하는 과정을 반복 수행하게 된다. 다만 이웃해 생성 시 정수계획법을 적용한다는 점이 기존 알고리즘과의 가장 큰 차이점이다. 단순 언덕오르기 탐색에 있어서 일반적인 이웃해 생성 방법은 임의의 k개의 변수값을 임의의 다른 값으로 변경하는 것이다. 그런데 이와 같은 무작위적인 방법으로는 대상 문제인 제약 만족 문제에 있어서 실행 가능 이웃해를 생성하는 것이 매우 어려울 것이라는 것을 쉽게 예상할 수 있다. 단적인 예로 N-Queens 최대화 문제의 경우 k의 값을 1로 설정한다면, 즉, 모든 제약조건을 만족하는 현재해로부터 1개의 변수만을 변경하여 이웃해를 만든다면 실행 가능 이웃해는 존재하지 않게 된다. 따라서 부득이하게 제약조건을 위배한 경우를 점수화하여 목적함수에 포함시킨 수정된 목적함수를 최적화하는 방식을 취하게 된다. 그러나 이와 같은 방식으로도 실행 가능해의 도출에 상당한 어려움을 겪을 수 있다.

본 연구에서 이웃해 생성 방법으로 정수계획법을 적용하는 방법은 그림 4(b)와 같다. 우선 기존 지역 탐색에서와 마찬가지로 N개의 변수 중 k개의 변수를 무작위로 선택한다. 그리고 나서 전체 변수들을 대상으로 정수계획법을 적용하되 현재해로부터 선택되지 못한 $(N-k)$개의 변수들의 값은 현재값으로 고정하고 선택된 k개 변수들의 값을 무효화하여 정수계획법에 의해 재설정되어야 할 변수들로 분류한다. 정수계획법을 적용하기 위해서는 대상 문제를 선형적으로 표현할 수 있어야 한다. 따라서 그림 4(b)와 같이 0과 1의 값을 갖는 $N \times N$의 변수들을 사용하여 수식 (1)~(5)를 그대로 적용할 수 있도록 하였다. 정수계획법을 통해 이웃해가 도출된 후에는 다시 N개의 변수로 구성된 새로운 현재해로 해석되어진다.

그림 3의 알고리즘에서 존재하는 유일한 파라미터는 k 값이다. 일반적인 단순 언덕오르기 탐색에 있어서 k가 너무 작으면 조기 수렴을 통해 좋지 않은 지역 최적해에 머물게 되는 경향이 있다. 반면에 k가 크면 더 좋은 지역 최적해를 찾을 수 있지만 지역 최적해를 도출하기까지 더 많은 시간을 요구하게 된다. 그러나 k를 너무 크게 할 경우 변수값을 무작위로 변경하는 방법으로는 현재해보다 더 좋은 해를 찾기가 매우 어렵기 때문에 작게 설정하는 것이 보통이다. 그런데 정수계획법을 적용하는 경우에는 k를 어느 정도 크게 설정하더라도 비교적 쉽게 현재해보다 더 좋은 해를 도출할 수 있다. 더군다나 최종적으로 도출된 이웃해는 선택된 k개의 변수값을 변경하여 만들 수 있는 모든 이웃해들 중 최적해가 된다. k가 커짐에 따라 생성 가능한 이웃해의 규모도 기하급수적으로 늘어나게 된다. 예를 들어 1000-Queens 문제에서 k를 5로 설정했을 때 특정 변수 5개에 대해 이론적으로 고려해 볼 수 있는 이웃해의 개수는 1000^5개나 된다. 그럼에도 불구하고 k가 100 이상인 경우에도 적용이 가능한 이유는 대상 문제 자체가 선형식으로 표현이 가능하기 때문이다. 그만큼 정수계획법은 선형 문제에 있어서 매우 효과적이다. k가 작으면 단위 시간당 더 많은 이웃해로의 이동이 가능하지만 한 번의 이웃해 생성으로는 개선 효과가 미미하게 된다. 반면에 k가 크면 한 번의 이웃해 생성으로 더 좋은 이웃해를 도출할 가능성이 높지만 한 번의 이웃해 생성 시 더 많은 시간을 요구하므로 단위 시간당 해의 이동 횟수가 줄어들게 된다. 따라서 적절한 k 값은 실험적으로 결정할 수밖에 없으며, 이는 대상 문제의 특성 및 데이터의 규모 등에 따라 달라질 것으로 예상된다.

본 논문에서 제시하는 정수계획법 기반 지역 탐색은 k 값 외에 더 이상 고려할 파라미터가 존재하지 않을 만큼 매우 간단한 구조를 취하고 있다. 시뮬레이티드 어닐링과 같은 기존

의 복잡한 지역 탐색과 달리 현재해보다 좋지 않은 이웃해가 생성될 때의 규칙도 고려할 필요가 없다. 정수계획법은 이웃해 생성 시 항상 최적해를 도출하며, k 값은 수십 개 이상이 될 정도로 일반적인 지역 탐색에 비해 매우 크다. 이와 같은 특징으로 인해 현재해보다 더 좋은 이웃해를 효과적으로 생성할 수 있을 뿐만 아니라 지역 최적해에 빠지지 않고 보다 넓은 영역으로의 탐색이 가능하다. 이로 인해 본 논문에서 제시한 탐색 기법은 자연스럽게 단순 언덕오르기 탐색의 구조를 취하고 있다.

IV. 실험 결과

실험을 위한 N-Queens 문제의 데이터는 표 1과 같다. 100-Queens로부터 3000-Queens에 이르기까지 다양한 규모의 문제를 사용하였다. 100-Q, 500-Q, 1000-Q, 2000-Q, 3000-Q는 [4]에서 사용한 데이터를 그대로 사용하였으며 나머지는 새로 추가한 것이다. 각 위치에 대한 가중치 값으로는 1부터 10까지의 값을 무작위로 부여하였다. 따라서 Queen의 개수가 N개인 N-Q 데이터의 이론적 최적해의 목적함수 값은 $(10 \times N)$이 된다. 이론적 최적해는 모든 Queen이 가중치 10에 위치하는 경우이다. 물론 제약조건을 모두 만족해야 되기 때문에 그러한 해가 존재한다고 보장할 수는 없다. 그러나 N의 값이 커질수록 가중치 값으로 10을 갖는 위치가 많아지므로 이론적 최적해가 존재할 가능성은 높아지게 된다. 실제로 실험 결과, 본 논문에서 제안한 기법을 통해 200-Q, 300-Q, 500-Q, 1000-Q에 대한 이론적 최적해를 도출할 수 있었다.

표 1. 실험 데이터
Table 1. Experimental Data

data	Queen의 개수	이론적 최적값	data	Queen의 개수	이론적 최적값
100-Q	100	1000	1000-Q	1000	10000
200-Q	200	2000	1500-Q	1500	15000
300-Q	300	3000	2000-Q	2000	20000
500-Q	500	5000	3000-Q	3000	30000

모든 실험은 [4]와 마찬가지로 Pentium D 3.4GHz, 2G RAM PC 상에서 수행되었으며, 각 실험 당 수행 시간은 1시간으로 제한하였다. 1시간이면 본 논문에서 제시한 기법과 비교 대상 기법들의 탐색 특성을 파악하기에 충분한 시간으로 판단하였다.

먼저 본 논문에서 제시한 기법인 정수계획법 기반 지역 탐색(SHC+IP)에 있어서 유일한 파라미터인 이웃해 생성 시 교체되는 변수의 개수(k)에 대한 영향을 살펴보았다. 실험 결과는 표 2와 같으며 모든 수치는 최종 목적함수 값으로서 각 데이터 및 k 별로 5회 실험 후 평균을 취한 값이다. '×' 표시는 해당 k 값의 경우 전체적인 경향을 파악하는 데 무의미하다고 판단하여 실험을 생략하였음을 의미하며, 'Fail' 표시는 메모리 문제로 인해 해의 도출이 불가능함을 의미한다.

표 2. SHC+IP의 k에 대한 실험 결과
Table 2. Experimental Results of SHC+IP w.r.t. k

k \ data	100-Q	200-Q	300-Q	500-Q	1000-Q	1500-Q	2000-Q 3000-Q
30	976	1970	2922	4788	8552	10441	
50	**994**	1996	2985	4947	9512	12843	
70	877	**2000**	2999	4991	9811	14005	
90	954	1692	**3000**	4999	9945	14491	
110	×	1639	2249	**5000**	9985	14750	Fail
130	×	1670	2239	4236	**10000**	14906	
150	×	1780	2329	3643	**10000**	14973	
170	×	×	×	×	9905	**14991**	
190	×	×	×	×	Fail	Fail	

표 2에 의하면 데이터 규모가 상대적으로 작은 경우에는 k를 비교적 작게 설정했을 때 더 좋은 해를 도출할 있으며 데이터의 규모가 커질수록 더 좋은 해를 도출할 수 있는 k도 점점 증가함을 알 수 있다. 기본적으로 k가 클수록 한 번의 이웃해 생성 시 더 좋은 해가 생성될 수 있지만 너무 크면 한 번의 이웃해 생성을 위해 너무 많은 시간이 소요된다. 따라서 각 데이터 별로 가장 적절한 k 값을 찾는 것이 정수계획법 기반 지역 탐색의 성능을 판가름하는 가장 중요한 요소라 할 수 있다. 1500-Q에 있어서는 k가 170보다 더 큰 값을 가질 경우 더 좋은 해를 도출할 가능성도 있다. 그러나 메모리 문제로 인해 해의 도출 자체가 불가능하였다. 정수계획법은 데이터의 규모가 큰 경우 최적해를 도출하기까지 많은 시간과 메모리를 필요로 하는 알고리즘이다. 특히 결정 변수와 제약조건의 수가 증가함에 따라 메모리 요구량도 급격하게 증가한다. k가 커지면 그만큼 고려해야 할 제약조건의 수도 늘어나게 된다. 이로 인해 1000-Q와 1500-Q에서 k가 190일 때 해의 도출이 불가능함을 확인하였다. 또한 문제의 규모가 큰 2000-Q와 3000-Q에서는 k 값에 관계없이 해의 도출이 불가능함을 확인하였다.

다음 실험에서는 정수계획법 기반 지역 탐색의 효과를 검

증하기 위해 대상 문제에 적용 가능한 다양한 탐색 기법들을 적용한 후 그 결과를 비교하였다. 대상 문제인 *N*-Queens 최대화 문제는 정수계획법 기반 지역 탐색 외에도 다양한 탐색 기법을 통해 해결될 수 있다.

첫 번째로는 대상 문제가 최적화 문제의 일종이므로 광범위한 최적화 문제에 적용 가능한 지역 탐색을 들 수 있다. 정수계획법 기반 지역 탐색의 기본 구조인 단순 언덕오르기 탐색(SHC) 역시 지역 탐색의 일종이다. 그러나 단순 언덕오르기 탐색은 지역 최적해에 도달한 후 더 이상 다른 영역으로의 탐색이 불가능하다. 이와 같은 단점을 보완하기 위해 개발된 대표적인 탐색 기법이 시뮬레이티드 어닐링(SA)이다. 시뮬레이티드 어닐링에서는 현재해보다 좋지 않은 이웃해로의 이동을 확률적으로 허용함으로써 지역 최적해로부터의 탈출이 가능하도록 하였다. 단순 언덕오르기 탐색이나 시뮬레이티드 어닐링에서는 제약조건을 그대로 처리하기 어렵기 때문에 식 (6)과 같이 제약조건을 목적함수에 포함시켜 사용한다. *Obj*는 원래 목적함수를 의미하며 *Penalty*는 제약조건을 위배한 경우를 점수화한 것으로서 대상 문제에서는 직선 상에 위치한 Queen 쌍의 개수로 표현될 수 있다. a는 원래 목적함수와 벌점의 비율로서 이 값에 따라 탐색 양상이 달라질 수 있다. [4]에 의하면 a가 5이고 k가 1일 때 성능이 가장 좋은 것으로 나타났다. 따라서 본 연구에서는 a를 5로, k를 1로 설정하여 실험을 수행하였다.

$$TotalObj = Obj - a \times Penalty \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (6)$$

두 번째는 제약 프로그래밍(CP)의 적용이다. 대상 문제는 제약 만족 최적화 문제의 일종이다. 제약 프로그래밍은 제약 만족 문제뿐만 아니라 제약 만족 최적화 문제를 해결하기 위해서도 적용될 수 있다. 기본적으로는 깊이 우선 탐색과 제약 전파를 통해 탐색 공간을 축소하되 지금까지 탐색한 가장 좋은 해의 정보를 바탕으로 탐색이 불필요한 영역을 추가로 축소시켜 나간다. 그러나 제약 프로그래밍은 기본적으로 탐색 공간을 모두 열거하는 알고리즘으로서 최적해의 도출을 보장하지만 최적해 또는 준최적해를 도출하기까지 너무 많은 시간이 소요되기 때문에 대규모 최적화 문제에는 부적합하다.

[4]에서는 제약 프로그래밍을 기반으로 한 지역 탐색(SHC+CP)을 제안하였으며 성능이 매우 뛰어남을 확인하였다. 기본 구조는 정수계획법 기반 지역 탐색과 동일하며 이웃해 생성을 위해 정수계획법 대신 제약 프로그래밍을 적용하였다는 점에서 차이가 있다. [4]의 실험에서는 모든 데이터에 있어서 k가 25 또는 30인 경우에 가장 좋은 결과를 보인 것으로 나타났다. SHC+CP에 대한 본 연구의 실험 결과로는 각 데이터 별로 가장 좋은 결과를 제시하였다. 본 연구에서는 제약 프로그래밍 개발 도구로서 ILOG Solver 6.7을 사용하였다[14].

마지막으로는 정수계획법(IP)만을 적용하는 것이다. 대상 문제가 선형 최적화 문제이므로 정수계획법의 적용이 가능하다. 본 연구에서 정수계획법 개발 도구로는 ILOG CPLEX 12.1을 사용하였다[15]. ILOG Solver와 CPLEX는 각각 제약 프로그래밍 라이브러리와 정수계획법 라이브러리로서 현재 전 세계적으로 학술적, 상업적 목적을 위해 가장 많이 사용되고 있다.

표 3은 정수계획법 기반 지역 탐색을 비롯하여 앞서 설명한 총 6가지 탐색 기법을 적용한 결과이다. 첫 번째 실험과 마찬가지로 각 수치는 5회 실험 후 평균을 나타낸 것이다. 정수계획법 기반 지역 탐색(SHC+IP)의 결과로는 표 2에서 각 데이터 별로 가장 좋은 결과를 표기하였다.

표 3. 탐색 기법들의 비교 실험 결과
Table 3. Comparison Results among Search Methods

data \ methods	SHC	SA	CP	SHC+CP	IP	SHC+IP
100-Q	930	987	727	991	993	**994**
200-Q	1920	1988	1249	1987	1988	**2000**
300-Q	2858	2978	1828	2975	2909	**3000**
500-Q	4813	4898	2961	4933	Fail	**5000**
1000-Q	Fail	Fail	5469	9616	Fail	**10000**
1500-Q	Fail	Fail	8427	13265	Fail	**14991**
2000-Q	Fail	Fail	11066	**16151**	Fail	Fail
3000-Q	Fail	Fail	16692	**20434**	Fail	Fail

SHC는 전반적으로 SA에 비해 좋지 않음을 알 수 있는데 SHC의 특성상 현재해보다 더 좋은 해로의 이동이 매우 어려우며 특히 지역 최적해에 도달한 경우 이를 빠져나오기가 매우 힘든 것으로 보인다. 이로 인해 SHC는 데이터 규모가 작은 100-Q의 경우에도 실행 가능해를 찾지 못하는 경우가 있었다. 표 3에는 이와 같은 경우를 제외한 평균을 표시하였다. SA의 경우 문제 규모가 상대적으로 작은 100-Q와 200-Q에서는 SHC+IP 등과 큰 차이가 없으나 300-Q부터 점점 큰 차이로 뒤처지다가 SHC와 마찬가지로 1000-Q부터는 허용 시간 내에 실행 가능해조차 도출할 수 없었다. 이는 SHC와 SA의 경우 원래 목적함수와 제약조건에 의한 목적함수를 모두 만족시키는 해를 도출하는 데 어려움이 있기 때문인 것으로 해석된다. CP는 모든 데이터에 있어서 가장 좋지 않은 결과를 보였다. CP의 특성상 제약 만족 문제에는 적합하지만

대규모 최적화 문제에는 부적합한 것으로 보인다. IP는 100-Q와 같이 데이터 규모가 작은 경우에는 상당히 효과적이지만 규모가 조금만 커지더라도 해의 개선 속도가 매우 느려지게 된다. 더군다나 정수계획법은 변수의 개수와 제약조건의 개수가 늘어남에 따라 과도한 메모리를 요구하기 때문에 500-Q 이후의 데이터에 대해서는 메모리 문제로 인해 실행 가능해의 도출이 불가능함을 알 수 있다. 결국 SHC+IP와 SHC+CP의 성능이 가장 뛰어남을 알 수 있는데 그 중에서도 본 논문에서 제시한 SHC+IP가 데이터 규모가 커질수록 훨씬 더 좋은 결과를 보임을 알 수 있다. 더군다나 SHC+IP는 200-Q에서 1000-Q에 이르는 데이터에 대해 항상 최적해를 도출할 수 있음을 확인할 수 있으며 이는 SHC+IP의 우수성을 반영하는 것이다.

다만 SHC+IP는 2000-Q 이후로 해의 도출이 불가능한데 이는 IP와 마찬가지로 메모리 문제에 의한 것이다. 제약프로그래밍 역시 메모리 소요량이 매우 높은 편이다. 따라서 SHC+CP 또한 메모리 소요량이 매우 높다. 그럼에도 불구하고 실험 결과에 의하면 SHC+CP는 6000-Queens 최대화 문제까지 적용이 가능하였다. 그런데 SHC+IP와 SHC+CP에 있어서 적용 가능한 데이터 규모의 차이는 알고리즘 그 자체보다는 모델링의 차이에서 오는 것이다. 예를 들어 SHC+CP를 SHC+IP와 같이 $N\times N$의 변수로 모델링하여 적용한 결과 SHC+IP와 마찬가지로 2000-Q 이후로는 해의 도출이 불가능함을 확인할 수 있었다. 결론적으로 본 실험을 통해 일차적으로 SHC+IP의 우수한 성능을 확인할 수 있다. 또한 부가적으로 SHC+IP를 적용하기 위해서는 대상 문제에 대한 선형식으로의 표현 자체가 얼마나 중요한지, 그것도 가능한 한 보다 적은 개수의 변수 및 제약조건으로의 표현이 얼마나 중요한지를 알 수 있다.

참고로 그림 4는 500-Q에 있어서 각 기법들의 수행 시간에 따른 목적함수 값의 전형적인 변화를 나타낸 것이다. 단, 정수계획법의 경우 해의 도출이 불가능하므로 제외하였다. CP는 해의 개선이 가장 느리며 SHC와 SA는 최초의 실행 가능해를 도출하기까지 많은 시간이 소요됨을 알 수 있다. 반면에 SHC+IP와 SHC+CP는 해의 개선 속도가 빠르며 전 시간대에 걸쳐 가장 좋은 결과를 보이고 있다. 초반에는 SHC+CP의 개선 속도가 빠르나 15분 정도 경과된 시점에서 SHC+IP가 추월하기 시작하여 이후로는 SHC+IP가 지속적으로 더 좋은 결과를 보이고 있다. 이와 같은 현상은 데이터의 크기가 클수록 더욱 뚜렷하게 나타나는데, 1500-Q의 경우 SHC+IP가 최초의 이웃해를 생성한 2분 정도 경과된 시점부터 SHC+CP를 추월하기 시작하여 그 차이가 큰 폭으로 증가함을 확인할 수 있었다. 따라서 실시간으로 보다 좋은 해를 요구하는 응용 문제에 있어서도 SHC+IP가 효과적일 것으로 판단된다.

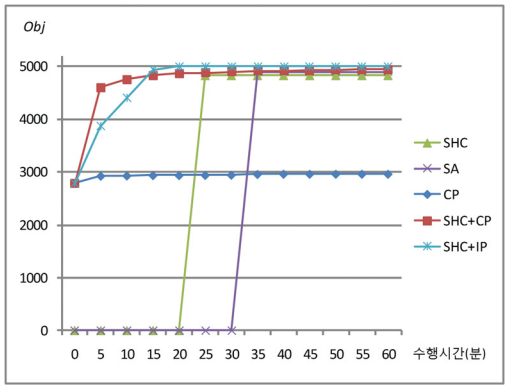


그림 5. 수행시간에 따른 탐색 기법들의 성능 추이
Fig. 5. Performance Curve of Search Methods w.r.t. Time

V. 결론 및 향후 과제

본 논문에서는 선형 제약 만족 최적화 문제를 해결하기 위해 지역 탐색의 틀 내에서 정수 계획법을 활용하는 방안을 제시하였다. 지역 탐색 기법들 중 단순 언덕오르기 탐색을 기반으로 하되 이웃해 생성 방법으로 정수계획법을 적용하였다. 정수계획법은 선형 최적화 문제에 있어서 최적해를 매우 효과적으로 도출할 수 있다. 이로 인해 이웃해 생성 시 기존의 지역 탐색에 비해 훨씬 더 많은 변수를 효과적으로 변경하는 것이 가능하며, 결국 지역 최적해에 빠지지 않고 보다 넓은 영역으로의 탐색이 가능하였다. 선형 제약 만족 최적화 문제의 일종인 N-Queens 최대화 문제에 대한 실험 결과, 본 논문에서 제시한 기법을 통해 시뮬레이티드 어닐링, 제약 프로그래밍 등 다른 탐색 기법보다 빠른 시간 내에 훨씬 더 좋은 해를 도출할 수 있음을 확인하였다.

본 논문에서 제안한 정수계획법 기반 지역 탐색은 대상 문제가 선형적으로 표현 가능한 경우, 즉, 정수계획법을 적용할 수 있는 경우에 한하여 효과가 클 것으로 기대되며, 일정계획 문제를 비롯한 다양한 실세계 문제들이 선형 최적화 문제 또는 선형 제약 만족 최적화 문제로 모델링될 수 있을 것으로 예상된다. 따라서 향후로 선형 제약 만족 최적화 문제 외에 일반적인 선형 최적화 문제에 대한 정수계획법 기반 지역 탐색의 효과를 살펴봄과 동시에, 정수계획법 기반 지역 탐색이 효과적으로 적용될 수 있는 실세계 문제를 발굴하고 적용함으로써 제안한 방법의 유용성을 입증할 필요가 있다.

참고문헌

〔1〕 C.R. Reeves, "*Modern Heuristic Techniques for Combinatorial Problems*," McGraw-Hill Book Company, pp.1-19, 1995.

〔2〕 C. Blum, and A. Roli, "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," ACM Computing Surveys, Vol. 35, No. 3, pp.268 - 308, September 2003.

〔3〕 S. Russell, and P. Norvig, "*Artificial Intelligence : A Modern Approach*," Prentice Hall, pp.110-119, 2005.

〔4〕 황준하, "제약 만족 최적화 문제의 해결을 위한 지역 탐색과 제약 프로그래밍의 결합," 한국컴퓨터정보학회논문지, 제 15권, 제 5호, 39-47쪽, 2010년 5월.

〔5〕 J.T. Linderoth, and M.W.P. Savelsbergh, "A Computational Study of Search Strategies for Mixed Integer Programming," INFORMS Journal on Computing, Vol. 11, No. 2, pp.173 - 187, Spring 1999.

〔6〕 L.A. Wolsey, "*Integer Programming*," John Wiley & Sons, pp.1-137, 1998.

〔7〕 E. Danna, E. Rothberg, and C.L. Pape, "Integrating Mixed Integer Programming and Local Search : A Case Study on Job-Shop Scheduling Problems," Proceedings CPAIOR'03, *http://www.crt.umontreal.ca/cpaior/*, May 2003.

〔8〕 E.K. Burke, J. Li, and R. Qu, "A Hybrid Model of Integer Programming and Variable Neighbourhood Search for Highly-constrained Nurse Rostering Problems," European Journal of Operational Research, Vol. 203, No. 2, pp.484-493, June 2010.

〔9〕 J. Paredis, "Genetic State-Space Search for Constrained Optimization Problems," Proceedings of the 13th International Joint Conference on Artificial Intelligence, pp.967-972, August 1993.

〔10〕 K.J. Chen, and P. Ji, "A Mixed Integer Programming Model for Advanced Planning and Scheduling(APS)," European Journal of Operational Research, Vol. 181, No. 1, pp.515 - 522, August 2007.

〔11〕 E. Silva, F. Alvelos, and J.M. Valério de Carvalho, "An Integer Programming Model for Two- and Three-stage Two-dimensional Cutting Stock Problems," European Journal of Operational Research, Vol. 205, No. 3, pp.699 - 708, September 2010.

〔12〕 E. Tsang, "*Foundations of Constraint Satisfaction*," Academic Press Limited, pp.1-319, 1996.

〔13〕 손석원, 한광록, "공동체 라디오 방송을 위한 주파수 할당의 최적화," 한국컴퓨터정보학회논문지, 제 13권, 제 2호, 51-57쪽, 2008년 3월.

〔14〕 IBM ILOG Solver, "*IBM ILOG Solver User's Manual and Reference Manual*," Version 6.7, 2009.

〔15〕 IBM ILOG CPLEX, "*IBM ILOG CPLEX User's Manual and Reference Manual*," Version 12.1, 2009.

저 자 소 개

황 준 하

2002 : 부산대학교 컴퓨터공학과 공학박사

2002 - 현재 : 금오공과대학교 컴퓨터공학부 교수

관심분야 : 인공지능, 최적화, 기계학습, 프로그래밍언어

김 성 영

2003 : 부산대학교 컴퓨터공학과 공학박사

2004 - 현재 : 금오공과대학교 컴퓨터공학부 교수

관심분야 : 영상처리, 패턴인식, 컴퓨터비전