정수계획법과 휴리스틱 탐색기법의 결합에 의한 승무일정계획의 최적화

(Crew Schedule Optimization by Integrating Integer Programming and Heuristic Search)

황 준 하 † 박 춘 희 †† 이 용 환 ††† 류 광 렬 ††††
(Junha Hwang) (Choonhee Park) (Yong Hwan Lee) (Kwang Ryel Ryu)

요 약 승무일정계획이란 특정 기간동안 운행할 차량들을 대상으로 각 차량마다 필요로 하는 승무원을 배정하는 계획을 말한다. 최적 승무일정계획의 수립은 일반적으로 가능한 모든 종류의 개별 근무표들을 생성한 다음 이들을 대상으로 투입 승무원의 수가 최소화 될 수 있는 최적조합을 선정하는 방식으로 이루어지고 있다. 근무표 최적조합의 선정을 위한 종래의 기법들은 주로 선형계획법에 기반을 두고 있으나, 목적함수에 선형식으로 표현하기 어려운 요소가 포함되어 있을 경우 적용이 어렵다는 문제가 있다. 본 논문은 선형식으로 표현하기 어려운 목적함수를 포함할 뿐만 아니라 동원 가능한 승무원의 수가 제한되어 있는 경우에도 계획 수립이 가능하도록, 기존의 정수계획법에 휴리스틱 탐색기법을 결합하는 방안을 제시한다. 휴리스틱 탐색은 정수계획법에 의해 일차로 도출된 계획의 불완전한 부분을 교정하기 위해 반복적 개선 탐색을 수행하는 방식으로 이루어진다. 기존의 방법으로 해결이 어려운 실제 현장의 승무일정계획 문제를 대상으로 한 실험 결과, 본 논문의 방법은 전문가의 수작업 결과보다 더 좋은 수준의 계획을 빠른 시간 내에 수립할 수 있음을 확인하였다.

키워드 : 승무일정계획, 정수계획법, 휴리스틱 탐색

Abstract Crew scheduling is the problem of pairing crews with each of the vehicles in operation during a certain period of time. A typical procedure of crew schedule optimization consists of enumerating all possible pairings and then selecting the subset which can cover all the operating vehicles, with the goal of minimizing the number of pairings in the subset. The linear programming approach popularly adopted for optimal selection of pairings, however, is not applicable when the objective function cannot be expressed in a linear form. This paper proposes a method of integrating integer programming and heuristic search to solve difficult crew scheduling problems in which the objective function cannot be expressed in linear form and at the same time the number of crews available is limited. The role of heuristic search is to improve the incomplete solution generated by integer programming through iterative repair. Experimental results show that our method outperforms human experts in terms of both solution quality and execution time when applied to real world crew scheduling problems which can hardly be solved by traditional methods.

Key words : Crew Scheduling, Integer Programming, Heuristic Search

1. 서 론

† 학생회원 : 부산대학교 컴퓨터공학과
jhhwang@hyowon.cc.pusan.ac.kr
†† 비 회 원 : LG전자 네트워크연구소
chepark@lge.com
††† 비 회 원 : 부산대학교 컴퓨터공학과
ygleel@hyowon.cc.pusan.ac.kr
†††† 종신회원 : 부산대학교 컴퓨터공학과 교수
부산대학교 컴퓨터 및 정보통신연구소 연구원
krryu@hyowon.cc.pusan.ac.kr
논문접수 : 2001년 12월 6일
심사완료 : 2002년 1월 2일

승무일정계획이란 특정 기간동안 운행할 차량들을 대상으로 각 차량마다 필요로 하는 승무원을 배정하는 계획을 말하며, 전체 투입 승무원의 수를 최소화하는 것이 계획의 주된 목표가 되는 경우가 많다. 승무일정계획에 앞서서 수립되는 차량운행계획은 각 차량별로 출고에서 입고까지 각 역에서의 도착시각과 출발시각을 결정한다. 차량 운행 중 승무원의 교대가 가능한 역 사이의 구간을 레그(leg)라 하며 차량운행계획은 레그들의 집합으로 표현된다. 승무일정계획에서 승무원 1인(혹은 1조)의 근무 계

획인 근무표는 출근 후 첫 레그로부터 시작하여 퇴근 시의 마지막 레그에 이르기까지 여러 개의 레그로 구성된다. 이들 레그는 승무원이 중간에 쉬는 시간이나 차량을 갈아타는 방법에 따라 매우 다양한 형태로 구성될 수 있지만 기본적으로는 근로조건에 따른 제약을 만족하는 것이어야 한다. 승무일정계획의 목표는 이러한 근무표들을 적절히 선택하여 차량운행계획의 모든 레그를 운행할 수 있게 하되 근무표의 총 수를 최소화하는데 있다. 그러나, 기본적으로 차량운행계획의 레그 수가 많을 경우 제약조건이 어떻게 주어지느냐에 따라 생성 가능한 개별 근무표의 수가 방대해질 수 있고, 이 때 근무표의 수를 최소화하는 최적조합을 선정하는 것은 매우 복잡도가 높은 문제가 된다.

근무표 최적조합의 선정을 위한 종래의 기법들은 주로 선형계획법에 기반을 두고 있다. 선형계획법 적용의 전제조건은 대상 문제의 목적함수와 제약조건들이 모두 선형식으로 표현되어야 하는 것이다. 본 논문은 선형식으로 표현하기 어려운 목적함수를 포함할 뿐 아니라 동원 가능한 승무원의 수가 제한되어 있을 경우에도 계획 수립이 가능하도록, 기존의 정수계획법에 휴리스틱 탐색기법을 결합하는 방안을 제시하고 있다. 정수계획법 역시 선형계획법에 기반을 두고 있으므로 대상 문제는 선형식으로 표현되어야 한다. 본 논문에서 제안하는 방법에서는 정수계획법을 적용하기 위해 선형식으로 표현하기 어려운 목적함수를 문제로부터 제거하는 대신 다른 제약조건을 추가함으로써 간접적으로 그 목적함수가 지향하는 바가 달성되도록 유도하고 있다. 그러나 아래의 3장에서 설명하듯이 이로 인해 최적조합 선정 문제의 난이도는 더욱 높아져 실행 가능한 계획의 수립이 어려워지게 되고 결국 최적조합 선정 결과 실행 가능한 계획의 수립이 불가능해진다. 휴리스틱 탐색은 이렇게 정수계획법에 의해 도출된 실행 불가능한 계획을 교정하여 실행 가능한 계획으로 만들기 위해 반복적 개선 탐색을 수행하는 방식으로 이루어진다.

실험 결과, 기존의 방법으로 해결이 어려운 실제 현장의 승무일정계획 문제에 대하여 본 논문의 방법은 전문가의 수작업 결과보다 더 좋은 수준의 계획을 빠른 시간 내에 수립할 수 있음을 확인하였다.

이하의 구성은 다음과 같다. 2장에서는 승무일정계획 수립을 위한 관련 연구를 검토하고 3장에서는 본 논문이 대상으로 하는 지하철 승무일정계획 문제에 대해 상세히 소개한 후 기존 승무일정계획 수립 방법의 적용 시 문제점을 설명한다. 4장에서는 본 논문에서 제안하는 정수계획법과 휴리스틱 탐색기법의 결합 방안에 대해 기술하고 5장에서는 실험 결과를 분석한다. 마지막으로 6장에서 결론 및 향후 과제를 제시한다.

2. 관련 연구

일반적으로 승무일정계획은 난이도가 매우 높은 문제이기 때문에 먼저 생성 가능한 모든 종류의 근무표를 생성한 후 이들을 대상으로 근무표의 수가 최소화 될 수 있는 최적조합을 선정하는 방식으로 이루어지고 있으며 각각의 단계를 근무표 생성 단계와 최적조합 선정 단계라고 한다.

근무표 생성 단계에서는 열거 방법 등을 사용하여 개별 근무표 제약조건을 만족하는 근무표들을 생성한다. 그러나 소규모 승무일정계획 문제를 제외한 대부분의 승무일정계획 문제의 경우 제약조건들을 만족하는 근무표의 개수가 적게는 수백만 개에서 많게는 수십억 개에 이르기 때문에 모든 근무표들을 생성하는 것은 불가능하다. 따라서 모든 근무표들을 생성하기보다는 일정 개수의 근무표들을 선별적으로 생성하는 방법을 사용하고 있다.

최적조합 선정 단계는 아래의 식 (1)과 같이 set covering 문제로 표현된다[1]. set covering 문제는 n개의 열(column)들 중 최소의 개수를 선택하여 m개의 행(row)을 모두 cover하는 문제로 정의된다. x_j는 0 또는 1의 값을 가지는 결정변수로서 j번째 열이 선택될 경우 1의 값을 취하게 되며 a_{ij}는 m개의 행과 n개의 열로 이루어진 행렬로서 j번째 열이 i번째 행을 cover할 경우 1의 값을 취하게 된다. c_j는 j번째 근무표의 비용을 나타내며 단순히 근무표 개수를 최소화하는 문제라면 c_j의 값은 1이 된다. 따라서 선택되는 열의 개수를 최소화하는 것이 목표이며 이 때 각 행들이 선택된 열들 중 최소 한 개 이상의 열에 의해 cover되어야 한다.

$$\begin{aligned} \text{Minimize} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{Subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i = 1,\ldots,m \\ & x_j = (0,1), \quad j = 1,\ldots,n \end{aligned} \tag{1}$$

그림 1은 다섯 개의 행과 네 개의 열로 이루어져 있는 set covering 문제의 예로서 각 열마다 cover하고 있는 행이 서로 다르다. 예를 들면 첫 번째 열은 1, 3, 4 행을 cover하고 있으며 두 번째 열은 2, 4 행을 cover하고 있다. 네 개의 열들 중 적절한 부분집합을 선택하여 모든 행을 cover하는 방법은 단 한 가지만 존재하는 것이 아니다. 열 1, 2, 3, 4를 선택하는 경우, 열 1, 2, 3을 선택하는 경우, 열 1, 3을 선택하는 경우 등이 모두 이에 해당한다. 이 해들 중 열의 수가 가장 최소화되는 경우는 열 1, 3을 선택하는 경우이다.

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

그림 1 set covering 문제의 예

최적조합 선정 문제에서 개별 근무표는 열에 대응되며 레그는 행에 대응된다. 따라서 최적조합 선정 단계의 목표는 모든 레그들이 한 개 이상의 근무표에 의해 운행될 수 있도록 근무표의 조합을 선정하되 근무표의 개수를 최소화하는 것이다. 그림 1을 최적조합 선정 문제의 예라고 가정한다면 첫 번째 근무표와 세 번째 근무표를 선정할 경우 가장 적은 개수의 근무표를 사용하여 모든 레그의 운행이 가능하게 된다. 이 때 레그 1, 2, 4, 5는 한 개의 근무표에 의해 운행되며 레그 3은 두 개의 근무표에 의해 운행된다. 이와 같이 두 개 이상의 근무표에 의해 운행되는 레그를 편승 레그라고 부르며 반면에 어떤 근무표에 의해서도 운행이 되지 않는 레그를 공백 레그라고 한다.

승무일정계획 문제에서는 모든 레그를 운행해야 한다는 제약조건 외에도 선택된 근무표들의 조합에 대한 별도의 제약조건이 추가될 수 있으며 식 (2)와 같이 표현될 수 있다. 예를 들어 b_{ij}가 j번째 근무표의 근무시간이고 v_I이 선택된 근무표 개수에 특정 근무시간을 곱한 값이라면 첫 번째 수식은 최대 평균근무시간을 특정 근무시간 이하로 제한하는 제약조건을 의미하게 된다. 이와 유사한 방식으로 p개의 제약조건을 표현할 수 있다.

$$\text{Subject to} \quad \sum_{j=1}^{n} b_{ij} x_j \le v_i, \quad i = 1,\ldots, p \qquad (2)$$

승무일정계획 문제를 해결하기 위해 동원된 방법들로는 Lagrangian relaxation, 유전 알고리즘(genetic algorithm), 열 생성 기법(column generation) 등이 있다[2][3][4]. 열 생성 기법을 제외한 방법들은 고정된 개수의 근무표들을 대상으로 set covering 문제를 효과적으로 풀기 위한 방안들을 제시하고 있다. 따라서 근무표들을 모두 생성할 수 있는 소규모 승무일정계획 문제에 적용할 경우에는 효과적이지만 근무표 개수가 매우 많은 대규모 승무일정계획 문제에는 부적합하다. 반면에 열 생성 기법은 최적조합 선정 결과를 바탕으로 필요한 근무표를 동적으로 생성하는 방법으로서 잠재적으로 모든 근무표들을 고려할 수 있는 기법으로 알려져 있다. 그러나 열 생성 기법은 선형계획법을 기반으로 하고 있기 때문에 set covering 모델의 목적함수와 제약조건식이 모두 선형식으로 표현되어야 하며 선형식으로 표현하기 어려운 요소가 포함되어 있을 경우에는 적용이 어렵다는 문제가 있다.

3. 지하철 승무일정계획에 대한 기존의 방법 적용

본 논문은 지하철 승무일정계획 문제를 대상으로 한다. 본 장에서는 대상 문제인 지하철 일정계획의 수립 과정을 설명하고 기존 방법을 적용할 경우 발생할 수 있는 문제점 및 해결 방안에 대해 기술한다.

3.1 지하철 승무일정계획

지하철 일정계획은 열차운행계획, 승무일정계획, 승무원 배정계획으로 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 다음 단계의 계획 수립 결과에 따라 이전 단계의 수정이 요구되기도 한다. 이와 같이 단계들 사이의 이상적인 연계가 요구되나 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.

본 논문의 대상 문제에서 지하철 선로는 그림 2와 같이 하나의 직선으로 표현된다. 노선 상에는 여러 역들이 존재하며 선로의 양쪽 끝 역에는 승무소(A, B)와 차량기지가 위치해 있다. 승무원은 승무소에서 출퇴근을 하게 되며 열차 운행 중 휴식을 취하기도 한다. 승무소 외에 승무원의 교대가 가능한 역이 노선의 중간역(C)에 위치해 있다. 따라서 출발역과 방향에 따라 A→C, C→B, A←C, C←B와 같이 총 4가지 종류의 레그가 존재한다. 각 열차들은 차량기지가 존재하는 역으로부터 출고하여 하루 일과를 시작하며 승무소 사이를 왕복 운행하다가 최종적으로 차량기지로 입고함으로써 하루 일과를 마무리하게 된다.

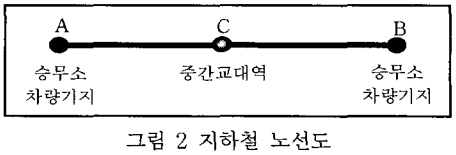

그림 2 지하철 노선도

그림 3은 차량운행계획 결과로 생성된 차량계획표를 단순화시킨 예로서 총 4개의 차량으로 이루어져 있다. 첫 번째 차량은 승무소 A와 B 사이에 2개의 레그가 존재하므로 총 8개의 레그로 이루어지며 차량기지 A로부터 사업을 시작하여 승무소 사이를 2회 왕복한 후 다시 차량기지 A로 입고함으로써 하루 일과를 마무리한다.

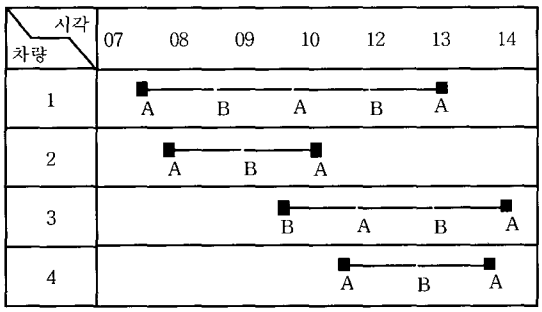

그림 3 차량 운행계획표

승무일정계획은 열차운행계획 결과를 기반으로 수립된다. 승무일정계획 문제는 제약조건과 목적함수로 표현될 수 있다. 제약조건은 각 근무표가 준수해야 할 제약조건과 최종적으로 생성된 근무표들의 집합이 준수해야 할 제약조건으로 분류되며 각각을 지역적 제약조건과 전역적 제약조건이라고 부른다.

지역적 제약조건들로는 다음과 같은 제약조건들이 존재한다. 근무표의 시작과 완료는 승무소에서만 가능하고 시작 승무소와 완료 승무소는 동일해야 하며 그 승무소를 해당 근무표의 소속 승무소라고 한다. 대기는 승무소와 중간대기역에서만 가능하고 각 근무표의 총 대기 횟수는 근로조건을 고려한 최소 대기횟수와 최대 대기횟수 범위 내에 존재해야 한다. 승무소 사이의 편도 운행을 1회 운행이라고 정의할 때 각 근무표는 지정된 횟수를 운행해야 한다. 만약 한 근무표의 운행 횟수가 홀수로 지정되어 있다면 시작 승무소와 완료 승무소가 동일해야 한다는 제약조건으로 인해 중간교대역 C에서의 교대가 불가피하게 된다. 그리고 운행 횟수 제약조건에 의해 각 근무표는 같은 개수의 레그를 운행하게 되는데 이 때 레그 하나를 운행하는 것을 태스크(task)라고 부른다. 승무원이 출근하여 퇴근할 때까지의 시간을 근무시간이라고 하며 각 근무표의 근무시간은 지정된 범위 내에 존재해야 한다. 대기하지 않고 연속으로 운전할 수 있는 최대 운전시간이 지정되어 있으며 대기 시에는 현재 운행중인 차량과 갈아타는 차량의 입출고 여부에 따라 주어진 범위의 대기시간을 만족해야 한다. 그림 4는 이상의 제약조건을 모두 만족하는 근무표의 예이다.

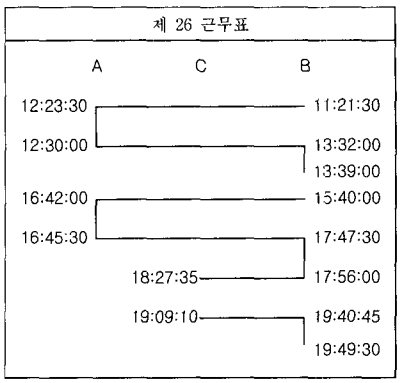

그림 4 근무표의 한 가지 예

전역적 제약조건은 다음과 같다. 최종적으로 생성된 근무표의 집합은 모든 레그들을 운행할 수 있어야 하며 승무소 A 소속의 근무표 개수와 승무소 C 소속의 근무표 개수가 지정되어 있다. 이외에도 평균근무시간의 상한값이 정해져 있고 대기횟수 별 근무표 개수의 최소값과 최대값이 정해져 있다.

제약조건 외에 달성해야 할 목적함수는 다음과 같다. 첫 번째, 대기횟수가 적은 근무표가 많을수록 좋다. 즉, 대기횟수가 적은 근무표를 최대한 많이 선택해야 하며 대기횟수가 많은 근무표의 개수를 최소화해야 한다. 두 번째, 평균근무시간을 최소화해야 한다. 마지막으로는 출퇴근 순서와 관련된 사항으로서 먼저 출근한 승무원이 이후에 출근한 승무원보다 늦게 퇴근하는 경우를 최소화해야 한다.

3.2 기존 방법 적용의 어려움

본 논문의 대상문제를 해결하기 위해 기존 방법과 마찬가지로 근무표 생성 단계와 최적 조합 선정 단계로 분리하여 접근하는 방법을 고려해 볼 수 있다. 먼저 근무표 생성 단계에서는 열거 방법 등을 동원하여 적절한 개수의 근무표를 생성한다.

최적조합 선정 단계에서는 문제를 set covering 문제로 모델링해야 한다. 그런데 근무표 개수가 고정되어 있기 때문에 기존의 set covering 모델과는 달리 근무표 개수가 목적함수가 아닌 제약조건으로 변경되어야 하며 대신 개별 근무표의 대기횟수 최소화 등이 목적함수로 추가되어야 한다. 이 때 선형계획법에 기반한 기법들을 동원하고자 한다면 목적함수와 제약조건들이 선형식으로 표현될 수 있어야만 한다. 목적함수들 중 대기횟수의 최소화는 대기횟수가 많은 근무표에 벌점을 부과하여

식 (1)의 c_j를 변경함으로써 해결이 가능하며 평균근무시간도 마찬가지로 개별 근무표의 평균근무시간에 비례하여 벌점을 부과하면 된다. 그러나 출퇴근 순서는 단순히 c_j의 변경만으로는 표현이 불가능하다.

다음과 같은 방법을 적용한다면 출퇴근 순서를 선형식으로 표현하는 것이 불가능한 것은 아니다. i번째 근무표의 출근시간과 퇴근시간을 각각 s_i, f_i라고 하면 n개의 모든 근무표로부터 출퇴근 순서 제약조건을 위배하는 근무표 쌍들의 집합 $V = \{(i, j) \mid s_i<s_j \text{ and } f_j<f_i,\ 1\le i,j\le n\}$를 정의할 수 있다. 그리고 z_{ij}를 출퇴근 순서 제약조건을 위배하는 근무표 i와 j가 동시에 선택될 경우 1의 값을 가지고 동시에 선택되지 않을 경우 0의 값을 가지는 결정변수라고 하면 출퇴근 순서 목적함수는 z_{ij}의 합을 최소화하는 것이 된다. 즉, 출퇴근 순서를 위배하는 근무표 쌍이 동시에 선택되는 경우를 최소화하는 것이다. 단, i번째 근무표와 j번째 근무표가 동시에 선택될 경우 z_{ij}의 값이 1이 되도록 i번째 근무표의 선택 결정변수인 x_i와 j번째 근무표의 선택 결정변수인 x_j를 사용하여 $(x_i + x_j - 1 \le z_{ij})$라는 제약조건을 추가해야 한다. 이상을 식으로 표현하면 식 (3)과 같다.

$$\begin{aligned} &\text{Minimize} && \sum_{(i,j)\in V} z_{ij} \\ &\text{subject to} && x_i + x_j - 1 \le z_{ij}, \quad (i,j)\in V \\ & && x_i = (0,1), \quad i = 1,\ldots n \\ & && z_{ij} = (0,1), \quad (i,j)\in V \end{aligned} \tag{3}$$

비록 이상과 같이 출퇴근 순서 목적함수를 선형식으로 표현하는 것이 불가능한 것은 아니지만 결정변수의 개수와 제약조건의 개수가 너무 많아져 정수계획법과 같은 선형계획법에 기반한 기법의 적용이 불가능해진다. 예를 들어 근무표의 개수가 10,000개라고 가정하면 최악의 경우 1억 개라는 매우 큰 수의 도입변수가 필요하게 되며 제약조건 역시 같은 개수만큼 필요하게 된다. 따라서 이와 같은 경우 이론적으로는 선형식으로 표현하는 것이 가능하지만 실질적으로 선형계획법에 기반한 기법을 적용하는 것은 불가능하다고 할 수 있다.

선형계획법에 기반한 기법을 적용할 수 있도록 문제를 표현하기 위한 간접적인 방법으로는 set covering 모델을 적용하기 이전 단계에서 새로운 제약조건을 추가함으로써 set covering 모델 적용 시 해당 목적함수를 제거하는 방법이 있을 수 있다. 예를 들어 최적조합 선정 단계에서 사용할 근무표들을 선택할 때 개별 근무표의 근무량을 특정한 범위 내로 제한한다면 최적조합 선정 시 출퇴근 순서에 관한 별도의 표현식이 없더라도 출퇴근 순서 목적함수를 어느 정도 만족하게 된다. 극단적으로 최적조합 선정 시 사용할 개별 근무표의 작업시간을 특정 시간으로 고정한다면 출퇴근 순서는 100% 만족하게 된다. 그러나 이와 같은 엄격한 제약조건을 추가한다면 최적조합 선정 단계에서 실행 가능한 해를 도출하는 것이 불가능해진다. 이와 같이 비선형 목적함수를 제거한 후에는 기존의 정수계획법과 같은 선형계획법에 기반한 기법을 적용할 수 있게 되지만 최적조합 선정 결과로 도출된 해는 실행 불가능한 해가 된다. 문제는 실행 불가능한 해들 중에서 제약조건의 위배가 최소인 해를 구하는 것이다. 만약 최대 평균근무시간과 대기횟수별 근무표 개수 제약을 완화한다면 모든 레그의 운행이 가능하면서 평균근무시간과 전체대기횟수를 최대한 만족하는 계획의 수립이 요구되고 반대로 모든 레그를 운행해야 한다는 제약을 완화한다면 이외의 모든 제약조건들을 만족하면서 레그를 최대한 많이 운행할 수 있는 계획의 수립이 요구된다. 어떤 경우든 원래 주어진 제약조건들을 모두 만족하지는 못하기 때문에 수립된 계획으로부터 출퇴근 순서 목적함수 등의 목적함수를 최대한 고려하면서 실행가능한 해로 변환하는 과정이 필요하다.

4. 정수계획법과 휴리스틱 탐색 기법의 결합

본 논문에서 제시하는 방안은 근무표 생성, 최적 조합 선정, 휴리스틱 교정 단계로 나누어지며 세 단계가 순차적으로 수행된다. 먼저 근무표 생성 단계에서는 이후의 단계에서 사용할 개별 근무표 집합을 생성한다. 최적조합 선정 단계에서는 모든 레그를 운행해야 한다는 제약조건을 완화하여 최대한 많은 레그를 운행할 수 있는 계획을 수립하며 휴리스틱 교정 단계에서는 이 계획으로부터 출발하여 모든 레그를 운행할 수 있는 계획을 도출하기 위해 반복적 개선 탐색 과정을 수행한다. 근무표 생성과 최적조합 선정 단계에서는 각각 제약만족 탐색기법과 정수계획법을 적용하였으며 휴리스틱 교정 단계에서는 대상 문제에 적합한 휴리스틱 탐색 기법을 개발하여 적용하였다.

4.1 근무표 생성

근무표 생성 단계에서는 기존의 방법과 마찬가지로 최적 조합 선정 단계와 휴리스틱 교정 단계에서 사용할 근무표의 집합을 생성한다. 그런데 모든 근무표를 생성하는 것은 불가능하므로 이후의 단계에서 처리가 가능한 규모의 근무표 집합을 생성하였다.

본 논문에서는 근무표 생성을 위해 제약만족 탐색기법을 적용하였다[5]. 제약만족 탐색기법에서는 문제를

변수, 도메인, 제약조건으로 표현하며 도메인은 각 변수가 취할 수 있는 값의 집합을 의미한다. 제약만족 탐색 기법을 사용하면 해를 탐색할 때 제약전파를 통해 변수들의 도메인을 축소시킴으로써 불필요한 backtracking을 효과적으로 줄일 수 있다. 따라서 하나의 해를 생성한 후 제약조건을 검사하여 합격 여부를 판단하는 단순한 열거 방법보다 훨씬 빨리 원하는 근무표들의 집합을 생성할 수가 있다.

4.2 최적조합 선정

최적조합 선정 단계에서는 근무표 생성 단계에서 생성된 근무표들을 대상으로 정수계획법을 적용하여 공백 레그를 최소화할 수 있는 최적 조합을 선정한다[6]. 그런데 정수계획법 적용 시 근무표 생성 단계에서 생성한 모든 근무표들을 수용할 수 없기 때문에 일부분만을 선택하여 사용하되 가능하면 모든 레그들이 골고루 운행 가능하도록 선택하였다.

정수계획법을 적용하기 위해서는 set covering 모델의 목적함수와 제약조건들을 선형식으로 표현해야 한다. 먼저 선형식으로 표현하기 어려운 출퇴근 목적함수를 간접적으로 달성하기 위해 근무표 선택 시 근무시간의 범위를 제한하였다. 이로 인해 최적조합 선정 단계에서는 근무표 개수 제약조건을 비롯한 모든 제약조건을 만족하는 계획의 수립이 불가능해지므로 근무표 개수 제약조건을 완화하고 근무표 개수의 최소화를 목적함수로 추가하였다. 따라서 최적조합 선정 결과는 지정한 근무표 개수를 초과하게 되며, 근무표 개수 제약조건을 만족하기 위해 초과한 개수만큼의 근무표를 제거해야 하는 문제가 발생한다. 본 논문에서는 이 문제를 최적조합 선정 문제의 부문제라고 정의한다.

지정된 근무표 개수를 d라고 하고 set covering 문제를 풀고 난 후의 근무표 개수를 n이라고 하면 ($d < n$)이므로 근무표 개수 제약조건을 만족하기 위해서는 n개의 근무표들 중 d개를 선택해야만 한다. 부문제의 목표는 공백 레그의 개수를 최소화하는 것이며 부문제 역시 정수계획법으로 모델링이 가능하다.

부문제의 정수계획법 모델은 기본적으로 maximal covering 문제와 동일하다[7]. x_j를 n개의 근무표들 중 j번째 근무표가 선택될 경우 1의 값을 취하고 선택되지 않을 경우 0의 값을 취하는 결정변수라 하고 a_{ij}를 j번째 근무표가 레그 i를 운행할 때 1의 값을 취하는 이진 상수라고 하자. 그리고 y_i를 i번째 레그가 1개 이상의 근무표에 의해 운행될 경우 0의 값을 가지고 공백 레그가 될 경우 1의 값을 가지는 결정변수라고 한다면 목적함수는 식 (4)와 같다.

$$\text{Minimize} \sum_{i=1}^{m} y_i \tag{4}$$

고려해야 할 제약조건은 두 가지가 존재한다. 첫 번째 제약조건은 n개의 근무표들 중 d개를 선택해야 한다는 것이며 이는 식 (5)와 같이 표현된다.

$$\sum_{j=1}^{n} x_j = d \qquad x_j = (0,1), \quad j = 1,\ldots,n \tag{5}$$

두 번째 제약조건은 만약 i번째 레그가 공백 레그라면 y_i의 값은 1이 되고 한 개 이상의 근무표에 의해 운행된다면 0이 된다는 것으로서 식 (6)과 같이 표현될 수 있고 좀 더 간결하게 표현하면 식 (7)과 같다.

$$y_i = \begin{cases} 1 \text{ if } \sum_{j=1}^{n} a_{ij} x_j = 0 \\ 0 \text{ if } \sum_{j=1}^{n} a_{ij} x_j \geq 1 \end{cases} \tag{6}$$

$$\sum_{j=1}^{n} a_{ij} x_j + y_i \geq 1 \qquad y_i = (0,1), \quad i = 1,\ldots,m \tag{7}$$

따라서 부문제의 정수계획법 모델을 정리하면 식 (8)과 같다.

$$\begin{aligned} &\text{Minimize} \sum_{i=1}^{m} y_i \\ &\text{subject to} \sum_{j=1}^{n} x_j = d \\ &\qquad x_j = (0,1), \quad j = 1,\ldots,n \\ &\qquad \sum_{j=1}^{n} a_{ij} x_j + y_i \geq 1 \\ &\qquad y_i = (0,1), \quad i = 1,\ldots,m \end{aligned} \tag{8}$$

최적조합 선정 단계에서는 먼저 set covering 문제를 푼 후 부문제로 식 (8)과 같은 maximal covering 문제를 풀면 된다. 그런데 set covering 문제와 maximal covering 문제를 단계적으로 푸는 것보다 set covering 문제를 풀기 위해 준비한 근무표들을 대상으로 바로 maximal covering 문제를 푼다면 공백 레그의 최소화 측면에서 더 좋은 결과가 도출될 수 있다. 이 때의 정수계획법 모델은 앞서 설명한 부문제의 모델링과 동일하며 여기에 set covering 문제에서 사용하고 있는 전역적 제약조건을 추가하면 된다.

최적조합 선정을 위해 근무표 생성 단계에서 생성한 근무표 집합으로부터 근무표들을 선택할 때 대기횟수와 근무시간 측면에서 다양한 근무표들을 선택하느냐, 아니면 대기횟수가 적은 양질의 근무표들만을 선택하느냐에 따라 최적조합 선정과 휴리스틱 교정 단계에 미치는 영

향이 다를 것이라고 예상할 수 있다. 먼저 최적조합 선정 시 공백 레그를 최소화하기 위해서는 다양한 근무표들을 생성하는 것이 대기횟수가 적은 양질의 근무표만을 생성하는 것보다 더 좋을 것으로 예상된다. 그러나 실험에 의하면 예상과는 달리 두 가지 선택 방법에 따른 차이는 없는 것으로 나타났다. 정수계획법을 적용하기 위해서는 근무표 개수를 수만 개 이하로 제한할 수 밖에 없으며 이 개수는 실제 생성 가능한 모든 근무표들에 비하면 매우 적은 개수이다. 그리고 정수계획법의 특성상 제한된 시간 내에 실제 최적해를 구하는 것 또한 매우 어렵다. 따라서 제한된 시간 내에 제한된 개수의 근무표들을 대상으로 공백 레그를 최소화하기 위한 해를 구할 경우에는 근무 조건이 다양한 근무표들이 선택되었을 경우와 양질의 근무표들만을 선택한 경우의 차이는 없다고 할 수 있다. 뿐만 아니라 최적조합 선정 단계에서 생성된 결과를 초기해로 사용하는 휴리스틱 교정 단계에서는 공백 레그를 제거하기 위해 양질의 근무표들을 보유하는 것이 유리하므로 최적조합 선정 단계에서 대기횟수가 적은 근무표들만을 대상으로 하는 것이 더 좋은 방법이라고 할 수 있다.

4.3 휴리스틱 교정

최적조합 선정 단계의 수행 결과 공백 레그가 발생하는 동시에 두 개 이상의 근무표에 의해 운행되고 있는 편승 레그도 발생하게 된다. 이 때 편승 레그를 운행하고 있는 근무표의 태스크를 편승 태스크라고 부른다. 휴리스틱 교정 단계의 목표는 모든 공백 레그를 운행할 수 있도록 편승 태스크들을 반복적으로 수정하는 것이다. 편승 태스크를 이동함으로써 발생할 수 있는 상황들 중 공백 레그의 제거를 위해 다음과 같은 세 가지 상황을 고려해 볼 수 있다.

상황 ① : 편승 태스크가 공백 레그를 운행하면서 제약조건을 만족한다. 그림 5는 근무표들이 레그를 운행하는 상황을 단순화시킨 예로서 3개의 근무표와 6개의 레그를 표현하였다. 개별 근무표에 대한 제약조건으로는 레그 개수가 2개로 지정되어 있고 태스크 간에는 시간적으로 겹쳐서는 안 되며 태스크 사이의 대기시간은 30분을 초과할 수 없는 것으로 가정하였다. 계획 A의 경우 레그 1과 레그 3이 편승 레그이며 레그 2와 레그 5가 공백 레그이다. 만약 편승 태스크인 3-1이 레그 2를 운행하도록 근무표 3을 수정한다면 계획 B와 같이 제약조건을 위배하지 않고도 공백 레그 하나를 해결할 수 있게 된다.

상황 ② : 공백 레그를 운행하지는 못하지만 제약조건을 만족한다. 계획 B에서 먼저 태스크 1-2가 레그 4를 운행하도록 근무표 1을 수정한다. 이 때까지는 제약조건을 위배하지 않으면서 편승 태스크만 이동한 것이다. 그러나 이로 인해 태스크 3-2의 이동이 가능해지며 계획 C1과 같이 공백 레그 5가 해결된다. 즉, 현 단계에서 공백 레그를 해결하지 못하지만 상황 ①의 테스트 결과에 따라 다음 단계에서 공백 레그의 해결이 가능할 수도 있다.

상황 ③ : 공백 레그를 운행하지만 제약조건을 위배한다. 계획 B에서 태스크 1-1을 수정하는 대신 태스크 2-1이 레그 5를 운행하도록 조정한다면 계획 C2와 같이 공백 레그는 해결되지만 근무표 2가 제약조건을 위배하게 된다. 만약 레그 6을 또 다른 근무표가 운행하고 있다고 가정하면 근무표 2가 제약조건을 만족할 수 있도록 태스크 2-2를 조정할 수 있다. 그러나 실제로는 제약조건을 만족하기 위해 특정 레그를 수정하면 또 다른 레그들이 영향을 받기 때문에 새로운 공백 레그를 발생시키지 않으면서 근무표를 수정하는 것은 매우 어려운 일이다.

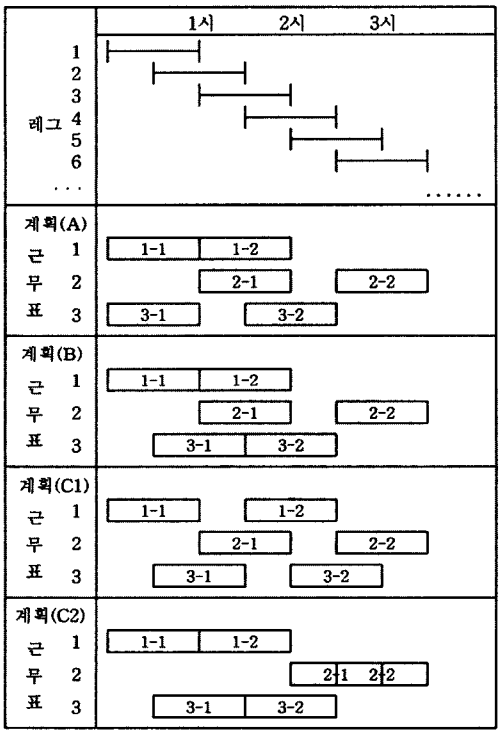

그림 5 휴리스틱 교정의 예

이상과 같은 상황을 고려하여 공백 레그를 제거하기 위한 휴리스틱 교정 알고리즘은 그림 6과 같다[8].

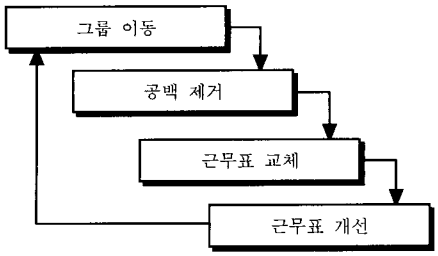

그림 6 휴리스틱 교정 알고리즘

상황 ③과 같이 편승 태스크 이동 시 제약조건을 위배하는 경우는 주로 그림 4에서 처음 네 개의 태스크와 같이 연속으로 연결되어 있는 레그들을 운행하는 태스크들 중 하나를 이동할 때 발생한다. 그룹 이동 단계에서는 이와 같은 문제가 발생하는 경우를 줄이기 위해 편승 태스크를 이동할 때 해당 편승 태스크의 전후로 연속해서 운행하는 태스크들을 그룹으로 지정하여 함께 이동하도록 하였다. 이동 시 공백 레그의 개수가 증가할 경우에는 이동이 불가능하기 때문에 편승 레그만을 이동할 때보다 이동할 수 있는 경우가 드물게 발생하지만 이동이 가능하다면 제약조건의 위배 없이 공백 레그를 해결할 수 있게 된다. 공백 제거 단계에서는 상황 ①과 ②를 반영하기 위해 iterative deepening search를 적용하였다[9]. 즉, 편승 태스크가 공백 레그를 운행할 수 있을 때까지 이동을 반복하되 실행 시간의 증가를 고려하여 지정된 최대 깊이까지만 탐색을 허용하였다.

그룹 이동 단계와 공백 제거 단계에서는 종류가 같은 레그들 사이의 이동만 가능하다. 예를 들어 A→C 종류의 레그를 운행하는 태스크는 또 다른 A→C 종류의 레그만 운행해야 하며 다른 레그를 운행하고자 할 경우에는 레그 사이의 도착역과 출발역이 달라지게 되어 운행이 불가능한 근무표가 생성된다. 따라서 공백 제거 시에도 같은 종류의 편승 레그가 공백 레그를 제거하게 되므로 종류별 편승 레그와 공백 레그의 분포 및 시간대가 매우 중요하다. 공백 제거 단계까지 수행한 후 남아있는 공백 레그들은 현재 공백 레그와 편승 레그의 상황 하에서는 해결이 어렵다고 판단된다. 근무표 교체 단계에서는 공백 레그와 편승 레그의 판도를 대폭 변경하기 위해 근무표를 교체함으로써 간접적으로 공백 레그와 편승 레그의 분포를 변경한다. 근무표 생성 단계에서 생성한 근무표 집합 내의 근무표들과 현재 계획에 존재하는 근무표들의 모든 쌍들을 교체해 본 후 공백이 가장 적은 한 쌍을 교체하게 된다. 단, 공백 레그의 변화를 위해 현재 공백 레그들 중 반드시 한 개 이상이 변경되도록 하였다. 만약 공백 레그의 수가 같다면 공백 레그의 변화가 큰 교체를 선호한다. 근무표 교체를 통해 공백 레그의 개수가 감소하는 경우도 발생하지만 대부분의 경우 같거나 일시적으로 증가하게 된다.

편승을 이동하거나 근무표를 수정하는 대부분의 경우 대기 횟수와 근무시간은 늘어나게 된다. 실험에 의하면 휴리스틱 교정 초반에 대기횟수와 근무시간 제약조건을 너무 많이 완화하면 초기에 해결되는 공백 레그의 개수는 많아지지만 최종적으로 해결되지 못하는 공백 레그가 남게 되므로 반복 횟수가 증가함에 따라 제약조건을 점진적으로 완화시키는 방법을 적용하였다. 같은 이유로 인해 최적조합 선정 단계에서 공백 레그를 최소화하는 것도 중요하지만 해의 질이 너무 나쁠 경우에는 휴리스틱 교정 단계에서의 공백 제거가 힘들게 되므로 최적조합 선정 단계에서는 해의 질이 어느 정도 보장된 상태에서 공백 레그를 최소화하는 것이 좋다.

근무표 개선 단계에서는 공백 제거 단계와 유사한 방법을 적용하여 각 근무표의 대기횟수를 줄임으로써 다음 반복 수행 시 보다 쉽게 공백 레그가 제거되도록 근무표의 질을 향상시킨다. 이상의 과정은 공백 레그가 모두 제거될 때까지 반복 수행된다. 실험에 의하면 탐색 초반에는 공백 레그가 빨리 제거되지만 공백 레그의 개수가 적은 상태에서는 더 이상의 개선 없이 공백 레그 개수의 증감이 반복됨을 알 수 있다. 이와 같은 상황이 발생할 경우에는 지금까지의 해들 중 공백 레그의 개수와 대기횟수 측면에서 가장 좋은 해들을 대상으로 하나를 선택하여 탐색을 재시작 하는 방법을 사용하였다.

5. 실험 결과

본 논문에서 제안한 방법의 검증을 위해 부산 지하철에서 실제 운행중인 차량 운행 계획을 대상으로 승무일정계획을 수립하였다. 근무표 생성 시 제약만족 프로그래밍 개발 도구로는 ILOG Solver를 사용하였으며[10] 최적조합 선정 시 정수계획법을 적용하기 위한 프로그래밍 개발 도구로는 ILOG CPLEX를 사용하였다[11].

대상 문제의 차량 개수는 총 57개이며 총 814개의 레그로 이루어져 있다. 각 근무표는 총 5회를 운행해야 하며 대기 횟수는 2회에서 4회까지 가능하다. 승무소별 근무표 개수는 A 소속과 B 소속이 각각 48개, 35개로 지정되어 있다.

근무표는 5회를 운행해야 하므로 총 10개의 태스크로 구성된다. 근무표 생성 단계에서는 이를 표현하기 위해 10개의 변수를 사용하였고 각 변수의 도메인은 814개의

레그를 표현하기 위해 0에서 813까지의 수를 사용하였다. 그리고 개별 근무표가 지켜야 할 제약조건을 추가한 후 근무표 생성을 위한 탐색을 수행한다. 총 백만 개의 근무표 집합을 생성하되 대기 횟수와 근무시간 대를 고려하여 다양한 근무표들을 생성하였다.

최적조합 선정 방법을 위한 실험 데이타는 근무표 선택 방법에 따라 DATA1과 DATA2로 나뉘어진다. DATA1은 대기횟수가 2회인 근무표들만으로 구성된 데이타이며 DATA2는 대기횟수가 2회, 3회, 4회인 근무표들이 골고루 선택된 데이타이다. 각 방법에 따라 선택되는 근무표의 개수를 달리하여 1,000개에서 20,000개까지 다양한 개수를 포함하는 데이타들을 준비하였다. 표 1은 set covering 문제와 maximal covering 문제를 순차적으로 적용한 경우(SCP+MCP)와 maximal covering 문제만을 적용한 경우(MCP)의 실험 결과를 나타낸 것이다. 최적조합 선정 시 DATA1에 대해서는 최대 평균근무시간을 10시간으로 제한하여 대기횟수뿐만 아니라 근무시간 측면에서도 양질의 근무표들이 선정될 수 있도록 하였다. 수행 시간은 각 데이타 당 2시간으로 제한하였다. 평가기준에서 LB(Lower Bound)는 정수계획법 적용 시 최초로 생성된 선형계획법의 결과로서 해당 데이타로부터 도출될 수 있는 해의 하한값을 의미하며 IP는 2시간이 경과한 후 최종적으로 생성된 해의 평가값을 의미한다. 즉, "SCP+MCP"에서 IP는 set covering 문제로부터 생성된 최소 근무표 개수를 나타내고 있으며 "MCP"에서 IP는 maximal covering 문제를 적용한 후의 최소 공백 레그 개수를 나타내는 것이다. "SCP + MCP"의 MCP는 set covering 문제로부터 도출된 해에 maximal covering 문제를 적용한 후의 최소 공백 레그 개수를 의미한다.

실험 결과에 의하면 근무표 개수가 증가함에 따라 하한값은 계속해서 좋아지고 있지만 실제 정수계획법 결과는 이에 비례하지 않음을 알 수 있다. 근무표 개수가 많을 경우 제한된 시간 내에 보다 최적에 가까운 해를 찾을 가능성이 적어지기 때문이다. 그리고 대기횟수가 다양한 근무표들로 구성된 데이타(DATA2)의 경우 근무표가 2회인 근무표들만으로 구성된 데이타(DATA1)보다 하한값은 더 작지만 결국 최종 정수해에 있어서는 오히려 더 좋지 않은 결과를 보이고 있다. 더욱이 표 1에 기술되지는 않았지만 최종적으로 생성된 근무표들의 질에 있어서 DATA1이 DATA2보다 훨씬 좋았다. 예를 들면 "MCP" 방법을 20,000개의 근무표에 적용했을 경우에는 DATA1의 근무표들의 대기횟수는 모두 2회인 반면 DATA2의 경우 대기횟수가 3회, 4회인 근무표가 각각 20개와 10개가 존재하고 있다. 비록 정수계획법을 적용하기 위해 근무표 개수를 최대 20,000개로 설정하였지만 20,000개의 근무표는 전체 근무표 중 극히 일부분에 불과하며 정수계획법의 특성상 제한된 시간 내에 최적해를 구하는 것은 매우 어렵다. 따라서 대기횟수 측면에서 골고루 선택된 데이타가 반드시 공백최소화를 위해 좋다고 말할 수 없으며 다음 단계인 휴리스틱 교정 단계를 고려한다면 오히려 양질의 근무표들만을 대상으로 하는 것이 좋다고 할 수 있다.

그림 7은 공백 레그가 24개인 결과를 대상으로 휴리스틱 교정을 적용하여, 약 1시간 동안 공백 레그의 개수와 근무조건의 변화를 그래프로 나타낸 것이다. 근무시간의 변화는 초기 계획의 평균근무시간인 9시간 50분을 기준으로 분 단위의 변화량을 의미하며 각 대기횟수의 변화는 근무표의 개수를 의미한다. 시간이 지남에 따라 공백 레그가 감소하는 반면에 평균근무시간이 증가하고

표 1 최적조합 선정 결과

방법	SCP + MCP						MCP			
근무표 선택	DATA1			DATA2			DATA1		DATA2	
평가기준 / 개수	LB	IP	MCP	LB	IP	MCP	LB	IP	LB	IP
1000	94.2410	95	50	94.0627	98	77	28.1779	38	30.5742	59
2000	93.6667	94	53	93.1250	99	63	25.0111	38	24.0058	48
3000	93.4232	94	47	92.7143	94	58	23.4873	31	22.4887	47
5000	93.0000	93	51	92.1458	96	60	22.8095	29	20.8115	38
7000	92.9091	93	50	92.1000	96	59	22.5211	28	20.5942	39
10000	92.7568	93	49	91.9317	95	60	22.0000	31	20.0437	45
15000	92.7500	93	47	91.9091	94	58	22.0000	32	19.8531	37
20000	92.4000	93	49	91.7597	99	64	20.8198	24	19.4710	36

있으며 대기회수가 3회, 4회인 근무표들의 개수가 점점 증가하고 있는 등 근로조건이 나빠지고 있다. 기본적으로 휴리스틱 교정 시 초기 계획의 근로조건이 좋지 않다면 근로조건을 유지하면서 공백 레그를 제거하는 것은 매우 어렵다. 따라서 공백 레그를 효과적으로 제거하기 위해서는 최적조합 선정 단계에서 공백 레그의 개수를 최소화해야 할 뿐만 아니라 근로조건이 최대한 좋은 계획을 수립해야만 한다.

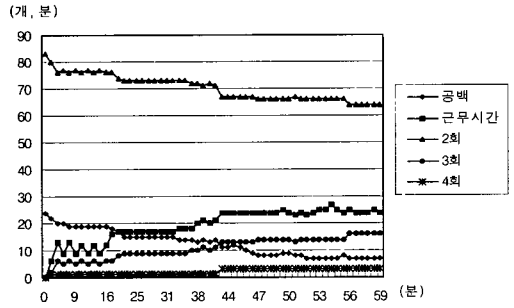

그림 7 휴리스틱 교정 시 공백의 변화

표 2는 휴리스틱 교정에 의해 수립된 최종 계획과 전문가에 의해 수립된 계획을 비교한 결과이다. 최종 계획을 수립하기 위해 약 10시간 정도의 시간이 소요되었지만 수개월이 소요되는 수작업을 고려한다면 시간이 매우 많이 단축되었음을 알 수 있다. 뿐만 아니라 평균근무시간을 10분 가량 단축할 수 있었고 대기횟수와 출퇴근순서 측면에서도 수작업에 비해 훨씬 좋은 계획이 수립되었다.

표 2 휴리스틱 교정과 수작업 결과의 비교

항목 \ 수립 방법	휴리스틱 교정	수작업
근무표	83	83
평균근무시간	10:23:53	10:33:43
대기횟수(2회:3회:4회)	39:24:2	33:26:6
출퇴근순서 위배	15	19

6. 결 론

본 논문은 승무일정계획 문제의 목적함수가 선형적으로 표현하기 어려운 요소를 포함할 뿐만 아니라 승무원의 수가 제한되어 있는 경우에 계획 수립이 가능하도록, 기존의 정수계획법과 휴리스틱 탐색기법을 결합하는 방안을 제시하였다. 먼저 정수계획법을 적용하기 전에 새로운 제약조건을 추가하여 선형적으로 표현하기 어려운 목적함수를 제거함으로써 정수계획법의 적용이 가능하도록 하였으며 근무표 최적조합 선정 시에는 maximal covering 모델로 문제를 정형화하여 공백 레그의 개수를 최소화하였다. 휴리스틱 교정 단계에서는 최적조합 선정 결과로 수립된 계획을 대상으로 공백 레그를 모두 제거하기 위해 반복적 개선 탐색 과정을 수행하였으며 실험결과 전문가의 수작업보다 훨씬 좋은 일정계획을 수립할 수 있음을 확인하였다. 휴리스틱 교정 시 공백 레그를 보다 빨리 제거하기 위해서는 최적조합 선정 단계에서 공백 레그의 개수를 최소화하는 것이 무엇보다 중요하다. 본 논문에서는 최적조합 선정을 위해 정수계획법을 적용하였지만 향후로는 타부 탐색 등과 같은 탐색 기법들을 적용하기 위한 방안에 관해 연구하고자 한다.

참 고 문 헌

[1] T. Grossmann, and A. Wool, "Computational Experience with Approximation Algorithms for the Set Covering Algorithm," *European Journal for Operations Research*, 101(1):81-92, 1997.

[2] S. Ceria, P. Nobili, and A. Sassano, "A Lagrangian-Based Heuristic for Large-Scale Set Covering Problems," *Mathematical Programming*, 81:215-228, 1998.

[3] J. E. Beasly, and P. C. Chu, "A Genetic Algorithm for the Set Covering Problem," *European Journal of Operational Research*, 94:392-404, 1996.

[4] 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.

[5] E. Tsang, *Foundations of Constraint Satisfaction*. Academic Press, 1996.

[6] L.A. Wolsey, *Integer Programming*, Wiley, 1998.

[7] B.T. Downs, and J.D. Camm, "An Exact Algorithm for the Maximal Covering Problem," *Naval Research Logistics*, 43, 435-461, 1996.

[8] 박춘희, 승무 일정계획의 개선을 위한 휴리스틱 교정 기법, 석사학위 논문, 부산대학교 2000.

[9] S. Russell, and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995.

[10] ILOG Solver, *Reference and User Manual*, Version 5.0, 2000.

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

황 준 하

1995년 부산대학교 컴퓨터공학과 학사. 1997년 부산대학교 컴퓨터공학과 석사. 1997년 3월 ~ 현재 부산대학교 컴퓨터공학과 박사과정. 관심분야는 인공지능, 최적화, 일정계획, 기계학습

박 춘 희

1998년 부산대학교 컴퓨터공학과 학사. 2000년 부산대학교 컴퓨터공학과 석사. 2000년 3월 ~ 현재 LG전자 네트워크연구소. 관심분야는 인공지능, 최적화, 일정계획

이 용 환

1999년 부산대학교 컴퓨터공학과 학사. 2001년 부산대학교 컴퓨터공학과 석사. 2001년 3월 ~ 현재 부산대학교 컴퓨터공학과 박사과정. 관심분야는 인공지능, 최적화, 일정계획

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 부교수. 관심분야는 인공지능, 기계학습, 사례기반추론, Genetic Algorithms, 최적화, 정보검색