항공기 지상 이동 Fast-Time 시뮬레이터 개발

김 태 영 · 박 배 선 · 이 현 웅 · 이 학 태

인하대학교 항공우주공학과

Development of Fast-Time Simulator for Surface Operation

Tae Young Kim · Bae-Seon Park · Hyeonwoong Lee · Hak-Tae Lee

Department of Aerospace Engineering, Inha University

Abstract

This study presents the development of a fast-time airport surface simulator. The simulator uses the output from a scheduler to move the aircraft on the surface while implementing conflict detection and resolution algorithms to maintain separation and prevent deadlock. The simulator was tested with an scheduling result at the Incheon International Airport that contains 60 aircraft. Various conflict situations are identified and prioritization strategies are compared.

I. 서론

매년 항공 교통량 크게 증가함에 따라 공항 내에서 항공기의 움직임이 복잡해지고, 출도착 지연이 빈번하게 발생하고 있다. 이러한 문제를 해결하기 위한 방안으로 출도착 통합 관리 시스템에 대한 연구[1,2]와, 이에 필요한 효율적인 항공기 스케줄링 알고리즘에 대한 연구 개발이 진행되고 있다 [2,3].

출도착 관리 시스템의 연구 개발 과정에서 항공기 지상 이동 시뮬레이터가 필수적이다. 일반적으로 스케줄링 알고리즘은 상대적으로 단순화된 지상의 모델을 이용해서 스케줄링 결과를 도출하게 되고, 이 결과를 지상 이동 시뮬레이션을 통해 검증하게 된다. 또한 지상 이동 시뮬레이터를 통해 항공기의 지상 궤적을 예측하고 이를 다시 스케줄링에 반영할 수도 있다. 미국 NASA에서는 Spot And Runway Departure Advisor (SARDA)라는 출발 관리 시스템을 개발하여[1] 미국 내의 몇몇 주요 공항에서 테스트를 진행 중에 있으며, 이를 위해 필요한 fast-time 시뮬레이터인 Surface Operations Simulator and Scheduler (SOSS)[4]를 개발하여 사용하고 있다.

본 논문에서는 기존의 First-Come First-Served (FCFS) 스케줄러[3]와 연동이 가능한 fast-time 시뮬레이터에 대해 설명한다. 이 중에서 특히 분리 유지 알고리즘에 중점을 두어, 같은 유도로 내에서의 전후 분리 간격 유지, 다수의 항공기가 교차로에 진입하는 경우, 교착 상태를 방지하고 순차적으로 항공기 이동을 관리하는 알고리즘을 포함하도 있다.

60대의 항공기에 대해 FCFS 알고리즘을 이용해 스케줄링 한 결과를 바탕으로 fast-time 시뮬레이터의 작동을 검증하였으며, 3가지 교차로 처리 방식에 대한 결과를 비교하였다.

II 장에서는 전반적인 시뮬레이터의 구성에 대하여 설명하고, III 장에서 분리 유지 알고리즘을 설명한다. IV 장에서 테스트 결과를 종합하고, V 장에서 결론 및 향후 계획을 제시하였다.

II. 시뮬레이터 구성

1. 항공기 이동 모델

매순간의 항공기의 위치를 업데이트하기 위해 각 항공기에 저장된 노드 데이터와 노드에서의 시간 정보를 사용한다. 항공기의 위치를 계산하기 위해 내분점 공식을 사용하였다.

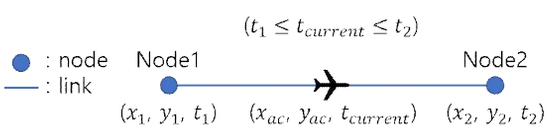

그림 1. 내분점 공식을 이용한 항공기의 위치 및 진행률 계산

$$x_{ac} = (x_2 - x_1) \times \frac{t_{current} - t_1}{t_2 - t_1} + x_1 \quad (1)$$

$$y_{ac} = (y_2 - y_1) \times \frac{t_{current} - t_1}{t_2 - t_1} + y_1 \quad (2)$$

각 링크의 진출입 시간 정보가 있으므로 현재 위치한 링크의 노드 좌표를 알 수 있으며, 이는 그림 1에서 Node1과 Node2로 표시되어 있다. 항공기가 동일 링크에서는 등속으로 이동한다는 가정 하에 식 (1)과 (2)를 사용하여 현재 시간으로부터 현재 위치를 알 수 있다.

2. 입력 데이터

시뮬레이터의 구동을 위해 노드-링크 모델과, 스케줄 파일이 필요하다. 노드 모델은 타입과 ID, 위, 경도 좌표로 이루어져 있으며, 링크 모델은 타입과 ID, 링크를 구성하는 두 노드의 노드 명으로 이루어져 있다.

* Links in RKSI			
* Type	ID	Node1	Node2
GATE	gate1	20000	20186
GATE	gate2	20001	20186
GATE	gate3	20002	20189
RAMP	Rp1	20186	20189
RAMP	Rp2	20189	10195
RAMP	Rp3	10195	10196
TAXI	Tx1	20591	10105
TAXI	Tx2	10105	10106
TAXI	Tx3	10106	20600
RWY15L/33R	Rwy1	20580	20610
RWY15L/33R	Rwy2	20610	20590
RWY15L/33R	Rwy3	20590	20591

(a) 노드 모델 파일 형식

* Links in RKSI			
* Type	ID	Node1	Node2
GATE	gate1	20000	20186
GATE	gate2	20001	20186
GATE	gate3	20002	20189
RAMP	Rp1	20186	20189
RAMP	Rp2	20189	10195
RAMP	Rp3	10195	10196
TAXI	Tx1	20591	10105
TAXI	Tx2	10105	10106
TAXI	Tx3	10106	20600
RWY15L/33R	Rwy1	20580	20610
RWY15L/33R	Rwy2	20610	20590
RWY15L/33R	Rwy3	20590	20591

(b) 링크 모델 파일 형식

그림 2. 노드-링크 모델 예시

스케줄 파일은 항공기의 편 명, 시작과 도착 노드 명, 경유한 링크 명, 각 링크의 진출입 시간 등으로 구성되어 있다. 시뮬레이션을 위해 항공기를 생성한 후 경유한 노드와 각 노드에서의 시간을 저장한다.

* Flight ID	Address	Type	Wake Category	State	Entry Time	Exit Time	Previous Link	Current Link	Next Link	Speed-Up	Slow-Down
KAL854	HL8001	A333	H	ARR	1980	1990	XXXX	33R	Rwy7	0	0
KAL854	HL8001	A333	H	ARR	1990	1997	Rwy7	Rwy6	Rwy5	0	0
KAL854	HL8001	A333	H	ARR	1997	2004	Rwy6	Rwy5	Rwy4	0	0
KAL854	HL8001	A333	H	ARR	2004	2049	Rwy5	Rwy4	Tx15	0	0
KAL854	HL8001	A333	H	ARR	2049	2097	Rwy4	Tx15	Tx23	0	0

그림 3. 스케줄 파일 예시

3. User Interface 화면 구성

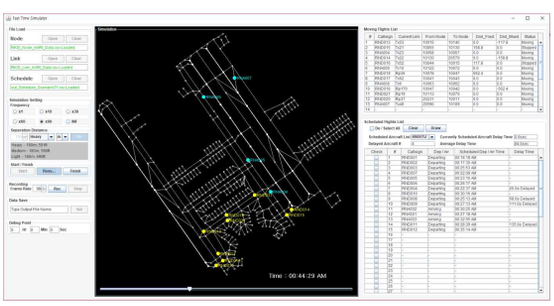

그림 4. 실행중인 시뮬레이터 화면

본 프로그램은 5개의 모듈로 구성되어 있는 설정 모듈과 이동 중인 항공기의 정보 확인을 위한 2가지 모듈로 구성된 시뮬레이션 모듈, 결과를 확인하기 위한 1개의 모듈, 총 8개의 모듈로 구성되어 있다.

설정 모듈에는 파일 입력 모듈과 배속 설정 모듈, 분리 간격 설정 모듈, 시뮬레이션 시작/종료 모듈, 항공기 통행 우선권 선택 모듈이 있다. 항공기의 wake turbulence category에 따라 분리 간격을 다르게 적용할 수 있다. 분리 간격 설정 모듈의 기능이 비활성화 되어 있는 경우, 입력된 스케줄 파일을 그대로 시뮬레이션을 수행할 수 있다. 시뮬레이션 모듈에서는 이동 중인 항공기의 위치를 가시화 하고, 항공기 목록에서 진행상황을 보여준다. 충돌 위험이 있는 항공기들은 위험도에 따라 색깔이 변하도록 되어있다.

III. 항공기 분리 유지 알고리즘

항공기의 위치 파악은 점 질량 모델이라는 가정 하에 이루어진다. 지상 이동 시 항공기간의 충돌 방지를 위한 분리 간격은 항공기 무게 중심을 기준으로 계산한다.

두 항공기의 충돌 위험이 발생하는 경우는 앞, 뒤 간격 미준수와 교차로에 다수의 항공기가 접근하는 경우가 있다. 항공기의 충돌 위험이 있는 4 가지 상황과, 4 가지 정지 및 재출발 알고리즘을 적용하였다.

1. Case 1: 동일 선상

두 항공기의 진행 방향이 일치하여 명확한 앞, 뒤 구분이 가능한 경우이며, 두 대의 항공기가 동일 링크에서 진행 중인 경우와, 이웃한 두 링크에 각각 한 대의 항공기가 진행 중인 경우가 있다.

i) 동일 링크

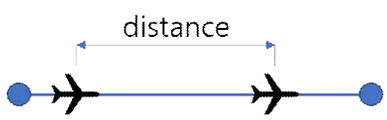

그림 5. 동일 링크 내에서 거리 계산

그림 5는 두 항공기가 동일 링크 내에서 이동하며 진행 방향이 일치하는 경우에서 분리 거리를 보여준다. 두 항공기의 거리가 분리 간격보다 가까운 경우, 후행 항공기를 정지시키고, 거리가 멀어져 분리 간격이 준수될 때, 후행 항공기가 재출발한다.

ii) 이웃한 링크

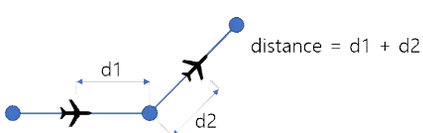

그림 6. 이웃한 링크에서 거리 계산

그림 6은 두 항공기가 진행 중인 링크가 서로 다르지만 한 항공기의 출발 노드와, 다른 항공기의 목표 노드가 일치하여 두 항공기가 동일 선상에 있다고 볼 수 있는 경우에 분리 거리를 보여준다. 두 항공기의 거리가 분리 간격보다 가까워 진 경우, 후행 항공기를 정지시키고, 분리 간격이 준수될 때, 후행 항공

기를 재출발시킨다.

2. Case 2: 교차로 접근

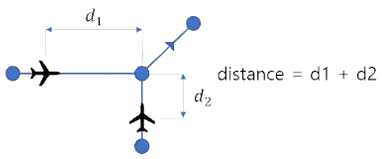

그림 7. 동일 노드로 접근하는 경우

그림 7은 두 항공기가 서로 다른 링크에서 진행 중이며 동일 노드로 향하는 경우를 보여주며 이 때 두 항공기 사이의 거리는 노드까지 거리의 합으로 계산하였다. 선행과 후행 항공기를 명확하게 구별하기 어려운 경우가 발생하면 사용자가 선택할 수 있는 3가지의 기준을 제공한다.

i) 남은 거리 기준

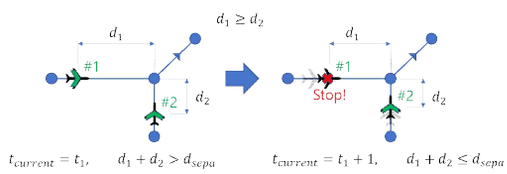

그림 8. 남은 거리로 선행 항공기 선정

그림 8은 두 항공기의 거리가 분리 간격보다 가까워진 경우 선행 항공기를 선정하는 방식을 보여주며 노드로부터 거리가 가까운 2번 항공기에 우선권을 부여한다. 선행 항공기가 노드를 빠져나간 후, 두 항공기의 거리가 분리 간격을 준수하는 경우 정지했던 항공기를 재출발시킨다.

ii) 남은 시간 기준

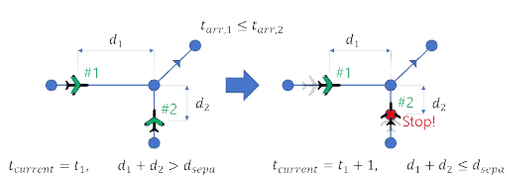

그림 9. 노드 도착 예정 시간으로 선행 항공기 선정

그림 9는 두 항공기간의 분리 간격이 준수되지 않았을 경우 노드 도착 예정 시간이 이른 1번 항공기에 우선권을 부여한다. 선행 항공기가 노드를 빠져나간 후 분리 간격이 준수될 때 후행 항공기를 재출발시킨다.

iii) 최소 지연 시간 기준

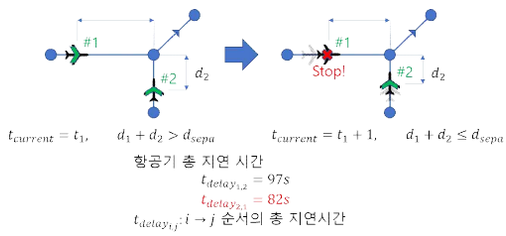

그림 10. 최소 지연 시간으로 선행 항공기 선정

그림 10은 교차로에 진입한 항공기들에 우선권을 부여할 수 있는 모든 경우의 수에 대해 총 지연 시간을 계산한 후, 지연 시간이 가장 짧은 경우의 순서대로 교차로를 통과시키도록 하는 것을 볼 수 있다. 그림 10에서 1번 항공기가 먼저 통과할 때 총 지연시간이 97초이며, 2번 항공기가 먼저 통과할 때 총 지연시간이 82초로 계산된 경우, 2번 항공기를 선행 항공기로 지정하여 통과하도록 한다. 항공기의 속도가 거의 비슷한 경우, 남은 거리 기준을 적용하였을 때와 비슷한 순서로 통과시키는 것을 확인하였다.

3. Case 3: 사용 중인 다음 링크

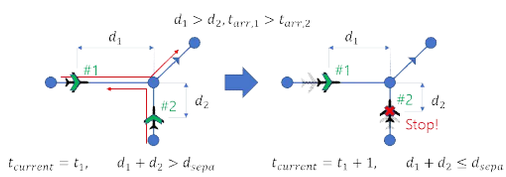

그림 11. 다음 링크를 고려하여 선행 항공기 선정

그림 11은 두 항공기가 동일 노드로 향하지만, 2번 항공기의 진행 예정 링크에 1번 항공기가 마주보는 방향으로 접근하는 경우에서 선행 항공기를 선정하는 방식으로 앞, 뒤 항공기가 명확하게 구별된다. Case 2에서 언급한 방법을 적용하여 2번 항공기가 선행 항공기로 지정된 경우 동일 링크에 반대방향의 두 항공기가 놓이는 상황이 발생한다. 이러한 상황을 방지하기 위해 1번 항공기를 선행 항공기로 지정하고, 노드를 빠져나간 후 분리 간격이 유지되면 2번 항공기를 재출발시킨다.

4. Case 4: 활주로 양보

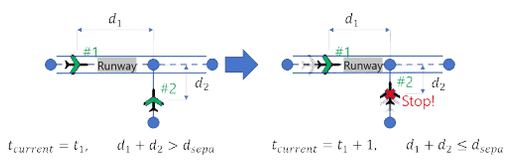

그림 12. 활주로에서 이동 중인 항공기가 선행 항공기

그림 12에서와 같이 이착륙중인 항공기는 우선권을 부여받고 선행 항공기로 지정되어 먼저 노드를 통과하게 된다.

5. Case 5: 마주보며 접근

항공기가 마주보는 방향으로 동일 링크에 진입하는 상황은 존재해서는 안 된다. 스케줄링 알고리즘에서 이를 고려하여 스케줄을 계산하기 때문에 현재 fast-time 시뮬레이터에는 이 경우를 감안하여 정지, 재출발 하는 기능은 구현되지 않은 상태이다. 두 항공기가 마주 오는 경우에는 둘 다 그대로 진행한다.

Ⅳ. 결과

시뮬레이션을 수행한 테스트 케이스는 인천국제공항에서 약 1시간 10분 동안 총 60대의 항공기가 이착륙하는 스케줄이며, 분리 간격은 180m로 설정 하였다.

항공기 정지 기능을 비 활성화하여 동일한 스케줄로 시뮬레이션 한 결과 후행 항공기의 분리 간격이 유지되지 못하는 횟수가 59회 발생하였으며, 교차로로 다수의 항공기가 접근하는 경우도 14회 발생하였다. 항공기가 마주 오는 상황이 2회 발생하였으며, 모두 게이트에서 빠져나오는 항공기와 진입하는 항공기가 마주 오는 상황이었다. 실제 운용에 있어서 계류장은 움직임이 보다 자유롭기 때문에 실제 상황에서는 우회가 가능할 것으로 예상되나, 추후에는 시뮬레이터에서도 이 부분을 처리할 수 있는 기능을 추가할 예정이다. 5가지 경우에 대한 발생 횟수는 표 1에 정리되어 있다.

항공기 접근 케이스	횟수
Case 1: 동일 선상 (동일 링크)	18
Case 1: 동일 선상 (이웃 링크)	41
Case 2: 교차로 접근	14
Case 3: 사용 중인 다음링크	5
Case 4: 활주로 양보	1
Case 5: 마주보며 접근	2

표 1. 항공기 정지 기능 비활성화 시 분리 거리 유지 실패 횟수

분리유지 기능을 사용하였을 경우, 3 가지 정지 알고리즘에 따라 지연 시간의 차이가 발생함을 확인하였다. 표 2에 정리된 결과는 오히려 최소 지연 시간 기준을 적용했을 경우, 전체 지연이 더 크게 됨을 볼 수 있는데, 이는 각 노드에서의 지연을 최소화 하는 것이 전체 시스템의 지연을 최소화 하지 못할 수도 있음을 보여준다.

정지 알고리즘	지연 항공기 대수	평균 지연 시간 (sec)
남은 거리 기준	28	242
남은 시간 기준	25	224
최소 지연 시간 기준	24	269

표 2. 정지 알고리즘 별 지연 항공기 대수 및 평균 지연 시간

Ⅴ. 결론

본 연구에서는 공항에서의 항공기 지상 이동을 모사할 수 있는 fast-time 시뮬레이터의 개발에 대하여 기술하고, 사용된 분리유지 알고리즘을 설명하였다. 개발된 시뮬레이터는 인천국제공항에서의 60대 항공기의 출도착 스케줄을 이용하여 테스트 되었으며 우선순위 설정 방식에 따라 다른 결과가 나타남을 확인하였다.

현재 개발된 항공기 모델은 계단 함수 형태의 정지와 가속이 가능하며 정해진 경로를 따라 이동하는 1차원 점 질량 모델이다[5]. 하지만 실제에는 항공기가 링크의 중심축을 기준으로 좌우로 움직이며 2차원 기동을 한다. 추후 2차원 점 질량 항공기 모델을 개발, 적용하여 보다 현실적인 시뮬레이션을 통한 검증을 수행할 계획이다.

참고 문헌

[1] Jung, Y., Malik, W., Tobias, L., Gupta, G., Hoang, T., and Hayashi, M., "Performance Evaluation of SARDA: An Individual Aircraft-based Advisory Concept for Surface Management," Air Traffic Control Quarterly, Vol. 22, Number 3, 2015, p. 195-221, April 2015.

[2] Yeonju Eun, Daekeun Jeon, Hanbong Lee, Yoon C. Jung, Zhifan Zhu, Myeongsook Jeong, Hyounkyong Kim, Eunmi Oh, and Sungkwon Hong. "Optimization of Airport Surface Traffic: A Case-study of Incheon International Airport", 17th AIAA Aviation Technology, Integration, and Operations Conference, AIAA AVIATION Forum,

[3] B. S. Park, H. W. Lee, and H. T. Lee, “Extended First-Come First-Served Scheduler for Airport Surface Operation”, International Journal of Aeronautical and Space Sciences, Vol 19, Issue 2, pp 509-517

[4] Zhifan Zhu, “Surface Operations Simulator and Scheduler (SOSS) Presentation”, Joint Workshop for KAIA/KARI/IIAC-NASA Collaboration, April 5-7, 2016

[5] Di Wu, Yiyuan J. Zhao, and Brian Capozzi, “Fundamental Surface Trajectory Models for Air Traffic Automation”, 10th AIAA Aviation Technology, Integration, and Operation (ATIO) Conference