항공교통흐름 제어 기법 연구

박 배 선 · 강 선 영 · 이 학 태
인하대학교 항공우주공학과

A Study of Air Traffic Flow Management Methods

Bae-Seon Park · Seon-Young Kang · Hak-Tae Lee
Department of Aerospace Engineering, Inha University

Abstract

This paper explains a First-Come First-Served (FCFS) scheduling algorithm and compares the formulation with Bertsimas Stock Paterson (BSP) algorithm that is based on 0-1 integer programing (IP). The paper presents how the objective functions and constraints can be translated between the two algorithms to facilitate future comparison study between scheduling algorithms.

I. 서 론

항공기 출발 및 도착 스케줄링 알고리즘의 목적은 항공기 지연 시간을 줄이고, 운용하는 항공기 수를 증가시켜 효율을 최대화하는 것에 있다. 그에 따라 현재 다양한 알고리즘이 연구 및 개발되고 있으며, 대표적으로 선착순 방식의 FCFS 알고리즘과 최적화 기반 알고리즘 등이 있다.

FCFS 알고리즘의 경우 노드-링크 구조 (node-link sturcture)에 적용한 연구가 활발히 이루어지고 있으며 [1], [2], 구조가 간단하여 다양한 분야에서 적용되고 있다.

최적화 기반 알고리즘으로는 BSP 알고리즘이 있다. Bertsimas와 Stock Patterson은 IP를 이용해 항공기 경로 정보를 0과 1로 단순화하여 항공기 지연으로 인한 비용을 계산하였다 [3]. 또한 유전 알고리즘 등 다른 알고리즘과의 비교 연구도 선행된 바 있다 [4].

본 논문에서는 FCFS 알고리즘과 BSP 알고리즘 간 변화 기법을 설명하였다. FCFS와 BSP의 필요 변수 및 제약 조건을 비교하고, 각 알고리즘의 시뮬레이션 결과 예시를 통해 장단점을 분석하였다.

II. 스케줄링 알고리즘

1. FCFS 알고리즘

FCFS 알고리즘에 필요한 데이터는 공항 및 공역 데이터, 그리고 항공기 스케줄이다 [1]. 가장 중요한 변수는 항공기가 통과하는 공항과 공역에 대한 '진입 시각 (entry time)'과 '진출 시각(exit time)'이다. 출발 공항 – 공역1 – 공역2 – ... – 도착 공항의 형태로 나타나는 경로는 노드-링크 형태를 가지며 각 노드 또는 링크에 대하여 진입이 가능한 시간 구간은 '인터벌 (intervals)' 또는 슬롯 (slots)이라 한다. 그림 1은 항공기 1대의 스케줄을 노드-링크 구조로 나타낸 것이다.

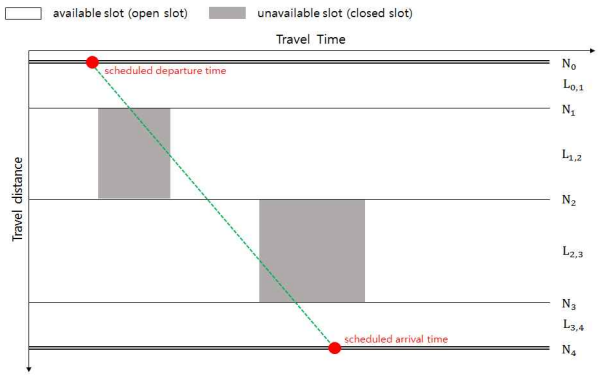

그림 1. 노드-링크 구조로 나타낸 항공기 1대의 스케줄

FCFS 알고리즘은 스케줄 내 항공기들에 대하여 우선순위를 부여한 후, 그 우선순위대로 각 항공기의 스케줄을 정한다. '출발 예정 시각 (scheduled departure time)' 순서대로 우선순위를 설정하는 것이 가장 일반적이다. 공항과 공역이 어느 시간대에 포화 상태가 되면 그 시간대를 닫힌 슬롯 (closed slots)으로 갱신하고, 다시 포화 상태가 끝나면 열린 슬롯 (opened slots)으로 갱신한다.

이 과정을 반복해 도착 공항에 도달하면 '최초 도착 시각 (earliest arrival time)'을 구할 수 있고, 이로부터 계산을 역으로 수행하면 '최초 출발 시각 (earliest departure time)'을 구할 수 있다. 이 때, 최초 도착 시각과 최초 출발 시각을 중심으로 한 각 공항의 Airport Departure Rate (ADR)과 Airport

Arrival Rate (AAR) 간격을 사용할 수 없는 슬롯으로 설정한 뒤, 다음 항공기 스케줄 계산을 반복해 최종적인 스케줄을 생성한다. 그림 2는 항공기 1대 스케줄의 알고리즘을 계산한 결과이다.

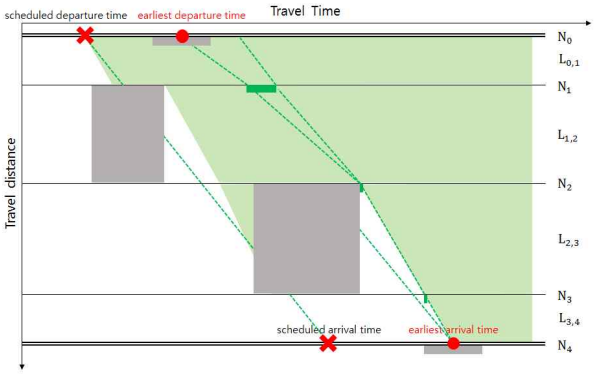

그림 2. FCFS 알고리즘 수행이 끝난 항공기 1대의 스케줄

2. BSP 알고리즘

BSP 알고리즘의 목적은 항공기 지연으로 인해 발생하는 비용을 최소화하는 것이며, 이는 식 (1)과 같다 [3], [4].

$$\text{Min}\sum_{f}[c_f^g g_f + c_f^a a_f] \qquad (1)$$

Flight 데이터의 집합 $\mathrm{F}=\{1,...,F\}$, 공항 데이터의 집합 $\mathrm{K}=\{1,...,K\}$, 이동 시간 (time period) 데이터의 집합 $\mathrm{T}=\{1,...,T\}$, 연속된 flight pair들의 집합 $\mathrm{C}=\{(f',f)\}$가 있다고 가정한다. 여기서 f는 f' 이후의 flight이며, 'flight'는 두 공항 사이의 'flight leg', 즉 비행 구간을 뜻한다. 식 (1)의 목적함수를 해결하기 위해 필요한 데이터는 다음과 같다.

N_f = flight f의 경로 상에 존재하는 공역들의 수

$$P(f,i)=\begin{cases}\text{출발 공항}\,(i=1)\\ \text{flight } f\text{의 경로 내 } i-1\text{번째 공역}\,(1<i<N_f)\\ \text{도착 공항}\,(i=N_f)\end{cases}$$

$P_f=(P(f,i):1\le i\le N_f)$

$D_k(t)$ = 시간 t일 때 공항 k의 departure capacity
$A_k(t)$ = 시간 t일 때 공항 k의 arrival capacity
$S_j(t)$ = 시간 t일 때 공역 j의 capacity

d_f = flight f의 출발 예정 시각
r_f = flight f의 도착 예정 시각
s_f = flight f의 turnaround time

c_f^g = 단위 시간 동안 flight f가 지상에서 대기할 때의 비용
c_f^a = 단위 시간 동안 flight f가 공중에서 대기할 때의 비용

g_f = flight f의 지상에서의 총 지연 시간
a_f = flight f의 공중에서의 총 지연 시간

l_{fj} = flight f가 공역 j에서 반드시 머물러야 하는 시간

T_f^j = flight f가 공역 j에 도달할 수 있는 이동 시간의 집합
$= \left\{\underline{T_f^j}, \overline{T_f^j}\right\}$
$\underline{T_f^j}$ = 집합 T_f^j의 첫 번째 이동 시간
$\overline{T_f^j}$ = 집합 T_f^j의 마지막 이동 시간 (2)

제약 조건과 이를 결정하는 결정 변수 w_{ft}^j는 다음과 같이 표현할 수 있다.

$$\sum_{f:P(f,1)=k}(w_{ft}^k - w_{f,t-1}^k) \le D_k(t) \quad \forall k\in\text{Airports}, t\in\text{Time}, \qquad (3)-1$$

$$\sum_{f:P(f,\text{last})=k}(w_{ft}^k - w_{f,t-1}^k) \le A_k(t) \quad \forall k\in\text{Airports}, t\in\text{Time}, \qquad (3)-2$$

$$\sum_{\substack{f:P(f,i)=j,\\ P(f,i+1)=j'}}(w_{ft}^j - w_{ft}^{j'}) \le S_j(t) \quad \forall j\in\text{Sectors}, t\in\text{Time}, \qquad (3)-3$$

$$w_{f,t-1_{fj}}^{j'} - w_{ft}^j \le 0 \quad \begin{cases}\forall f\in\mathrm{F}, t\in T_f^j, j=P(f,i),\\ j'=P(f,i+1), i<N_f,\end{cases} \qquad (3)-4$$

$$w_{f,t}^k - w_{f,t-s_f}^k \le 0 \quad \begin{cases}\forall (f',f)\in\mathrm{C}, t\in T_f^j,\\ k=P(f,1)=P(f',N_f),\end{cases} \qquad (3)-5$$

$$w_{f,t}^j - w_{f,t-1}^j \ge 0 \quad \forall f\in\mathrm{F}, j\in P_f, t\in T_f^j. \qquad (3)-6$$

$$w_{ft}^j=\begin{cases}1,\ \text{flight } f\text{가 시간 } t\text{에 공역 } j\text{에 도달했을 경우}\\ 0,\ \text{이를 제외한 모든 경우}\end{cases} \qquad (4)$$

이를 이용해 식 (1)을 치환하면 Traffic Flow Management (TFM)의 최종 공식을 도출해낼 수 있다.

$$w_{ft}^j = w_{ft}^j - w_{f,t-1}^j$$
$$g_f = \sum_{t\in T_f^k, k\in P(f,1)} t(w_{ft}^k - w_{f,t-1}^k) - d_f$$
$$a_f = \sum_{t\in T_f^k, k\in P(f,N_f)} t(w_{ft}^k - w_{f,t-1}^k) - r_f - g_f$$

$$IZ_{TFMP} = \text{Min}\sum_{f\in\mathrm{F}}\left[\begin{array}{l}(c_f^g - c_f^a)\sum_{t\in T_f^k, k\in P(f,1)} t(w_{ft}^k - w_{f,t-1}^k)\\ +c_f^a\sum_{t\in T_f^k, k\in P(f,N_f)} t(w_{ft}^k - w_{f,t-1}^k)\\ +(c_f^a - c_f^g)d_f - c_f^a r_f\end{array}\right] \qquad (5)$$

예를 들어, flight 1과 flight 2가 그림 3과 같이 $P_1=\{1,\mathrm{A},\mathrm{C},\mathrm{D},\mathrm{E},4\}$, $P_2=\{2,\mathrm{F},\mathrm{E},\mathrm{D},\mathrm{B},3\}$의 경로를 따라 비행할 때, 시간 t에 대한 결정 변수 w_{ft}^j는 식 (6)와 같이 0과 1로 나타낼 수 있으며, 이를 목적 함수에 대입해 비용을 계산한다.

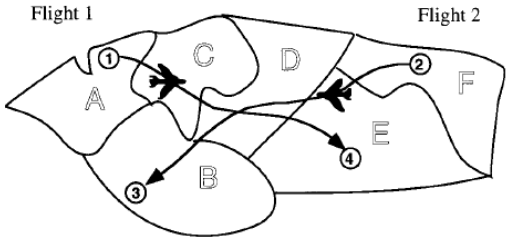

그림 3. 주어진 경로를 비행하는 2대의 항공기

$$\text{flight 1}\begin{cases} w_{1,t}^{1}=1, \\ w_{1,t}^{A}=1, \\ w_{1,t}^{C}=1, \\ w_{1,t}^{D}=0, \\ w_{1,t}^{E}=0, \\ w_{1,t}^{4}=0. \end{cases} \quad \text{flight 2}\begin{cases} w_{2,t}^{2}=1, \\ w_{2,t}^{F}=1, \\ w_{2,t}^{E}=1, \\ w_{2,t}^{D}=0, \\ w_{2,t}^{B}=0, \\ w_{2,t}^{3}=0. \end{cases} \tag{5}$$

III. 알고리즘 비교

1. 변수 및 제약 조건 비교

FCFS 알고리즘과 BSP 알고리즘의 주요 변수와 제약 조건을 표 1, 2에 정리하였다. * 표시는 동일한 의미로 사용되는 경우이다.

표 1. 두 알고리즘의 변수 비교

#	FCFS	BSP
1*	출발 예정 시각	d_f
2*	도착 예정 시각	r_f
3	$c_f^g = c_f^a$	$c_f^g,\ c_f^a$
4	$a_f + g_f$	$a_f,\ g_f$
5*	maximum speed-up transit time	l_{fj}
6	feasible transit times	$T_f^j = \left\{\underline{T_f^j}, \overline{T_f^j}\right\}$

FCFS 알고리즘은 항공기가 대기할 때 발생하는 비용을 계산할 때 지상과 공중을 구분하지 않는다. 따라서 c_f^g와 c_f^a를 동일한 것으로 보며, 항공기 지연 시간 역시 지상, 공중 구분 없이 총 지연 시간으로 계산한다.

'Transit time'은 FCFS 알고리즘에서 항공기가 링크 구간을 이동하는 시간, 즉, 공역 이동 시간이다. 기준 이동 시간 (nominal transit time)에 항공기 속도 변화율 (speed rate)을 적용하면 이동 시간은 범위를 가지게 된다. 이 때 가장 빠른 시간인 'maximum speed-up transit time'은 항공기 f가 공역 j에서 소요하는 최소 시간 l_{fj}와 동일하며, 그 이동 시간 범위는 T_f^j와 같다.

표 2. 두 알고리즘의 제약 조건 비교

#	FCFS	BSP
1	feasible transit times	$T_f^j = \left\{\underline{T_f^j}, \overline{T_f^j}\right\}$
2	ADR	$D_k(t)/\triangle t$
3	AAR	$A_k(t)/\triangle t$
4*	시간별 공역 capacity	$S_j(t)$

BSP 알고리즘은 공역 이동 시간들의 집합인 T_f^j를 변수로 이용하지만, FCFS 알고리즘은 이를 항공기 이동 속도의 제약 조건으로 적용한다.

ADR과 AAR은 공항의 시간 당 이, 착륙하는 항공기 대수이다. $D_k(t)$와 $A_k(t)$는 시간 t일 때 공항의 departure capacity와 arrival capacity이고, $\triangle t$는 BSP 알고리즘에서 설정한 단위 시간을 뜻한다. 따라서 $D_k(t)$와 $A_k(t)$를 단위 시간 $\triangle t$로 나눠 단위를 동일하게 할 경우 그 의미는 ADR, AAR과 같다.

FCFS 알고리즘에서는 항공기 1대의 계산이 끝난 후, 최초 출발, 도착 시각을 기준으로 $\pm dt = \frac{1hour}{ADR \text{ or } AAR}$ 만큼의 시간을 닫힌 슬롯으로 갱신한다. 이는 전체 스케줄링이 끝난 후 계산된 ADR, AAR이 초기에 주어진 제약 조건인 ADR, AAR보다 클 수 없기 때문이다. 공역 capacity 역시 이와 같으며, 스케줄링이 끝난 후 모든 공역 내 항공기 수는 주어진 공역의 최대 capacity보다 클 수 없다.

BSP 알고리즘의 제약 조건인 식 (3) 역시 위와 같은 의미이며, 따라서 두 알고리즘의 제약 조건은 본질적으로 동일하다고 볼 수 있다.

FCFS 알고리즘에서 i번째 항공기의 제약 조건은 그 전에 계산된 $i-1$번째까지의 모든 항공기들이 제약 조건으로 누적된다. 이는 곧 항공기들이 처음 스케줄에 어떻게 배치되었느냐가 결과에 큰 영향을 준다고 볼 수 있으며, 따라서 FCFS 알고리즘은 전체적인 스케줄의 최적화가 아닌, 각 항공기에 주어진 제약 조건 내에서 항공기별 지연을 최소화하는 부분적 최적화 알고리즘이다.

2. 시뮬레이션 예시

각 알고리즘 별로 수행한 시뮬레이션 결과 예시를 바탕으로 그 장단점을 분석하였다 [1], [3].

그림 4는 FCFS 알고리즘을 이용해 28시간 동안 48000여대의 항공기 시뮬레이션을 수행한 것으로, 출발 지연 감소 시간을 그래프로 나타낸 것이다. C++로 구성한 시뮬레이션을 수행하는 데 소요된 시간은 약 1분이다 [1].

표 2, 3는 [5]의 Ground-Holding Problem에 사용된 데이터로 시뮬레이션을 수행한 결과이다 [3]. 항공기 수가 증가하고 capacity가 다양할수록 object value와 계산 시간이 크게 증가하는 것을 확인할 수 있다.

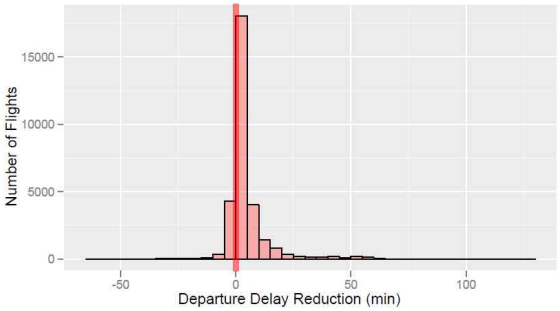

그림 4. FCFS 시뮬레이션 결과 예시

표 2. BSP 알고리즘을 이용한 항공기 1000대 시뮬레이션 수행 결과 예시

$\|\mathcal{F}\|$	$\|\mathcal{C}\|/\|\mathcal{F}\|$	Dep. Capacity	Arr. Capacity	Obj. Value	Time	% Nonint.
1,000	0.20	32	17	50,750	342	0
1,000	0.20	32	16	55,450	227	0
1,000	0.20	32	15	63,525	262	0
1,000	0.20	32	14	inf	—	—
1,000	0.40	18	12	47,000	290	0
1,000	0.40	18	10	79,916	521	2.2
1,000	0.40	17	10	88,241	741	4
1,000	0.40	16	10	inf	—	—
1,000	0.60	20	18	22,316	369	0
1,000	0.60	20	15	33,292	376	0
1,000	0.60	20	14	39,266	359	0
1,000	0.60	20	13	inf	—	—
1,000	0.80	30	30	17,000	183	0
1,000	0.80	20	20	28,250	283	0
1,000	0.80	19	19	inf	—	—

표 3. BSP 알고리즘을 이용한 항공기 3000대 시뮬레이션 수행 결과 예시

$\|\mathcal{F}\|$	$\|\mathcal{C}\|/\|\mathcal{F}\|$	Dep. Capacity	Arr. Capacity	Obj. Value	Time	% Nonint.
3,000	0.20	30	30	42,000	4,537	0
3,000	0.20	20	20	228,000	5,475	0
3,000	0.20	19	19	inf	—	—
3,000	0.40	30	30	42,000	5,062	0
3,000	0.40	20	20	234,000	4,703	0
3,000	0.40	19	19	inf	—	—
3,000	0.60	30	30	42,000	5,629	0
3,000	0.60	20	20	234,000	5,407	0
3,000	0.60	19	19	inf	—	—
3,000	0.80	30	30	42,000	6,021	0
3,000	0.80	20	20	252,000	9,411	0
3,000	0.80	19	19	inf	—	—

FCFS 알고리즘의 가장 큰 장점은 선착순 방식으로 인한 '단순함'이다. 또한 노드-링크 구조로 표현할 수 있는 시스템이라면 바로 적용이 가능하며, 계산이 매우 빠르고 많은 양의 데이터를 시뮬레이션할 수 있다. 그러나 부분적 최적화로 인해 그 결과가 전체적으로 좋은 결과라는 것을 장담할 수 없다.

BSP 알고리즘은 단위 시간 $\triangle t$를 작게 설정할수록 더 정확한 계산이 가능하다. 다양한 기준에서 시뮬레이션을 수행할 수 있으며, 스케줄의 전체적인 최적화를 수행하기 때문에 FCFS 알고리즘보다 좋은 결과를 도출할 수 있다. 그러나 데이터의 양이 많고 단위 시간을 작게 설정할수록 계산 시간이 급격히 증가하는 단점이 존재한다.

Ⅳ. 결 론

본 논문에서는 항공교통흐름 제어에 사용되는 다양한 알고리즘 중 FCFS 알고리즘과 BSP 알고리즘을 비교할 수 있도록 목적 함수와 제약조건들이 상호 어떻게 변환될 수 있는지를 분석하였다. FCFS 알고리즘의 경우 부분적 최적화 방식으로 구조가 간단하여 많은 양의 데이터를 빠르게 계산할 수 있으며, BSP 알고리즘은 비교적 정확한 결과를 도출하지만 단위 시간이 작고 데이터가 많아질수록 계산 시간이 크게 증가하는 것을 각 시뮬레이션 결과 예시를 통해 확인하였다.

후 기

본 연구는 인하대학교의 '공항과 공역을 통합한 항공 흐름 제어에 대한 연구 (과제번호 48590)'에 의해 수행되었습니다.

참고 문헌

[1] C. Park, H. T. Lee, and L. Meyn, "Computing Flight Departure Times Using an Advanced First-Come First-Served Scheduler," 12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2012.

[2] L. Meyn, "A Closed-Form Solution to Multi- Point Scheduling Problems," AIAA Modeling and Simulation Technologies Conference, Toronto, Ontario, Aug. 2-5, 2010.

[3] D. Bertsimas and S. S. Patterson, "The Air Traffic Flow Management Problem with Enroute Capacities," Operations Research, Vol. 46, No. 3, pp.406~422, May-June 1998.

[4] J. Rios and J. Lohn, "A Comparison of Optimizaion Approaches for Nationwide Traffic Flow Management," Proceedings of the AIAA Guidance, Navigation, and Control (GNC) Conference, Chicago, Illinois. 2009.

[5] P. Vranas, D. Bertsimas and A. R. Odoni, "The multi-airport ground-holding problem in air traffic control," Operations Research 42.2 (1994): 249-261.