철도차량소요계획을 위한 열거알고리듬

An Enumeration Algorithm for the Rolling Stock Requirement Plan

김성호* 김동희** 최태성***
Kim, Seongho Kim, Dong-Hee Choi, Tae-Sung

ABSTRACT

A routing is the path that an actual trainset follows as it moves from one train to another train in a timetable. The number of routings is equivalent to the number of trainsets required to cover the timetable. The primary factors of rolling stock requirement plan include evaluating the minimum number of routings. This can be formulated as a set partitioning problem and solved using enumeration method or column generation method. In this paper we presents an enumeration algorithm which is useful to implement the enumeration method for the rolling stock requirement plan.

---

1. 서론 및 배경

열차스케쥴설계는 철도운영기관이 해결해야할 문제중 가장 복잡한 문제이며 철도운영기관의 여러 측면에 큰 영향을 미치는 중요한 문제라고 할 수 있다. 열차스케쥴은 철도운영기관이 판매하는 교통서비스의 목록으로 각 열차의 출발역, 도착역, 출발시각, 도착시각이 포함되어 있는 목록이라 할 수 있다. 아래의 도표 1은 전형적인 열차스케쥴의 일부분을 나타낸 것이다.

도표 1 열차스케쥴의 예

열차번호	출발역	도착역	출발시각	도착시각
101	서울	부산	6:00	8:40
102	부산	서울	6:00	8:46
103	서울	부산	6:30	9:16
106	부산	서울	6:30	9:10

* 인하대학교, 경영대학, 조교수, 032-860-7768, shk7768@inha.ac.kr, 회원
** 한국철도기술연구원, 운영·정보시스템연구팀, 선임연구원, 공학박사, 031-460-5483, kdh777@krri.re.kr, 회원
*** 인하대학교, 경영대학, 교수, 032-860-7738, tschoi@inha.ac.kr, 비회원

출발 및 도착시각을 결정하고 나면 열차의 좌석수를 결정해야 한다. 일반적으로 이 문제는 몇가지 편성유형(fleet)을 설정해 두고 이를 열차에 할당하는 방식으로 접근하며 이를 편성유형할당문제(fleet assignment problem)라고 한다. 각 열차에 편성유형이 할당되면 철도운영기관이 보유하고 있는 차량으로 실행이 가능한가를 검토하기 위한 철도차량소요계획(rolling stock requirement plan)을 수립해야 한다.

기존의 철도차량소요계획문제에 대한 연구는 다상품네트워크흐름문제(multicommodity network flow problem)로 접근한 연구(Booler 1980; Wright 1989; Forbes, Holt and Watts 1991; Ziarati, et al. 1997)와 집합분할문제(set partitioning problem)로 접근한 연구(Ben-Khedher et al. 1998)로 대별할 수 있다. 특히 후자는 프랑스 SNCF의 스케쥴 최적화 시스템인 RailPlus에 반영되어 실무에서 활용되고 있다.

차량소요계획을 수립하기 위한 집합분할문제 접근법에서는 해를 구하기 위한 해법으로 열거법(enumeration method)과 열생성기법(column generation method)이 사용될 수 있다. 후보운용(candidate routing)[2]을 열거한 후 이가정수계획법모형(binary integer programming model)의 해를 구하는 열거법보다는 필요한 실행가능운용을 생성하며 해를 개선해 가는 열생성기법이 일반적으로 선호된다. 그러나 차량소요계획을 수립함에 있어서 고려해야 할 제약조건이 매우 다양하고 많을 경우[3]에는 열생성기법 보다는 열거법이 효과적일 수 있다.

본 논문에서는 열거법으로 차량소요계획을 수립할 경우 유용하게 사용될 수 있는 열거알고리즘(enumeration algorithm)을 제시하고자 한다.

2. 집합분할문제 접근법

차량소요계획을 위한 집합분할문제는 다음과 같은 이가정수계획법모형으로 나타낼 수 있다.

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

$$\text{subject to} \sum_{j=1}^{n} a_{ij} x_j = 1, \text{ for } i = 1, 2, \ldots, m \tag{2}$$

$$x_j = 0 \text{ or } 1, \text{ for } j = 1, 2, \ldots, n \tag{3}$$

여기서 i : 열차를 나타내는 첨자

j : 후보운용(candidate routing)을 나타내는 첨자

c_j : j번째 후보운용의 비용계수

a_{ij} : i번째 열차가 j번째 후보운용에 포함되는가의 여부를 나타내는 계수

$a_{ij} = 1$: i번째 열차가 j번째 후보운용에 포함되어 있음을 나타냄

$a_{ij} = 0$: i번째 열차가 j번째 후보운용에 포함되어 있지 않을 나타냄

---

2) 후보운용(candidate routing)은 제약조건을 만족하여 실행가능한 운용을 의미함.

3) 예를 들어 차량의 정비와 관련한 제약조건

열차스케쥴이 그림 1과 같이 주어져 있다고 하자[4]. 이 그림에서 수평선은 각 역에서 시간의 흐름을 나타낸다. 화살표는 꼬리에서 출발하여 머리가 나타내는 역에 도착하는 열차를 나타내며 화살표의 머리부분 옆에 적혀있는 번호는 열차번호를 나타낸다. 예를 들어 1번 열차는 A역에서 가장 먼저 출발하여 B역에 도착하는 열차이다.

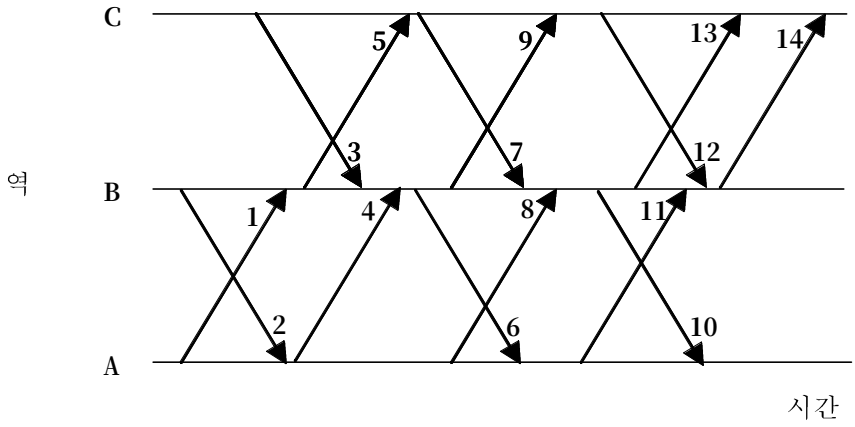

그림 1 열차스케쥴의 예

후보운용이 되기 위한 가장 기본적인 제약조건은 운용에 포함된 열차들 중에서 이웃한 두 열차가 ① 후행열차의 출발역과 선행열차의 도착역이 일치해야 하며, ② 후행열차의 출발시각이 선행열차의 도착시각 이후이어야 한다는 것이다. 아래의 도표 2는 그림 1의 열차스케쥴에서 기본 제약조건을 만족하는 후보운용 중 일부를 작성하여 나열한 것이다.

도표 2 그림 1의 열차스케쥴로부터 작성한 후보운용

후보 운용	구성열차	후보 운용	구성열차	후보 운용	구성열차
1	1-5-7-10	2	1-5-7-13	3	1-5-7-14
4	1-5-12-14	5	1-6-11-14	6	1-9-12-14
7	1-10	8	1-13	9	1-14
10	4-6-11-14	11	4-9-12-14	12	4-10
13	4-14	14	8-10	15	8-13
16	8-14	17	11-14	18	2-4-6-11-14
19	2-4-9-12-14	20	2-4-10	21	2-4-13
...	...	...	...	n	...

도표 2에는 그림 1에 나타낸 열차스케쥴로부터 작성 가능한 후보운용 중 일부를 나타낸 것이다. 기본제

4) 이 열차스케쥴은 차량소요계획을 위한 집합분할문제의 예를 제시하기 위해 Teodorović(1988)의 201 page에 있는 항공편스케쥴을 열차스케쥴로 변경한 것임.

약조건을 만족하는 후보운용 전체의 수를 n이라고 하자. 그림 2는 도표 2에 나타낸 후보운용으로 집합분할 문제의 제약조건 식(2)를 구성하여 행렬방정식(matrix equation)으로 나타낸 것이다. 열차의 수가 14이고 후보운용의 수가 n이므로 계수행렬의 크기는 $(14 \times n)$이 되며 따라서 결정변수벡터는 $(n \times 1)$의 열벡터가 된다. 우변상수벡터는 크기가 (14×1)이고 모든 요소가 1로 되어있다.

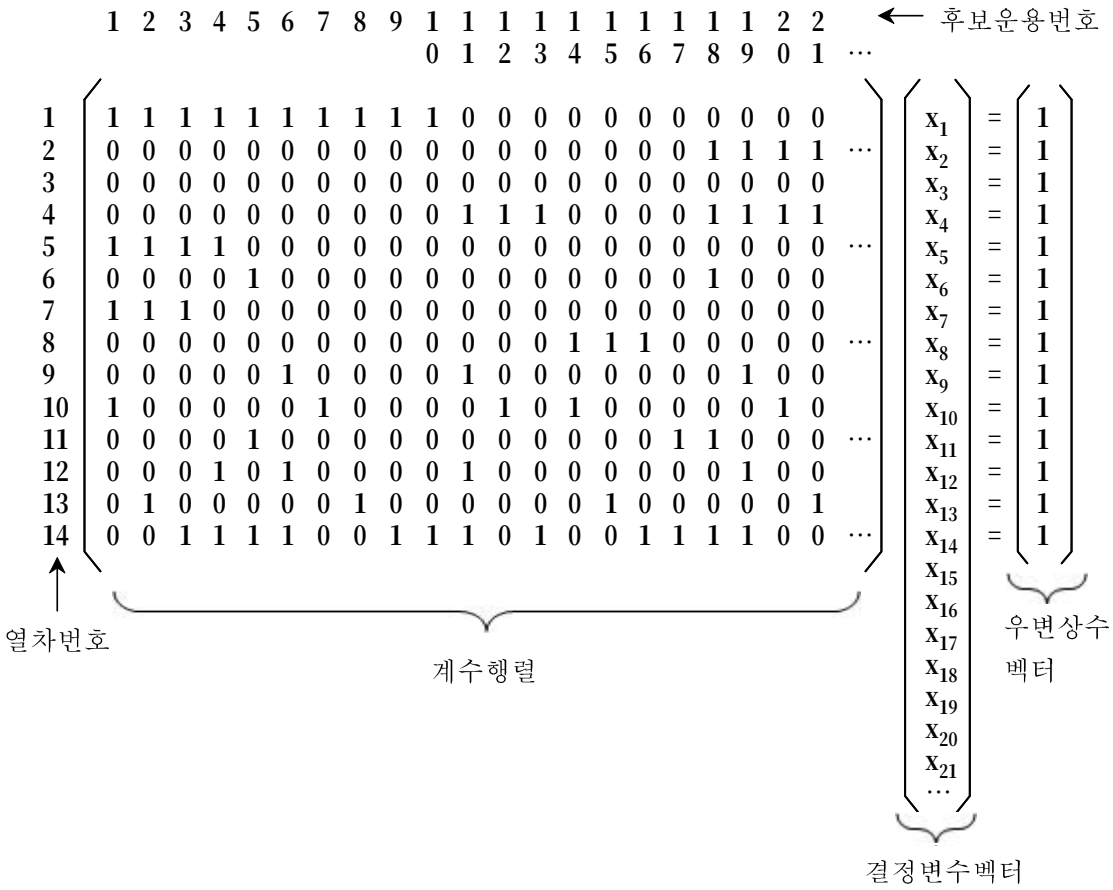

그림 2 행렬방정식으로 나타낸 집합분할문제의 제약조건

도표 2의 열차스케쥴에서 기본 제약조건만을 고려하여 작성가능한 모든 후보운용을 나열하면 그 수는 매우 많아진다. 열차스케쥴을 구성하는 열차의 수가 많아지면 후보운용의 수는 지수적으로 증가하게 되며 그에 따라 집합분할문제의 해를 구하기 위한 시간역시 지수적으로 증가한다.

열거법(enumeration method)은 가능한 모든 후보운용을 한번에 모두 나열한 후 이들로 구성된 집합분할문제의 해를 구한다. 반면에 열생성기법은 소수의 선택된 후보운용만으로 구성된 집합분할문제의 선형계획법완화(linear programming relaxation)의 해를 구하고 그 쌍대해(dual solution)를 이용하여 목적함수의 개선에 기여할 것으로 판단되는 후보운용을 생성하여 추가한다. 후보운용을 추가한 새로운 집합분할문제의 선형계획법완화의 해를 구하고 그 쌍대해를 이용한 후보운용 생성과 추가를 반복한다. 이러한 과정은 목적함수가 개선되지 않을 때까지 반복된다. 열생성기법에서 후보운용은 추가제약을 갖는 최단경로문제(shortest path problem with side constraints)로 생성하며 이 문제 대한 해법으로 일반화된 영구꼬리표 알고리즘(generalized permanent labelling algorithm)(Desrochers and Soumis 1988)과 제약프로그래밍

(constraint programming)(De Silva 2001)이 알려져 있다. 그림 3은 그림 1의 열차스케쥴을 시작노드와 종료노드를 갖는 네트워크로 표현한 것이며 이러한 네트워크에서 시작노드로부터 종료노드까지의 최단경로를 찾고 이를 후보운용으로 사용하는 것이다.

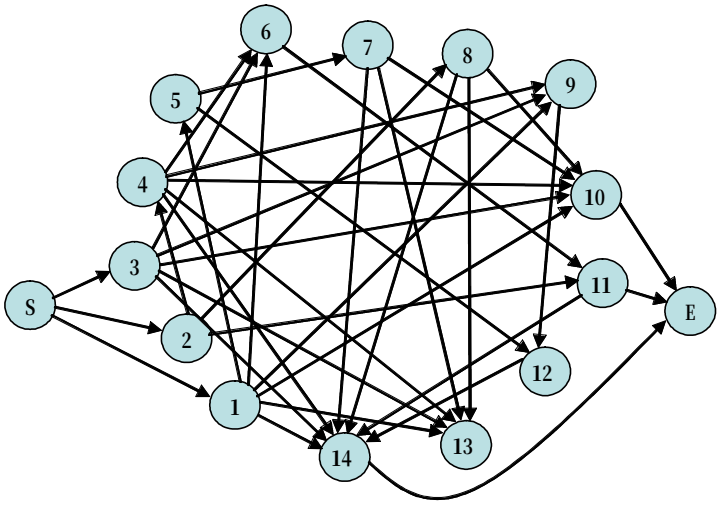

그림 3 후보운용을 생성하기 위한 네트워크

영구꼬리표알고리즘이나 제약프로그래밍은 그림 3과 같은 네트워크에 부여되는 추가제약조건을 융통성 있게 반영할 수 있으나 모든 형태의 제약조건을 자유롭게 반영하지는 못한다. 영구꼬리표알고리즘이나 제약프로그래밍으로 반영하기 어려운 제약조건이 존재하거나 또는 제약조건이 매우 많은 경우에는 열거법이 상대적으로 유효할 수 있다.

3. 열거알고리즘

열차시각표를 입력자료로 하여 기본 제약조건을 만족하는 가능한 모든 후보운용을 생성하는 열거알고리즘을 C++ 의사코드(pseudocode)(Neapolitan and Naimipour 2004)로 나타내면 다음과 같다.

```
void generateRotation (set_of_trains& tripVector);
{
    index last
    INIT();
    repeat
        note := nodeVector(last);
        nodeVector에서 nodeVector(last)를 제거함;
        t1 := node의 path(last)
        foreach t2 ∈ tripVector
            do if t1의 도착역 == t2의 출발역 and
```

$t1$의 도착시각 + 최소반복시간 < $t2$의 출발시각

then

$node$를 복사하여 $node2$를 생성;

$node2$의 $path$에 $t2$를 add;

$nodeVector$에 $node2$를 add;

$columnVector$에 $node2$를 add;

until $nodeVector == \varnothing$

}

$tripVector$는 열차시각표 개체를 나타낸다. $tripVector$의 하위개체는 열차시각표를 구성하는 각 열차가 된다. $last$는 개체에 포함되어 있는 마지막 하위개체의 위치를 나타내는 index이다. Node는 후보운용을 만들기 위한 $path$를 유지하고 있는 개체이다(그림 4 참조). $nodeVector$는 Node를 하위개체로 가지고 있는 벡터형 컨테이너[5]이다(그림 4 참조). 그리고 $node$는 탐색나무에서 현재 방문중인 Node를 나타낸다. $columnVector$는 후보운용을 보관하기 위한 벡터형 컨테이너이다.

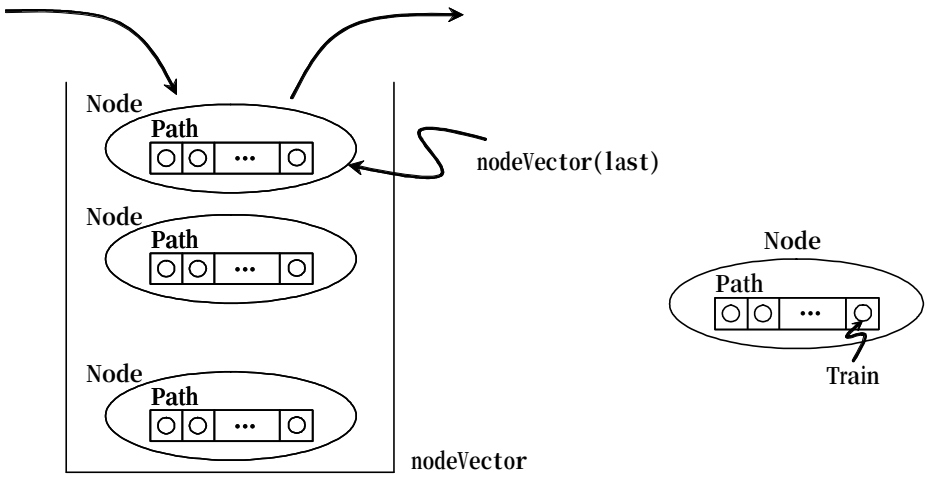

그림 4 Node와 $nodeVector$의 개념

4. 계산실험

앞에서 제시한 generateRotation을 C++로 구현[6]한 후 3가지 열차시각표에 대하여 계산실험을 수행하였다. 후보운용은 기본 제약조건(운용에 포함된 열차들 중에서 이웃한 두 열차가 ① 후행열차의 출발역과 선행열차의 도착역이 일치해야 하며, ② 후행열차의 출발시각이 선행열차의 도착시각 이후이어야 한다는 것)과 최소반복시간을 40분으로 설정하여 생성하였다. 집합분할문제의 목적함수 식(1)에서 후보운용의 비용

5) 표준템플릿라이브러리(Standard Templete Library: STL) 스택(stack)의 자료구조를 구현해 놓은 컨테이너이다. STL에 대한 자세한 내용은 Josuttis(1999)를 참조하기 바람.

6) generateRotation을 구현한 C++코드와 계산실험에 사용된 3가지 열차시각표는 http://seongho.inha.ac.kr/research/setpar에서 찾을 수 있음.

계수는 $c_j = 1$, for $j = 1, 2, \ldots, n$으로 설정하였다. 도표 3은 계산실험의 결과를 요약한 것이다.

도표 3 계산실험결과

열차시각표	열차시각표의 특성		계산실험 결과	
	열차수	포함된 역의 수	후보운용 수	집합분할문제의 최적해 (최소운용 수)
t160d1	160	5	26,729	33
t180d1	180	6	31,657	46
t192d1	192	8	29,303	45

5. 결론

본 논문에서는 차량소요계획을 수립할 경우 유용하게 사용될 수 있는 열거알고리즘(enumeration algorithm)을 제시하고 3가지 열차시각표를 대상으로 제시한 알고리즘에 대한 계산실험을 수행하였다. 본 논문에서 제시한 알고리즘은 추가제약을 갖는 최단경로문제에서 영구꼬리표알고리즘이나 제약프로그래밍으로 반영하기 어려운 제약조건이 존재하거나 또는 제약조건이 매우 많은 경우에 효과적으로 사용될 수 있을 것으로 기대된다.

본 논문에서는 간단한 3가지 열차시각표만을 대상으로 C++코드로 구현한 열거알고리즘의 계산실험만을 수행하였다. 향후 일정한 비교조건하에서 영구꼬리표알고즘, 제약프로그래밍을 활용하는 열생성기법과 다양한 측면에서 비교해볼 필요가 있다고 판단된다.

참고문헌

Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance (1996), Branch-and-Price: Column Generation for Solving Huge Integer Programs, Georgia Institute of Technology, School of Industrial and Systems Engineering

Ben Khedher, N., J. Kintanar, C. Queille, and W. Stripling (1998), "Schedule Optimization at SNCF: From Conception to Day of Departure," *Interfaces* Vol.28, No.1, pp. 6-23

Booler, J. M. P. (1980), "The Solution of a Railway Locomotive Scheduling Problem," *Journal of the Operational Research Society* Vol.31, pp. 943-948

De Silva, A. (2001), "Combining Constraint Programming and Linear Programming on an Example of Bus Driver Scheduling," *Annals of Operations Research* Vol.108, pp. 277-291

Desrochers, M., and F. Soumis (1988), "A Generalized Permanent Labelling Algorithm for the Shortest Path Problem with Time Windows," *Infor* Vol. 26, No. 3, pp. 191-211.

Forbes, M. A., J. N. Holt, and A. M. Watts (1991), "Exact Solution of Locomotive Scheduling Problems," *Journal of the Operational Research Society* Vol.42, No.10, pp. 825-831

Josuttis, N. M. (1999), *The C++ Standard Library: A Tutorial and Reference*, Indianapolis, IN, Addison-Wesley.

Neapolitan, R. E., and K. Naimipour (2004), *Foundations of ALGORITHMS Using C++*

Pseudocode, 3rd Edition, Jones and Bartlett Publishers Inc., MA.

Teodorovic, D. (1988), *Airline Operations Research*. Transportation Studies (N. Ashford, and W. G. Bell, Eds.) Gordon and Breach Science Publishers, New York

Wright, M. B. (1989), "Applying Stochastic Algorithms to a Locomotive Scheduling Problem," *Journal of the Operational Research Society* Vol.40, No.2, pp. 187-192

Ziarati, K., F. Soumis, J. Desrosiers, S. Gelinas, and A. Saintonge (1997), "Locomotive Assignment with Heterogeneous Consists at CN North America," *European Journal of Operational Research* Vol.97, pp. 281-292