승무일정계획의 최적화를 위한 이웃해 탐색 기법과 정수계획법의 결합

(A Hybrid of Neighborhood Search and Integer Programming for Crew Schedule Optimization)

황 준 하 † 류 광 렬 ††
(Junha Hwang) (Kwang Ryel Ryu)

요 약 정수계획법에 기반한 기법들은 다양한 승무일정계획 최적화 문제를 해결하는 데 매우 효과적인 것으로 알려져 있다. 그러나 정수계획법은 대상 문제의 제약조건 및 목적함수가 모두 선형적으로 표현되어야만 적용이 가능하다는 단점이 있으며 문제의 규모가 클 경우 과도한 수행 시간과 메모리 자원을 요구하게 된다. 반면 이웃해 탐색 기법과 같은 휴리스틱 탐색 기법은 대상 문제의 제약조건이나 목적함수의 형태에 관계없이 쉽게 적용이 가능하다. 그러나 이웃해 탐색 기법은 복잡한 탐색 공간을 탐색할 경우 국소 최적해에 도달한 후 국소 최적해로부터 쉽게 빠져나오지 못하는 경우가 많다. 본 논문에서는 이웃해 탐색 기법과 정수계획법의 장점을 효과적으로 결합하기 위한 방안을 제시하고 있으며 실제 운행중인 지하철 승무일정계획 문제에 적용해 봄으로써 대규모 승무일정계획 최적화 문제에 성공적으로 적용될 수 있음을 확인하였다.

키워드 : 승무일정계획, 이웃해 탐색, 정수계획법

Abstract Methods based on integer programming have been shown to be very effective in solving various crew pairing optimization problems. However, their applicability is limited to problems with linear constraints and objective functions. Also, those methods often require an unacceptable amount of time and/or memory resources given problems of larger scale. Heuristic methods such as neighborhood search, on the other hand, can handle large-scaled problems without too much difficulty and can be applied to problems having any form of objective functions and constraints. However, neighborhood search often gets stuck at local optima when faced with complex search spaces. This paper presents a hybrid algorithm of neighborhood search and integer programming, which nicely combines the advantages of both methods. The hybrid algorithm has been successfully tested on a large-scaled crew pairing optimization problem for a real subway line.

Key words : Crew Scheduling, Neighborhood Search, Integer Programming

1. 서 론

일반적으로 지하철 등과 같은 운송수단의 운행계획은 특정 기간동안 동원되는 차량들 각각에 의해 운행되는 구간들을 표시한 운행시간표로서 표현된다. 각 구간은 레그(leg)라고 불리는 몇 개의 소구간으로 나뉠 수 있는데, 하나의 레그는 한 승무원이 휴식을 취하지 않고 연속으로 운행해야만 하는 최소의 구간을 의미한다. 이들 레그는 그 출발역과 도착역이 어디인가에 따라 여러 가지 유형으로 나뉜다. 승무일정계획 수립 시 작성하는 승무원 1인(혹은 1조)의 근무 계획인 근무표는 한 승무원이 출근하여 퇴근할 때까지 운행해야 할 여러 개의 레그들로 구성된다. 기본적으로 하나의 근무표가 실행 가능한 근무표가 되기 위해서는 하루 일과를 시작하는 역과 마감하는 역이 동일해야 한다. 즉, 하루 일과를 수행하는 승무원이 출근과 퇴근을 동일한 역에서 할 수 있도록 각 근무표의 계획을 수립해야만 한다. 이와 같이 승무원의 출근과 퇴근이 가능한 역을 승무소라 하며 경우에 따라 2개 이상의 승무소가 존재할 수 있다. 이외에도 근로조건에 따른 여러 가지 제약조건을 만족해야만

· 본 연구는 부산대학교 연구(보조)비(4년과제)에 의한 연구임
† 종신회원 : 금오공과대학교 컴퓨터공학부 교수
jhhwang@kumoh.ac.kr
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
부산대학교 컴퓨터 및 정보통신연구소 연구원
krryu@pusan.ac.kr
논문접수 : 2003년 8월 4일
심사완료 : 2004년 2월 27일

실행 가능한 근무표가 될 수 있다. 대부분의 실세계 승무일정계획 문제의 경우 다양한 특성을 포함하는 방대한 개수의 근무표들이 생성될 수 있다. 예를 들어, 같은 레그로부터 운행을 시작하는 근무표라 하더라도 다음 대기역에서의 쉬는 시간에 따라 이후에 운행해야 할 레그들이 달라지게 되며 이에 따라 하루 근무시간도 달라지게 된다. 이와 같은 각 근무표의 다양한 특성에 의해 해당 근무표의 비용이 결정된다.

일반적으로 승무일정계획 문제는 근무표 생성 단계와 최적조합 선정 단계의 2단계로 분리하여 접근할 수 있다[1]. 근무표 생성 단계에서는 열거 방법 등을 사용하여 개별 근무표 제약조건을 만족하는 근무표들을 생성하고, 최적조합 선정 단계에서는 근무표 생성 단계에서 생성된 근무표들을 대상으로 최소의 비용으로 모든 레그들을 운행할 수 있는 부분집합을 선정한다. 그런데 운행해야 할 레그의 수가 많을 경우 제약조건이 어떻게 주어지느냐에 따라 생성 가능한 개별 근무표의 수가 방대해질 수 있고 이 때 최적조합을 선정하는 것은 복잡도가 매우 높은 문제가 된다.

최적조합 선정 문제는 다음 수식과 같이 집합 커버링(set covering) 문제로 표현될 수 있다. 집합 커버링 문제는 n개의 열(column)들로부터 m개의 행(row)을 모두 커버할 수 있는 부분집합을 선택하되 비용의 합을 최소화하는 문제로 정의된다. 승무일정계획 문제에 있어서 각 근무표는 열에 대응되며 레그는 행에 대응된다.

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i=1,\ldots,m$$

$$\sum_{j=1}^{n} b_{ij} x_j \leq v_i, \quad i=1,\ldots,p$$

$$x_j \in \{0,1\}, \quad j=1,\ldots,n$$

x_j는 0 또는 1의 값을 가지는 결정변수로서 j번째 열이 선택될 경우 1의 값을 취하게 되며 c_j는 j번째 근무표의 비용을 의미한다. 상수 a_{ij}는 m개의 행과 n개의 열로 이루어진 행렬의 한 원소로서 j번째 열이 i번째 행을 커버한다면(즉, 근무표 j가 레그 i를 운행한다면) 1로 지정되며 이외의 경우에는 0으로 지정된다. 첫 번째 제약조건은 각 레그들이 선택된 근무표들 중 한 개 이상의 근무표에 의해 운행되어야 함을 의미하며 두 번째 제약조건의 상수 b_{ij}와 v_i는 선택된 근무표들의 조합이 준수해야 할 p개의 선형 제약조건들 각각을 규정한다.

승무일정계획 문제를 해결하기 위해 동원된 기존의 방법들로는 라그랑지안 완화 기법(Lagrangian relaxation)[2,3], 유전 알고리즘(genetic algorithm)[4], 열 생성 기법(column generation)[5,6] 등이 있다. 처음 두 가지 기법은 고정된 개수의 근무표들을 대상으로 집합 커버링 문제를 효과적으로 풀기 위한 방법으로서 생성된 근무표의 개수가 비교적 적은 경우에 적용이 가능한 방법이다. 반면에 열 생성 기법은 최적조합 선정 결과를 바탕으로 필요한 근무표를 동적으로 생성하는 방법으로서 근무표의 개수가 방대할 경우에도 적용이 가능하다. 그러나 열 생성 기법은 정수계획법(integer programming)을 기반으로 하고 있기 때문에 실행 가능한 해를 도출하기까지 과도한 시간이 소요될 수 있으며 해당 문제의 목적함수와 제약조건들이 모두 선형식으로 표현되어야만 적용이 가능하다는 단점이 있다. 보다 최근의 연구에서는 정수계획법과 휴리스틱 탐색 기법을 결합하여 문제를 해결한 바 있다[7]. 이 연구에서는 최적조합 선정 단계에서 정수계획법을 적용한 후 도출된 결과의 불완전한 부분을 보완하기 위해 휴리스틱 탐색 기법을 개발하여 사용하고 있다. 그러나 순수한 정수계획법만으로는 최적조합 선정 결과가 만족스럽지 못하기 때문에 휴리스틱 탐색 단계에서 실행 가능한 해를 도출하기까지 너무 많은 시간이 소요된다는 단점이 있다.

본 논문에서는 복잡한 승무일정계획 문제를 효과적으로 해결하기 위한 방안으로 최적조합 선정 단계에서 이웃해 탐색 기법과 정수계획법을 결합하는 방안을 제시하고 있다. 일반적으로 이웃해 탐색 기법은 대규모 최적화 문제 해결에 있어서 완전 열거 방식을 기반으로 하는 정수계획법보다 비교적 좋은 해를 훨씬 짧은 시간 내에 찾을 수 있는 기법으로 알려져 있다. 뿐만 아니라 비선형적 목적함수나 제약조건을 포함하는 문제에도 쉽게 적용이 가능하다는 장점이 있다. 반면에 정수계획법은 문제의 규모가 작을 경우 전역적 탐색 과정을 통해 최적해에 근접한 해를 빠르게 찾을 수 있다는 장점이 있다. 본 논문에서는 두 가지 기법의 장점을 효과적으로 결합할 수 있는 방안을 제시하고 있다.

본 논문에서 제시하고 있는 알고리즘은 3장에서 기술할 최대 커버링(maximal covering) 문제를 해결하기 위해 적용된다. 먼저 근무표 생성 단계를 통해 생성된 근무표 풀(pool)을 대상으로 간단한 그리디(greedy) 휴리스틱을 사용하여 문제에서 주어진 개수만큼의 근무표를 선택함으로써 초기해를 생성한다. 그러나 이렇게 간단한 방법으로 생성된 근무표 집합으로는 운행되지 못하는 레그들이 발생하게 된다. 따라서 이웃해 탐색 기법을 사용하여 현재 근무표 집합을 바탕으로 더 많은 레그를 운행할 수 있도록 반복적인 개선 탐색 과정을 수행해 나간다. 만약 이웃해 탐색 도중 일정한 기간이 경과해도 해의 개선이 이루어지지 않는다면 탐색 다각화를 유도하기 위해 정수계획법 모듈이 호출되는데, 정수계획법을 호출하기 전에 현재까지의 가장 좋은 해(근무

표 집합)를 대상으로 많은 레그를 커버하는 데 도움이 되지 못한다고 판단되는 일부 근무표들을 제거한다. 정수계획법 모듈은 나머지 근무표 집합을 대상으로 수행되며 근무표 풀로부터 제거된 근무표들을 대체할 가장 좋은 근무표들을 찾는 것이 목표이다. 정수계획법만을 사용하여 최대 커버링 문제를 풀 경우에는 전체 풀로부터 문제에서 주어진 근무표 개수만큼을 모두 선택해야 하지만 이웃해 탐색 도중 호출될 경우에는 근무표들 중 일부만을 교체하도록 함으로써 원문제보다 훨씬 적은 규모의 최대 커버링 문제를 풀면 되도록 하였다. 정수계획법 모듈을 수행한 후에는 그 결과를 초기해로 또 다시 이웃해 탐색 기법을 수행하는 방식으로 두 가지 모듈을 반복적으로 수행하게 된다. 정수계획법을 수행하는 동안에는 대상 문제에 존재하는 비선형적인 목적함수 및 제약조건을 고려할 수 없으나 반복적인 이웃해 탐색 과정을 통해 또 다시 고려될 수 있으므로 큰 문제가 되지 않는다. 실험 결과 두 가지 기법을 결합한 알고리즘을 사용한 경우 각 기법을 단독으로 사용할 때보다 더 좋은 수준의 계획을 보다 빠른 시간 내에 수립할 수 있음을 확인하였다.

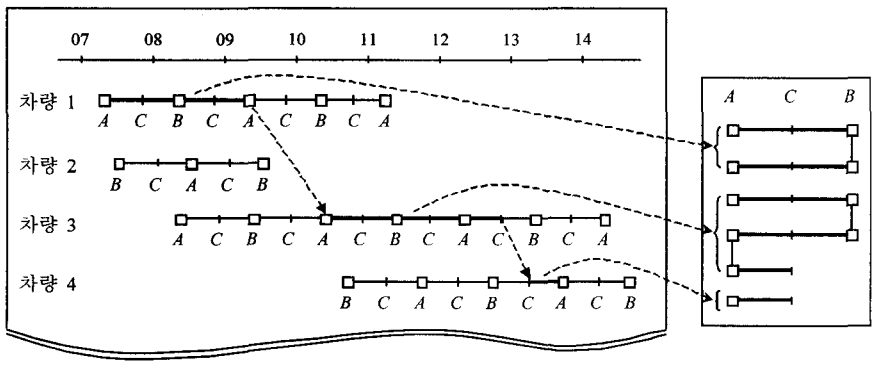
이하의 구성은 다음과 같다. 2장에서는 본 연구에서 대상으로 하고 있는 승무일정계획 문제에 대해 상세히 설명한다. 그리고 3장에서는 기존의 방법을 적용할 경우의 문제점과 해결 방안을 설명하고 4장에서는 이웃해 탐색 기법과 정수계획법을 결합하는 방안에 대해 기술한다. 5장에서는 실험 결과를 분석하며 마지막으로 6장에서 결론을 맺는다.

2. 승무일정계획 문제

본 논문은 실제 운행중인 지하철 노선에 대한 일정계획 중 승무일정계획 문제를 대상으로 하고 있다. 지하철 일정계획은 열차운행계획, 승무일정계획, 승무원배정계획으로 이루어진다. 먼저 1일 수송계획, 배차간격 및 가용차량 등을 고려하여 열차운행계획을 수립하고 승무일정계획 단계에서는 각 차량을 운행할 근무표 집합을 생성하며 마지막으로 각 근무표에 따라 운행할 승무원을 배정한다. 각 단계의 결과는 다음 단계의 입력으로 사용되어 다음 단계의 결과에 직접적인 영향을 미치게 되며 역으로 각 단계의 계획 수립 결과에 따라 이전 단계의 수정이 요구되기도 한다. 이와 같이 단계들 사이의 이상적인 연계가 요구되나 각 단계별 문제의 난이도가 매우 높아 각 단계별 계획을 순차적으로 수립하고 있다. 특히 본 논문의 대상 문제인 승무일정계획 문제는 타 단계에 비해 난이도가 월등히 높아 전체적인 성능 향상의 관건이 되고 있다.

그림 1(a)는 열차운행계획 수립 결과로 생성된 열차운행계획표의 예를 보인 것으로 각 차량들은 승무소 A와 B 사이를 반복적으로 왕복 운행하게 된다. 승무원은 승무소에서 출퇴근을 하며 열차 운행 중 휴식을 취하기도 한다. 승무소 사이에는 실제 많은 역이 존재하지만 그림에서는 승무원이 다른 열차를 갈아타기 위해 휴식과 교대가 가능한 역인 중간역 C만 표시되어 있다. 본 대상 문제에서 레그의 유형은 출발역과 방향에 따라 $A \rightarrow C$, $C \rightarrow B$, $B \rightarrow C$, $C \rightarrow A$의 4가지로 나뉘어지며, 전체 운행계획표는 총 640개의 레그로 이루어져 있다.

그림 1(b)는 차량운행계획표의 레그들 중 일련의 레그들로 구성된 하나의 근무표의 예를 보인 것으로서 총 10개의 레그로 이루어진다. 이 근무표에 따라 근무하는 승무원은 승무소 A로 출근하여 하루 일과를 시작하며 승무소 A와 B 사이를 1회 왕복한 후 다른 차량으로 갈아타기 위해 잠시 대기하게 된다. 그리고 다른 차량을 타고 5개의 레그를 연속으로 운행한 후 중간 대기역 C에서 잠시 대기하게 되며 최종적으로 승무소 A로 돌아와 퇴근함으로써 하루 일과를 마무리한다.



(a) 차량운행계획표 (b) 근무표

그림 1 차량운행계획표 및 근무표의 예

실행이 가능한 근무표가 되기 위해서는 다음 예와 같은 여러 가지 제약조건을 준수해야만 한다. 근무표의 시작과 완료는 승무소(A 또는 B)에서만 가능하고 시작 승무소와 완료 승무소는 동일해야 한다. 이 때 하루의 과업을 시작하고 완료하는 승무소를 해당 근무표의 소속 승무소라 한다. 그리고 하나의 근무표는 총 10개의 레그로 구성되어야 하는데 이 제약조건과 앞서 설명한 제약조건으로 인해 중간 대기역 C에서의 교대가 불가피하게 된다. 승무원이 쉬지 않고 연속으로 운전할 수 있는 시간은 피로와 같은 안전상의 문제로 인해 최대 3시간을 초과할 수 없지만, 하루 과업을 수행하는 동안 너무 잦은 대기를 하게 되면 오히려 번거로워질 수 있으므로 대기횟수는 2회에서 4회까지 허용하되 대기횟수가 적은 근무표를 선호한다. 승무일정계획을 통해 이상의 제약조건 및 그 외의 다양한 제약조건을 만족하는 근무표들 중 일부를 선정하여 차량운행계획표 상의 모든 레그들의 운행이 가능하도록 계획을 수립해야만 한다.

개별 근무표 뿐 아니라 선정된 근무표 집합에 대해서도 준수해야 할 제약조건과 목적함수가 존재한다. 우선 주요 제약조건으로는 선택해야 할 근무표 개수가 총 65개로 고정되어 있는데 이 중 38개가 A 소속 근무표이고 나머지는 B 소속 근무표여야 한다는 것이다. 총 레그의 개수가 640개이고 각 근무표마다 10개의 레그를 운행하게 되므로 일부 레그들은 한 개 이상의 근무표에 의해 운행될 수 있다. 만약 같은 레그를 두 개 이상의 근무표가 운행할 경우 그 중 하나의 근무표에 해당하는 승무원만이 실제 기관사로서의 역할을 담당하게 되고 나머지 근무표의 승무원은 승객으로서 단지 이동만 하게 된다. 이와 같이 두 개 이상의 근무표에 의해 운행되는 레그를 편승 레그라 한다. 승무일정계획 시 고려해야 할 목적함수 중 하나는 평균근무시간을 최소화하는 것으로서, 각 근무표의 근무시간은 첫 번째 레그의 출발시각으로부터 마지막 레그의 도착시각까지를 의미한다. 또 다른 한가지 목적함수로는 대기횟수가 많은 근무표의 개수를 최소화하는 것이다. 마지막 목적함수는 근무표 사이의 출퇴근 순서 제약과 관련된 것이다. s_i와 f_i를 각각 근무표 i의 과업 시작시각과 완료시각이라 할 때 두 근무표 i와 j가 $s_i < s_j \rightarrow f_i < f_j$를 만족한다면(즉, 먼저 출근한 승무원이 먼저 퇴근) 두 근무표 i와 j는 출퇴근 순서 제약을 만족한다고 할 수 있다. 본 논문의 대상 문제에서는 출퇴근 순서 제약을 위배하는 근무표 쌍의 개수를 최소화하는 것이 중요한 목표가 되고 있다.

3. 최대 커버링(Maximal Covering) 모델

앞서 언급한 바와 같이 최적조합 선정을 위해서는 먼저 근무표 생성 단계에서 생성 가능한 근무표를 모두 생성할 필요가 있다. 모든 제약조건을 만족하는 개별 근무표를 생성하는 일은 비교적 단순한 작업으로 단순 열거 방법 등의 간단한 방법을 동원할 수 있다. 문제는 640개의 레그들로부터 10개를 선정하여 만들 수 있는 근무표의 개수가 너무 방대하여 생성 가능한 모든 근무표를 생성하는 것 자체가 불가능하다는 것이다. 따라서 지금까지 승무일정계획 수립을 위해 동원되었던 많은 방법들 중 열 생성 기법을 적용하는 것이 가장 적절할 것으로 판단된다. 열 생성 기법은 최적화 도중에 필요로 하는 열들을 동적으로 생성함으로써 모든 열들을 생성하지 않고도 잠재적으로 모든 열들을 대상으로 탐색을 수행할 수 있는 알고리즘으로 알려져 있다. 그러나 정수계획법에 기반하고 있는 열 생성 기법을 규모가 큰 문제에 적용할 경우 어느 정도 좋은 해를 도출하기까지 너무 많은 시간이 소요된다는 단점이 있다. 더군다나 열 생성 기법은 선형식으로 표현하기 어려운 목적함수나 제약조건을 포함하는 문제에 대한 적용 자체가 불가능하다. 따라서 출퇴근 순서 목적함수와 같이 대상 문제가 선형식으로 표현하기 어려운 요소를 포함할 경우 적용이 불가능하다.

사실, 출퇴근 순서 제약 위배를 최소화하기 위한 목적함수를 선형식으로 표현하는 것이 이론적으로 불가능한 것은 아니다. 근무표 i의 시작시각과 완료시각을 각각 s_i, f_i라 하고 n을 전체 근무표의 개수라 정의할 때, 모든 근무표 쌍들 중 출퇴근 순서를 위배하는 쌍들의 집합 V를 $\{(i, j) | s_i < s_j \text{ and } f_j < f_i,\ 1 \le i, j \le n\}$과 같이 정의할 수 있다. 그리고 출퇴근 순서 목적함수를 식으로 표현하면 다음과 같다.

$$\text{Minimize} \quad \sum_{(i,j)\in V} z_{ij}$$

$$\text{subject to} \quad x_i + x_j - 1 \le z_{ij}, \quad (i,j) \in V$$

$$x_i \in \{0,1\}, \quad i = 1, \ldots, n$$

$$z_{ij} \in \{0,1\}, \quad (i,j) \in V$$

근무표 i가 선택될 경우 x_i의 값은 1이 되고 선택되지 않으면 0의 값을 가진다. 제약조건의 부등식에 의해 출퇴근 순서 제약을 위배하는 근무표 i와 j가 동시에 선택될 경우(x_i=1, x_j=1) z_{ij}의 값은 1이 되어야 하며, 이외의 경우에는 모든 z_{ij} 값의 합이 최소화되어야 하므로 0의 값을 가지게 된다. 즉, z_{ij}의 합을 최소화하기 위해 출퇴근 순서 제약을 위배하는 쌍들이 동시에 선택되는 것을 최소화하게 된다. 이와 같이 출퇴근 순서 목적함수를 선형식으로 표현할 수 있음에도 불구하고 대상 근무표의 개수가 늘어남에 따라 결정변수 집합 V의 원소 개수가 너무 많아지게 되고 제약조건의 수가 너무 많아지게 되므로 비교적 적은 규모인 1만개 정도만 된다 하더라도 정수계획법의 적용이 불가능해지게 된다. 그렇다면 근무

표 생성 단계에서 개별 근무표의 근무시간을 특정한 범위 내로 제한함으로써 최적조합 선정 단계에서 출퇴근 순서 목적함수를 고려하지 않더라도 간접적으로 이를 달성할 수 있는 방법을 생각해 볼 수 있다. 극단적인 예를 든다면 근무시간이 8시간인 근무표들만을 대상으로 최적조합을 선정한다면 선정된 근무표들 사이에는 항상 출퇴근 순서 제약을 만족하는 관계가 성립될 것이다. 그러나 이와 같은 엄격한 제약조건이 추가된 상태에서는 근무표 생성 단계에서 생성할 수 있는 근무표의 개수가 대폭 줄어들게 되고 이에 따라 최적조합 선정 단계에서 고려할 수 있는 대상이 줄어들게 된다. 결국 모든 레그들을 운행하기 위해서는 문제에서 요구하는 근무표 개수보다 훨씬 더 많은 근무표들을 요구하게 될 것이다.

이웃해 탐색과 같은 휴리스틱 탐색 기법은 목적함수나 제약조건의 형태에 관계없이 쉽게 적용이 가능하다는 장점이 있다. 따라서 본 논문에서는 대상 문제를 해결하기 위한 주된 알고리즘으로 이웃해 탐색 기법을 사용하고 있으며 동시에 정수계획법의 전역적 탐색 능력을 부가적으로 사용할 수 있도록 두 가지 기법을 결합하여 사용하고 있다. 그런데 열 생성 기법과는 달리 이웃해 탐색 기법은 대규모 승무일정계획 문제에 있어서 생성 가능한 모든 근무표들을 대상으로 효율적인 탐색을 수행하는 것이 불가능하다. 이웃해 탐색 기법을 통해 효과적인 탐색이 가능하도록 하기 위해서는 근무표 생성 단계에서 적절한 규모의 제한된 근무표 풀을 만들 필요가 있다. 결과적으로 최적조합 선정 단계에서 제한된 근무표 풀을 대상으로 함에 따라 해의 질이 저하될 수 밖에 없다. 즉, 문제에서 요구하는 주어진 근무표 개수만으로는 모든 레그들을 운행할 수 있는 조합을 찾기가 어려워지게 되어 집합 커버링 문제를 푼 결과 더 많은 근무표를 포함하게 된다. 따라서 문제에서 주어진 근무표 개수 제약조건을 준수하기 위해서는 초과된 개수만큼의 근무표를 제거해야만 하며 이에 따라 어떤 근무표에 의해서도 운행되지 못하는 레그가 발생하게 되는데 이와 같은 레그를 공백 레그라 부른다.

대상 문제의 경우 기존의 승무일정계획 문제와는 달리 근무표의 개수가 고정되어 있다는 특징이 있다. 따라서 앞에서 설명한 바와 같이 집합 커버링 문제로 모델링된 문제를 푼 후 초과한 만큼의 근무표를 제거하기보다는 최대 커버링 문제로 모델링함으로써 두 단계를 한 번에 처리할 수 있도록 하였다. 최대 커버링 문제는 주어진 개수의 열들을 선택하여 가장 많은 레그를 커버하는 문제로 정의되어진다. m과 n을 각각 레그의 개수와 근무표 풀에 존재하는 근무표 개수라 하고 d를 선택해야 할 근무표 개수라 하자. 그리고 결정변수 x_j와 상수 a_{ij}, b_{ij} 및 v_i를 각각 1장에서 집합 커버링 문제를 표현할 때와 같이 정의하면 최대 커버링 문제는 다음 수식과 같이 표현된다.

$$\text{Minimize} \quad \sum_{i=1}^{m} y_i$$

$$\text{subject to} \quad \sum_{j=1}^{n} x_j = d$$

$$\sum_{j=1}^{n} a_{ij} x_j + y_i \geq 1, \quad i=1,\ldots,m$$

$$\sum_{j=1}^{n} b_{ij} x_j \leq v_i, \quad i=1,\ldots,p$$

$$x_j \in \{0,1\}, \quad j=1,\ldots,n$$

$$y_i \in \{0,1\}, \quad i=1,\ldots,m$$

첫 번째 제약조건은 주어진 d개만큼의 근무표를 선택하도록 하기 위한 제약조건이다. 두 번째 제약조건은 i번째 레그가 공백 레그일 경우 y_i의 값이 1이 되도록 하고 한 개 이상의 근무표에 의해 운행된다면 0이 되도록 함으로써 공백 레그의 개수가 최소화될 경우 목적함수 값인 y_i의 합이 최소화되도록 유도하고 있다. 마지막 제약조건은 집합 커버링 모델에서와 마찬가지로 선정된 근무표 집합이 준수해야 할 제약조건들을 의미한다. 실험 결과에 의하면 최대 커버링 모델을 적용하는 것이 집합 커버링 모델을 적용한 후 초과한 만큼의 근무표를 제거하는 것보다 훨씬 더 적은 공백 레그가 발생함을 확인할 수 있었다.

최대 커버링 모델을 적용함으로써 공백 레그의 개수를 줄일 수는 있다 하더라도 최적조합 선정 시의 대상 근무표 풀 자체가 이웃해 탐색 기법을 효과적으로 적용할 수 있는 수준으로 제한되어 있기 때문에 최적조합 선정 단계만으로는 공백 레그를 완전히 제거하는 것이 매우 어렵다. 본 연구에서는 기존 연구 [7]과 같이 대상 문제에 적합하게 개발된 휴리스틱 교정 탐색을 통해 일부 근무표들을 수정 또는 교체하는 작업을 반복함으로써 남아있는 공백 레그들이 제거될 수 있도록 하였다. 그러나 본 연구에서는 최적조합 선정 단계에서 정수계획법만을 사용했던 기존 연구와는 달리 이웃해 탐색 기법과 정수계획법을 결합한 방법을 사용함으로써 공백 레그를 훨씬 더 많이 줄일 수 있게 하였으며 이에 따라 휴리스틱 단계에서 보다 쉽게 모든 공백 레그를 제거할 수 있도록 하였다.

그림 2는 [7]에서 사용한 수정을 통한 휴리스틱 교정의 기본 개념을 표현한 것이다. 그림과 같이 레그 L_1을 운행하고 있던 근무표 P에 대해 L_1 대신 공백 레그인 L_2를 운행하게 함으로써 근무표 P'으로 수정할 수 있다. 물론 L_2가 이전 레그 또는 이후 레그를 운행하는 데 지장이 없어야 하며 이외의 제약조건을 모두 준수해야만 이와 같은 수정이 가능하다. 이 때 만약 L_1이 편승 레그였다면 근무표 P가 운행하지 않는다 하더라도 또

다른 근무표에 의해 운행될 수 있으므로 추가적인 공백 레그의 발생 없이 공백 레그 하나가 제거될 수 있다. 만약 L_1이 편승 레그가 아니었다면 편승 레그를 이용한 공백 레그의 제거가 가능할 때까지 이와 같은 일련의 수정 작업을 반복해 나간다. 그런데 L_2와 같은 공백 레그에 대해 이 레그를 운행할 수 있도록 수정 가능한 근무표가 여러 개 존재할 수 있으므로 어떤 근무표부터 수정해 볼 것인지 결정하는 탐색 과정이 필요하다. 너비 우선 탐색(breadth-first search)이나 반복적 깊이 우선 탐색 기법(iterative deepening search) 등이 공백 레그로부터 편승 레그까지의 수정 경로를 찾기 위한 탐색 기법으로 동원될 수 있다. 이상과 같은 수정 작업을 비롯하여 근무표를 교체하는 등의 여러 가지 휴리스틱 탐색 기법들이 휴리스틱 교정을 위해 사용되고 있다.

휴리스틱 교정은 근무표 생성 단계에서 생성된 근무표 풀 내에 존재하는 근무표들 이외의 새로운 근무표들을 생성할 수 있다는 점에서 잠재적으로 생성 가능한 모든 근무표들을 대상으로 할 수 있다는 장점이 있다. 뿐만 아니라 출퇴근 순서 목적함수와 같은 비선형적 요소까지도 고려할 수 있기 때문에 최적조합 선정 단계에서 정수계획법만을 사용할 때뿐만 아니라 하이브리드 기법을 사용한 경우에도 출퇴근 순서 목적함수를 어느 정도 개선할 수 있는 효과가 있다. 그러나 대개의 경우 휴리스틱 교정 과정을 통해 근무시간 또는 대기횟수 면에 있어서 수정된 근무표의 질이 저하되는 경향이 있다. 그림 2의 예에서도 기존 근무표 P의 대기횟수가 2회인 반면 수정된 근무표 P'의 대기횟수가 3회로 늘어남으로써 질이 저하됨을 알 수 있다. 따라서 휴리스틱 교정 단계를 수행하기 이전 단계인 최적조합 선정 단계에서부터 공백 레그의 발생을 최소화할 뿐 아니라 선정된 근무표 집합의 전체적인 질을 좋은 상태로 유지할 필요가 있다.

4. 이웃해 탐색 기법과 정수계획법의 결합

이웃해 탐색 기법은 하나의 완전한 해로부터 출발하여 이웃해를 반복적으로 탐색함으로써 해를 개선시켜 나간다. 이웃해 탐색 기법들 중 대표적인 기법으로는 언덕 오르기 탐색(hill-climbing search), 시뮬레이티드 어닐링(simulated annealing)[8], 타부 탐색(tabu search)[9]이 있다. 언덕 오르기 탐색은 가장 단순한 이웃해 탐색 기법으로서 현재해를 수정하여 생성된 이웃해들 중 현재해보다 더 좋거나 같은 해들이 존재한다면 가장 좋은 해를 새로운 현재해로 설정하여 이동하는 과정을 반복하며 그렇지 않다면 알고리즘은 종료된다. 언덕 오르기 탐색은 알고리즘의 특성 상 국소 최적해(local optima)에 머무는 현상이 발생한다. 즉, 해의 탐색 도중 국소 최적해에 도달하게 되면 전체 탐색 공간 내에 더 좋은 해들이 많이 존재한다 하더라도 이웃해들 모두가 현재해보다 좋지 않기 때문에 현재 국소 최적해를 벗어날 방법이 없게 되는 것이다. 시뮬레이티드 어닐링이나 타부 탐색 등과 같은 이웃해 탐색 기법들은 모두 언덕 오르기 탐색의 국소 최적화 현상을 보완하기 위해 만들어졌다. 시뮬레이티드 어닐링은 현재해로부터 이웃해를 하나만 생성한 후 이동 여부를 판단하되 현재해보다 좋다면 무조건 이동하게 되며 현재해보다 좋지 않다 하더라도 확률적으로 이동할 가능성이 있도록 하고 있다. 현재해보다 좋지 않은 해로의 이동을 허용함으로써 국소 최적해를 벗어날 수 있도록 유도하는 것이 시뮬레이티드 어닐링의 기본 개념이라 할 수 있다. 타부 탐색은 이웃해 집합을 대상으로 그 중 가장 좋은 해로 이동하게 하는데 언덕 오르기 탐색과는 달리 이웃해들 모두가 현재해보다 좋지 않다 하더라도 그 중에서 가장 좋은 해로의 이동을 허용함으로써 국소 최적해로부터 벗어날 수 있는 기회를 제공한다. 그러나 이웃해 탐색 기법들은 모두 현재해를 기반으로 탐색을 진행해 나가는 국지적 방법을 사용하기 때문에 국소 최적화 현상을 완전히 해결하지는 못하고 있다. 반면 정수계획법은 탐색 공간의 해

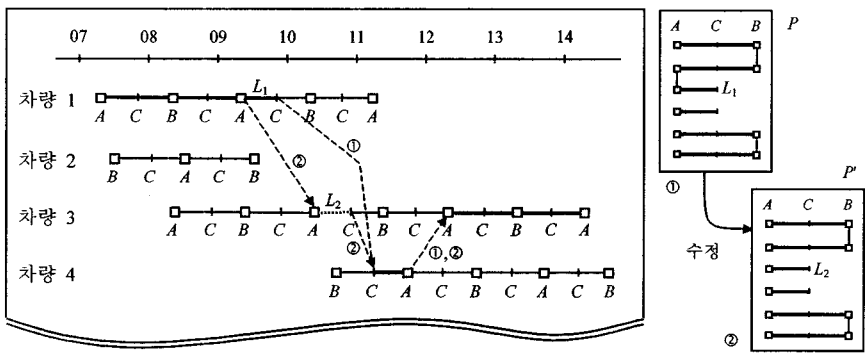


그림 2 수정에 의한 휴리스틱 교정 과정

들을 완전히 열거하는 방식이기 때문에 전역 최적해의 도출을 보장할 수 있다[10]. 단, 문제 자체가 선형적으로 표현될 경우에만 적용이 가능하며 비교적 효율적인 탐색 알고리즘을 사용함에도 불구하고 기본적으로 방대한 탐색 공간을 완전하게 열거해야 하기 때문에 최적해를 구하는 데 너무 많은 시간이 소요된다는 단점이 있다.

근무표 생성 단계에서 적절한 규모의 근무표 풀이 생성되면 이 풀을 대상으로 레그들을 가장 많이 커버할 수 있는 부분집합을 찾기 위해 본 논문에서는 이웃해 탐색 기법과 정수계획법을 결합한 알고리즘을 적용한다. 그림 3은 이웃해 탐색 기법들 중 타부 탐색과 정수계획법을 결합한 알고리즘을 표현한 것이며 시뮬레이티드 어닐링 등 적용되는 구체적인 이웃해 탐색 기법에 따라 약간의 수정만으로 적용이 가능하다. 초기해는 그리디 추가(greedy adding) 휴리스틱[11]과 유사한 방법을 사용하여 생성된다. 문제에서 주어진 개수만큼 근무표들을 하나씩 순차적으로 선택하되 현재 커버되어 있지 않은 레그들을 가장 많이 커버할 수 있는 근무표들 중 하나를 선택하는 것이다. 타부 탐색에서는 초기해를 현재해로 설정한 후 현재해를 기반으로 이웃해들을 생성하고 이웃해들 중 하나를 새로운 현재해로 설정하는 과정을 반복적으로 수행해 나간다.

이웃해 탐색을 위한 알고리즘을 설계할 때 성능에 영향을 미치는 가장 중요한 요소가 바로 이웃해를 정의하는 방법이라 할 수 있다. 일반적으로 이웃해의 범위가 넓으면 넓을수록 한 반복(iteration) 내에서 고려할 수 있는 이웃해들 중 더 좋은 해가 존재할 가능성이 높아지지만 동시에 각 반복마다 가장 좋은 해를 고르기 위해 더 많은 이웃해들을 고려해야 하므로 주어진 시간동안 탐색할 수 있는 반복 횟수가 줄어들게 된다. 따라서 대상 문제에 따라 효과적인 탐색이 가능하도록 이웃해의 범위를 적절하게 유지할 필요가 있다. 이웃해를 만드는 방법 중 가장 단순한 방법은 현재해 x에 포함된 근무표 중 하나를 제거하고 근무표 풀에 포함된 근무표들 중 하나를 추가하여 새로운 이웃해를 만드는 것이다. 실험 결과에 의하면 이와 같이 이웃해를 정의할 경우 너무 쉽게 국소 최적해에 빠지는 경향이 있으며 해의 질도 좋지 않음을 확인할 수 있었다. 따라서 본 연구에서는 단 1개의 근무표를 교체하는 방법을 확장하여 임의의 k개까지를 교체하는 방법을 도입하였다. 그림 3에서 이웃해 집합 S_k는 k개의 근무표를 교체하여 생성된 이웃해들을 의미한다. 전체 이웃해 집합은 $S_1 \cup S_2 \cup ... \cup S_k$로 정의되며 이 때 k 값은 실험적으로 결정된 값인 5로 설정하였다.

이웃해 생성 시 고려해야 할 다른 한 가지는 S_k에 포함된 근무표의 개수를 정하는 것이다. 교체되는 근무표

```
Find an initial solution x
Define tabu structure
count = 0
repeat until stopping condition is met
  Generate neighborhood sets of x : S1, S2, ..., Sk
  Select the best non-tabu solution x' from S1, S2, ..., Sk
  x ← x'
  if x is better than the current best-so-far solution then
     count = 0
  else count = count + 1
  if count > threshold then
     Update x by calling Integer Programming
     count = 0
```

그림 3 타부 탐색과 정수계획법의 결합 알고리즘

의 개수가 늘어남에 따라 이웃해의 개수가 기하급수적으로 증가하게 된다. 따라서 교체되는 근무표의 개수를 2개까지만 허용한다 하더라도 매 반복마다 모든 이웃해들을 생성하는 것은 불가능하다. 그렇다고 일정 규모의 이웃해를 무작위로 선택하게 되면 현재해보다 더 좋은 해가 나올 가능성이 낮아지게 된다. 따라서 일정 규모의 이웃해들만을 생성하되 좋은 이웃해를 생성할 수 있는 효과적인 방법이 요구된다. 본 연구에서 하나의 이웃해는 현재해에 포함된 근무표들 중 일부 근무표들을 제거한 후 근무표 풀로부터 제거된 근무표들을 대체할 같은 개수의 근무표들을 선택하여 추가하는 과정을 순차적으로 수행함으로써 만들어진다. 제거되는 근무표는 현재해를 구성하는 근무표들 중 중요도가 가장 낮다고 판단되는 정도에 따라 확률적으로 선택되는데, 한 근무표의 중요도는 해당 근무표에 의해서만 커버되고 있는 레그들의 개수에 비례한다. 즉, 편승 레그를 많이 포함하는 근무표일수록 삭제될 확률이 높아지게 된다. 근무표 풀로부터 추가될 근무표를 선정하는 방법은 초기해를 생성할 때와 마찬가지로 그리디 휴리스틱을 사용한다. 각각의 이웃해 집합 S_k에 포함되는 이웃해의 개수는 k 값과 마찬가지로 실험적으로 결정하였다. 생성된 이웃해들 중 공백 레그의 개수를 최소화하는 이웃해를 다음해로 설정하게 되는데 만약 공백 레그의 개수가 동일한 해들이 존재할 경우에는 목적함수 값이 가장 좋은 해를 선택한다. 목적함수는 출퇴근 순서 제약을 얼마나 많이 만족하고 있느냐에 따라 계산되어진다.

시뮬레이티드 어닐링과 같이 하나의 이웃해만을 생성하고 다음해로의 이동 여부를 결정하는 경우에는 이웃해를 개념적으로 $S_1 \cup S_2 \cup ... \cup S_k$에 포함된 하나의 원소로 정의하되 타부 탐색에서처럼 미리 일정 규모의 이웃해들을 생성하는 것이 아니라 매 반복마다 k개 이하를 교체하여 하나의 이웃해를 생성하게 된다. 만약 생성된 이웃해의 공백 레그 개수가 현재해보다 적거나 공백 레그의 개수가 같더라도 목적함수 값이 더 좋을 경우에는

무조건 이동하게 되며 이외의 경우에는 확률적으로 이동하게 된다.

일정한 반복 횟수가 경과한 후에도(즉, *count* > *threshold*) 지금까지의 가장 좋은 해보다 더 좋은 해를 발견하지 못한다면 정수계획법 모듈이 호출된다. 일반적인 이웃해 탐색 기법에서는 이와 같은 경우 국소 최적해로부터 벗어나기 위해 탐색 다각화를 위한 여러 가지 방법들을 동원한다. 타부 탐색에 있어서 전형적인 탐색 다각화 방법으로는 지금까지의 해의 변화를 비교적 긴 기간동안 저장해 놓은 후 향후 탐색 방향을 조정하기 위해 이 정보를 활용하는 방안이 있다. 예를 들어 탐색 방향을 지금까지와는 다른 방향으로 유도하고자 한다면 지금까지 해를 변화시키는 과정 중에 교체가 많이 이루어졌던 근무표들에 대해서는 또 다시 교체되는 것을 금지하거나 벌점을 부과하는 것이다. 본 연구에서는 정수계획법을 해의 개선뿐 아니라 이웃해 탐색 기법의 탐색 다각화를 위해 활용하고 있다. 정수계획법 모듈의 호출이 결정되면 호출하기 전에 먼저 현재해를 대폭 변화시키기 위해 이웃해 탐색 수행 시와는 달리 현재해로부터 비교적 많은 근무표들을 제거하되 이웃해 탐색 때와 마찬가지로 각 근무표의 중요도에 기반한 확률에 따라 제거하게 된다. 그 후 정수계획법 모듈이 호출되는데 정수계획법의 역할은 가장 많은 레그들을 커버할 수 있도록 근무표 풀로부터 제거된 개수만큼의 근무표를 선택하는 것이다. 정수계획법 호출 시 교체되는 근무표의 개수는 정수계획법을 호출할 때마다 점차 증가하게 된다. 공백 레그의 개수가 적으면 적을수록 더 많은 근무표를 교체해야만 해의 개선을 기대할 수 있기 때문이다. 본 연구의 실험에서는 첫 번째 호출 시의 교체 개수를 20으로 설정하였고 이후로는 매 호출 시마다 3개씩 증가시켰다. 실험을 통해 정수계획법이 다른 다각화 방법에 비해 공백 레그를 줄이는 데 보다 효과적임을 확인하였다. 뿐만 아니라 정수계획법에 의해 변화된 해는 이전해에 비해 대폭 수정된 상태이므로 이후의 이웃해 탐색 과정을 통해 이전과는 다른 영역에서의 탐색이 가능해지고 이로 인해 해의 개선이 보다 쉬어지는 것으로 나타났다.

비록 정수계획법 호출 시 교체되는 근무표의 개수를 약 20여개 정도로 제한함으로써 문제의 규모를 대폭 줄였다 하더라도 정수계획법은 최적해를 도출하기 위해 탐색 공간을 완전히 열거하는 방식으로 동작하기 때문에 여전히 과도한 시간이 소요될 가능성이 있다. 따라서 정수계획법이 수행되는 최대시간을 설정하고 최적해를 찾지 못할 경우에는 그 때까지의 가장 좋은 해를 다음해로 지정하였다. 정수계획법은 대상 문제의 목적함수 및 제약조건이 모두 선형식으로 표현될 경우에만 효율적으로 최적해 탐색을 수행할 수 있는 알고리즘이다. 따라서 출퇴근 순서 목적함수는 정수계획법을 수행하는 동안 명시적으로 고려될 수 없다. 그러나 정수계획법이 수행된 후 다시 출퇴근 순서를 감안하는 이웃해 탐색 기법이 수행되는 과정이 반복되기 때문에 큰 문제가 없는 것으로 확인되었다.

5. 실험 결과

본 논문에서 제안한 알고리즘의 검증을 위해 2장에서 기술한 바와 같이 실제 운행중인 지하철 승무일정계획 문제를 대상으로 구현 및 실험을 수행하였다. 모든 실험은 Pentium-III 800 PC에서 수행되었으며 정수계획법 적용을 위한 개발 도구로는 ILOG CPLEX[12]를 사용하였다. 대상 문제의 목표는 차량운행계획표에 존재하는 640개의 레그를 모두 커버할 수 있는 65개의 근무표 집합을 찾아내는 것이다. 개별 근무표 제약조건을 만족하는 모든 근무표의 개수는 대략 30억 개 정도에 달할 것으로 추정되는데 이 중 대기횟수가 2회인 근무표들이 약 18만개 정도이고 대기횟수가 3회인 근무표들이 약 5천만개 정도이며 나머지는 대기횟수가 4회인 근무표들이다. 근무표 생성 단계에서 모든 근무표를 생성하는 것은 분명 불가능한 일이다. 근무표를 모두 생성했다 하더라도 최적조합 선정 단계에서 대상 근무표 풀을 효과적으로 다루는 것은 불가능하다. 본 연구에서는 최적조합 선정을 위해 대기횟수가 2회인 근무표들만을 대상으로 하였다. 이는 최적조합 선정 결과 대기횟수가 2회인 근무표들만을 포함하게 함으로써 이후 단계인 휴리스틱 교정 단계에서 보다 쉽게 공백 레그들을 제거할 수 있도록 하기 위함이다. 앞서 설명한 바와 같이 휴리스틱 교정 단계를 수행하는 도중에 대기횟수가 많은 근무표들이 점차 증가하는 등 근무표의 질이 저하되는 경향을 확인하였다. 따라서 최적조합 선정 단계의 수행 결과가 이미 대기횟수가 3회 또는 4회인 근무표들을 포함하게 된다면 휴리스틱 교정 단계를 통해 공백 레그들을 완전히 제거하는 것이 매우 힘들어지게 된다.

첫 번째 실험에서는 정수계획법만을 적용하였을 때 공백 레그의 개수를 살펴보았으며 아울러 집합 커버링 모델(IP/SCP)과 최대 커버링 모델(IP/MCP)을 사용했을 때 각각의 결과를 살펴보았다. 수행 시간은 각 실험 당 2시간으로 제한하였으며 대기횟수가 2회인 18만개의 근무표들 중 최소 1,000개부터 최대 20,000개까지 다양한 규모의 근무표를 선택해 만든 집합들을 대상으로 하였다. 실험 결과 IP/MCP에 의해 도출된 결과는 대상 근무표의 개수가 20,000개일 때 공백 레그의 개수가 24로 가장 좋았으며 IP/SCP의 결과(집합 커버링 문제를 푼 후 초과한 개수만큼의 근무표를 제거하는 방법)는 대상 근무표의 개수가 15,000개일 때 공백 레그의 개수가 47

로 가장 좋았다. 이 실험으로부터 집합 커버링 문제를 푼 후 근무표를 제거하는 것보다 최대 커버링 문제를 푸는 것이 공백 레그를 최소화하는 데 훨씬 효과적임을 알 수 있다. 20,000개를 초과한 근무표 집합을 대상으로 할 경우에는 과도한 수행 시간 및 메모리를 요구하는 정수계획법의 특성 상 주어진 시간 내에 더 좋은 해를 구하는 데 어려움이 있었다.

표 1은 이웃해 탐색 기법들 중 언덕 오르기 탐색을 최대 커버링 모델에 적용한 결과로서 이웃해 생성 시 교체하는 근무표의 개수를 달리함에 따른 공백 레그의 개수를 나타낸 것이다. 각 결과는 10회 실험 결과를 요약한 것으로 각 실험 당 수행 시간은 2시간으로 제한하였으며 대상 근무표 풀로는 대기 횟수가 2회인 18만개의 근무표 집합을 사용하였다. 실험 결과에 의하면 근무표를 1개만 교체할 때보다 2개 이상의 근무표 교체를 허용할 때 공백 레그의 개수가 현저히 줄어듦을 알 수 있다. 이웃해의 범위를 보다 넓게 정의하고 좋은 이웃해를 찾기 위한 교체 방법을 사용함으로써 보다 좋은 국소 최적해로의 이동이 가능한 것으로 판단된다. 그러나 2개 이상의 근무표 교체를 허용한 모든 경우에 있어서 거의 비슷한 결과를 보였는데 이는 이웃해의 범위가 너무 커짐에 따라 한 반복 당 수행 시간이 증가하여 주어진 시간 내에 수행되는 반복 횟수가 감소하기 때문이다.

표 1 언덕 오르기 탐색의 수행 결과

공백 레그 개수	[illegible]	[illegible]	[illegible]	[illegible]	7개 이하
[illegible]	35	17	16	16	16
[illegible]	32.4	15.4	15.1	14.9	15
[illegible]	29	14	14	14	14

표 2는 여러 가지 이웃해 탐색 기법에 대한 실험 결과를 요약한 것으로서 언덕오르기 탐색(HC)과 시뮬에이티드 어닐링(SA) 그리고 타부 탐색(TS)을 비교하였으며 타부 탐색 기법에 다각화를 추가한 실험 결과(TS+D)를 함께 비교하였다. 실험은 표 1의 실험과 같은 조건으로 수행되었으며 이웃해 생성을 위해 5개 이하의 근무표 교체를 허용하였다. 실험 결과에 의하면 이웃해 탐색 기법들 사이에 큰 차이가 없음을 알 수 있다. 심지어는 언덕 오르기 탐색까지도 타 기법에 뒤지지 않음을 알 수 있다. 이것은 이웃해 생성 시 교체되는 근무표의 개수를 5개까지 허용함으로써 나타난 효과라 할 수 있다. 앞에서 설명한 바와 같이 교체되는 근무표 개수를 늘리게 되면 한 반복 내에서 고려할 수 있는 이웃해의 개수가 훨씬 많아지게 된다. 실험에서도 교체되는 근무표 개수를 5개까지 허용한 경우 현재해의 이웃해들 중에는 현재해보다 좋거나 동등한 해가 항상 존재하고 있었다. 따라서 언덕 오르기 탐색의 경우에도 알고리즘의 중단 없이 항상 다른 해로의 이동이 가능해지고 경우에 따라 해의 개선이 이루어지기도 하였다. 실제로 1개의 근무표만을 교체한 실험에서 HC, SA, TS의 평균 공백 레그 개수가 각각 32.4, 18.8, 19.6으로 나온 것을 보면 교체되는 근무표 개수가 이웃해 탐색 기법의 성능에 영향을 미침을 알 수 있다. 그러나 타부 탐색에 있어서 탐색 다각화는 별다른 추가적인 효과가 없는 것으로 나타났다.

표 2 이웃해 탐색 기법의 수행 결과

공백 레그 개수	HC	SA	TS	TS+D
최대	16	17	16	16
평균	14.9	15.2	14.8	14.7
최소	14	14	14	14

표 3은 각각의 이웃해 탐색 기법과 정수계획법을 결합했을 때의 결과를 나타낸 것으로 이전 실험과 마찬가지로 각 실험 당 수행 시간은 2시간으로 제한하였으며 총 10회 실험 결과를 요약하였다. 이 실험 결과 역시 이웃해 탐색 기법들 사이의 차이는 거의 없는 것으로 나타났다. 그러나 표 2와 비교해 볼 때 이웃해 탐색 기법과 정수계획법을 결합함으로써 이웃해 탐색 기법을 단독으로 사용할 때보다 공백 레그의 개수가 대폭 감소되고 있음을 확인할 수 있다. 또한 정수계획법만을 사용했을 경우의 가장 좋은 결과인 24개보다 공백 레그의 개수가 훨씬 적음을 알 수 있다. 정수계획법만을 사용할 경우에는 알고리즘의 특성 상 매우 제한된 근무표 집합만을 대상으로 할 수밖에 없기 때문에 좋은 결과를 기대하기 힘들다. 이웃해 탐색 기법과 정수계획법을 결합한 알고리즘에서 정수계획법은 각 실험 당 대략 4번 정도 호출되고 있다. 간혹 정수계획법 수행 중 최적해를 찾기 위해 과도한 시간이 소요되는 경우가 있으므로 너무 많은 시간이 소요되지 않도록 한번의 정수계획법 수행 시간을 최대 20분으로 제한하였다.

표 4는 최적조합 선정 시 정수계획법(IP), 이웃해 탐색(NS), 두 기법을 결합한 기법(HYBRID)을 적용하여 생성된 결과를 대상으로 휴리스틱 탐색 교정을 수행한

표 3 하이브리드 기법의 수행 결과

공백 레그 개수	HC/IP	SA/IP	TS/IP
최대	13	14	12
평균	12.2	11.8	11.2
최소	10	10	10

표 4 휴리스틱 교정 결과

	IP	NS	HYBRID
공백 레그 개수	24	14	10
평균근무시간	10시간 33분	10시간 38분	10시간 34분
기계횟수 (2회:3회:4회)	33:24:8	37:21:7	34:27:4
출퇴근 순서 위배	5	4	4
수행시간	16시간 42분	1시간 42분	38분

결과의 예를 보인 것으로 대부분 이와 같은 경향의 결과를 보이고 있다. 표에서와 같이 공백 레그가 10개인 경우에는 공백 레그를 모두 제거하는 데 38분밖에 소요되지 않은 반면에 공백 레그가 14개와 24개인 경우에는 각각 1시간 42분과 16시간 42분이 소요되었다. 이것은 최적조합 선정 단계에서 공백 레그의 개수를 최소화하는 것이 얼마나 중요한 것인가를 보여주는 것으로서 두 기법을 결합한 방법이 가장 좋음을 알 수 있다. 해의 질에 있어서는 모든 기법들이 거의 유사한 결과를 보였으나 HYBRID의 경우 모든 면에서 약간 더 좋게 나타났다. 이는 전문가에 의해 수립된 결과와 비슷하거나 약간 좋은 결과이다. 참고로 수 개월에 걸쳐 전문가에 의해 작성된 계획의 평균근무시간은 10시간 34분이며 출퇴근 순서 제약을 위배하는 경우는 3개가 있었다. 그리고 대기횟수가 2회, 3회, 4회인 근무표의 개수는 각각 33개, 26개, 6개이다.

6. 결 론

본 논문은 대규모 승무일정계획 문제를 효과적으로 해결하기 위해 이웃해 탐색 기법과 정수계획법을 결합하는 방안을 제시하고 있으며 실제 운행중인 지하철 승무일정계획 문제를 대상으로 구현 및 실험을 수행함으로써 알고리즘의 성능을 검증하였다. 제안한 알고리즘에서 이웃해 탐색 기법은 순수한 정수계획법만을 적용할 때보다 더 큰 규모의 근무표 집합을 쉽게 다룰 수 있게 함으로써 보다 좋은 해를 발견할 가능성을 높여주고 있다. 뿐만 아니라 이웃해 탐색 기법은 비선형 제약조건 및 목적함수가 포함된 문제에도 쉽게 적용할 수 있다는 장점이 있다. 정수계획법은 이웃해 탐색 기법의 틀 내에서 보다 좋은 해를 찾기 위한 도구로 활용될 뿐 아니라 긴 시간동안 해의 개선이 없다고 판단될 경우 탐색 다각화를 위한 역할을 담당하고 있다. 정수계획법을 호출할 경우에는 정수계획법만을 단독으로 사용할 때와는 달리 근무표들 중 일부만을 교체하도록 하여 복잡도가 훨씬 낮은 부문제(subproblem)를 풀게 함으로써 더 많은 근무표 집합을 대상으로 하는 데 큰 어려움이 없도록 하였다. 실험 결과 본 논문에서 제안한 알고리즘은 이웃해 탐색 기법과 정수계획법의 장점을 결합함으로써 각 기법을 단독으로 사용할 때보다 훨씬 더 좋은 결과를 얻을 수 있음을 확인하였다.

참 고 문 헌

[1] L. Bodin, B. Golden, A. Assad, and M. Ball, "Routing and scheduling of vehicles and crews: the state of the art," Computers and Operations Research, 10:63-211, 1983.

[2] A. Caprara, M. Fischetti, P.L. Guida, P. Toth, and D. Vigo, "Solution of large-scale railway crew planning problems: The Italian experience," Technical Report OR-97-9, DEIS University of Bologna, 1997.

[3] S. Ceria, P. Nobili, and A. Sassano, "A Lagrangian-based heuristic for large-scale set covering problems," Mathematical Programming, 81:215-228, 1998.

[4] J.E. Beasly, and P.C. Chu, "A genetic algorithm for the set covering problem," European Journal of Operational Research, 94:392-404, 1996.

[5] S. Lavoie, M. Minoux, and E. Odier, "A new approach for crew pairing problems by column generation with an application to air transportation," European Journal of Operations Research, 35:45-58, 1988.

[6] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance, "Branch and price: Column generation for huge integer programs," Operations Research, 46:316-329, 1998.

[7] 황준하, 박춘희, 이용환, 류광렬, "정수계획법과 휴리스틱 탐색기법의 결합에 의한 승무일정계획의 최적화", 정보과학회논문지, 8(2):195-205, 2002.

[8] E. Aarts, J. Korst, and P. Laarhoven, "Simulated annealing," Local Search in Combinatorial Optimization, John Wiley & Sons, 91-120, 1997.

[9] F. Glover, and M. Laguna, Tabu search, Kluwer Academic Publishers, 1997.

[10] L.A. Wolsey, Integer programming, Wiley, 1998.

[11] R. Church, and C. ReVelle, "The maximal covering location problem," Papers of the Regional Science Association, 32:101-118, 1974.

[12] ILOG CPLEX, Reference and User Manual, Version 7.0, 2000.

황 준 하

1995년 부산대학교 컴퓨터공학과 학사
1997년 부산대학교 컴퓨터공학과 석사
2002년 부산대학교 컴퓨터공학과 박사
2002년 9월~현재 금오공과대학교 컴퓨터공학부 전임강사. 관심분야는 인공지능, 최적화, 일정계획, 기계학습

류 광 렬

1979년 서울대학교 전자공학과 학사 1981년 서울대학교 전자공학과 석사 1983년 3월~1984년 8월 충북대학교 컴퓨터공학과 전임강사. 1992년 University of Michigan 전기 및 컴퓨터공학과박사 1992년 3월~1993년 2월 Scientific Research Lab., Ford Motor Company, 선임연구원. 1993년 3월~현재 부산대학교 컴퓨터공학과 교수. 관심분야는 인공지능, 기계학습, 최적화, 정보검색