Journal of The Korea Society of Computer and Information
Vol. 19, No. 9, September 2014
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2014.19.9.161

항공기 착륙 문제의 다항시간 알고리즘

이 상 운*

A Polynomial Time Algorithm for Aircraft Landing Problem

Sang-Un Lee *

요 약

공항에 불규칙한 시간간격으로 접근하는 항공기들을 최소의 비용으로 착륙시키는 항공기 착륙 문제 (ALP)는 최적 해를 구하기 어려워 다양한 메타휴리스틱 방법들이 제안되고 있다. 본 논문에서는 ALP에 대해 $O(n\log n)$의 다항시간으로 최적 해를 구하는 휴리스틱 알고리즘을 제안한다. 제안된 알고리즘은 착륙 목표시간 오름차순으로 정렬시키고, 항공기들 간의 분리 시간과 착륙 비용을 고려하여 착륙순서를 변경시킨 최적화 과정을 수행하는 방법을 적용하였다. ALP에 대한 예제 데이터인 Airland1 ~ Airland8에 대해 소요비용이 0이 되는 활주로 개수 m까지 25개 데이터를 실험한 결과 모든 데이터에 대해 최적 해를 구하였다. 특히, Airland8의 $m=1$ 데이터에 대해서는 기존에 알려진 최적 해를 개선하였다.

▸ Keywords : 항공기 착륙 문제, 목표시간, 비용함수, 분리시간, 착륙순서 최적화

Abstract

The optimal solution of minimum cost for aircraft landing problem (ALP) is very difficult problem because the approached aircraft are random time interval. Therefore this problem has been applied by various metaheuristic methods. This paper suggests $O(n\log n)$ polynomial time heuristic algorithm to obtain the optimal solution for ALP. This algorithm sorts the target time of aircraft into ascending order. Then we apply the optimization of change the landing sequence take account of separation time and the cost of landing. For the Airland1 through Airland8 of benchmark data of ALP, we choose 25 data until the number of runway m that the total landing cost is 0. We can be obtain the optimal solution for all of the 25 data. Especially we can be improve the known optimal solution for $m=1$of Airland8.

▸ Keywords : Aircraft landing problem, Target time, Cost function, Separation time, Landing sequence optimization

•제1저자 : 이상운
•투 고 일 : 2014. 6. 17, 심사일 : 2014. 7. 6, 게재확정일 : 2014. 8. 24.
* 강릉원주대학교 멀티미디어공학과 (Dept. of Multimedia Eng., Gangneung-Wonju National University)

I. 서 론

공항의 활주로 처리용량 제한조건으로 인해 공항에 불규칙한 시간 간격으로 접근하는 항공기들을 소요비용이 0인 목표시간에 모든 항공기 들을 착륙시킬 수는 없다. 따라서 병목현상이 발생할 경우 관제사는 일부 항공기는 보다 일찍 도착하도록 지시하기도 하며, 일부 항공기는 공항 상공에서 선회비행으로 착륙을 대기하도록 지시하기도 한다. 다수의 항공기들을 대상으로 착륙에 따른 소요비용을 최소화하도록 착륙순서를 결정하는 문제를 항공기 착륙 문제 (aircraft landing problem, ALP)라 한다[1].

ALP에서는 목표시간보다 빠르거나 늦게 착륙시키는데 소요되는 단위시간 (분)당 비용이 소요되며, 한 대의 항공기가 착륙하면서 날개부분의 소용돌이 현상으로 다음 항공기는 바로 이어서 착륙하면 안전상 문제가 발생하여 분리시간 (separation time)이 적용된다[2].

ALP는 최소 착륙비용을 갖는 착륙순서를 결정하는 최적해를 찾기가 어려워 선형계획법 (linear programming, LP), 휴리스틱한 트리탐색 (tree search, TS), 뿐 아니라 다양한 메타휴리스틱 기법들이 적용되고 있다[3-10]. 이와 같이 다양한 메타휴리스틱 기법들이 적용되고 있음에도 불구하고 최적 해를 찾지 못하는 경우도 발생한다.

본 논문에서는 ALP의 주어진 항공기들을 선 목표시간-선착륙 (first-target first served, FTFS)으로 착륙시키기 위해 목표시간 오름차순으로 정렬하고, 분리시간과 목표시간과의 차이로 인해 발생하는 착륙비용을 고려하여 일부 항공기의 순서를 변경하는 최적화 방법을 적용하는 휴리스틱 방법을 제안한다. 2장에서는 ALP의 개념을 고찰해 본다. 3장에서는 ALP의 최적 해를 다항시간으로 찾을 수 있는 선목표시간-선착륙 최적화 (FTFSO)의 휴리스틱 알고리즘을 제안한다. 4장에서는 Beasley[11]의 OR-Library에서 인용된 Airland1 ~ Airland8의 8개 데이터에 대해 제안된 알고리즘의 적합성을 검증해 본다.

II. ALP 개념

ALP는 그림 1과 같이 순항비행으로 순차적으로 공항에 접근하는 항공기들을 공항 활주로의 착륙 능력 범위 내에서 모든 항공기를 최소의 비용으로 착륙시키는 것이 목적이다[12]. 이 때 최소 비용은 목표 시간 (target time, T_i)이며, 연속되는 항공기들의 착륙시간은 분리시간 (separation time, t_s) 간격으로 착륙시켜야 한다. 이는 항공기 착륙시 날개부분에서 발생하는 소용돌이로 인한 난기류의 영향을 후속 항공기가 받지 않도록 안전한 거리를 유지하기 위함이다[13]. 일반적으로, 공항에 접근하는 모든 항공기들 간의 거리 간격은 분리시간을 충족할 수 없기 때문에 특정 항공기들이 병목현상이 발생하기도 한다. 따라서, 일부 항공기는 예정된 시간보다 빨리 도착하여 착륙하도록 하고, 일부 항공기는 공항 상공에서 착륙을 대기하는 순회비행을 하도록 하기도 한다. 이와 같이 순항비행으로 정시에 활주로에 착륙하는 목표시간 (target time, T_i)보다 빨리 착륙시키면 E_i로, 늦게 착륙시키면 L_i로 설정하여 $E_i \le T_i \le L_i$시간에 착륙시키며, 각각의 단위시간당 소요비용을 c_E와 c_L이라 한다.

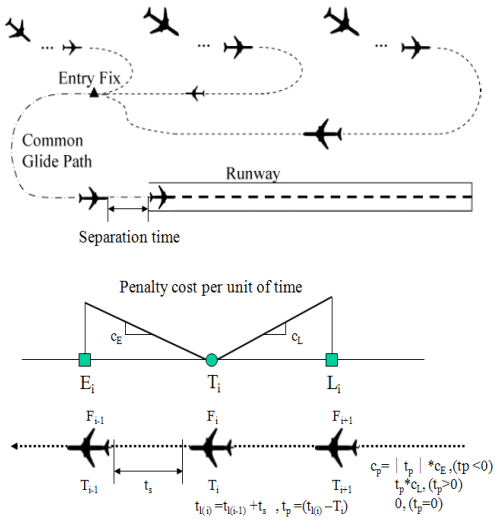


그림 1. ALP 개념
Fig. 1. Concept of ALP

ALP는 표 1의 Airland1[11] 예제와 같이 주어진다. ALP는 식 (1)을 만족시키도록 주어진 n대의 항공기들을 m개 활주로를 가진 공항에 $[E_i, L_i]$ 범위 내에서 착륙시키는 것이 목표이다.

$$z = \min \sum_{i=1}^{n} c_{p(i)} \qquad (1)$$

$$\text{where } t_{l(i)} = t_{l(i-1)} + t_s$$
$$t_{p(i)} = (t_{l(i)} - T_i)$$
$$c_{p(i)} = \begin{cases} |t_{p(i)}| \times c_E, t_{p(i)} < 0 \\ t_{p(i)} \times c_L, t_{p(i)} > 0 \\ 0, t_{p(i)} = 0 \end{cases}$$

표 1. Airland1 예제
Table 1. Example of Airland1

Plane (P_i)	Apperance time (a_i)	Landing time			Penalty cost per unit time		Separation time after i landing before j can land (t_s)									
		Earliest time (E_i)	Target time (T_i)	Latest time (L_i)	Landing before target (c_E)	Landing after target (c_L)	1	2	3	4	5	6	7	8	9	10
1	54	129	155	559	10	10	-	3	15	15	15	15	15	15	15	15
2	120	195	258	744	10	10	3	-	15	15	15	15	15	15	15	15
3	14	89	98	510	30	30	15	15	-	8	8	8	8	8	8	8
4	21	96	106	521	30	30	15	15	8	-	8	8	8	8	8	8
5	35	110	123	555	30	30	15	15	8	8	-	8	8	8	8	8
6	45	120	135	576	30	30	15	15	8	8	8	-	8	8	8	8
7	49	124	138	577	30	30	15	15	8	8	8	8	-	8	8	8
8	51	126	140	573	30	30	15	15	8	8	8	8	8	-	8	8
9	60	135	150	591	30	30	15	15	8	8	8	8	8	8	-	8
10	85	160	180	657	30	30	15	15	8	8	8	8	8	8	8	-

III. ALP의 다항시간 FTFSO 알고리즘

본 장에서는 ALP의 최적 해를 $O(n\log n)$의 다항시간으로 찾을 수 있는 첫 번째 목표시간 비행기 우선 착륙지원(FTFS) 원칙을 적용하여 주어진 n대의 항공기를 목표시간 오름차순으로 정렬한 초기 해를 구한다. 초기 해에 대해 최적화 과정을 수행하여 최적 해를 구한다. 제안된 알고리즘을 FTFSO라 하자. FTFSO는 다음과 같이 수행된다.

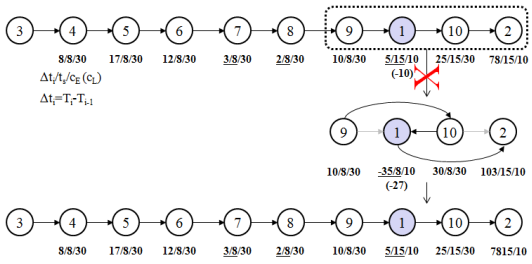
Step 1. 초기 착륙순서 결정
for $i=1$ to n /* 수행 복잡도 : $O(n\log n)$ */
목표시간 T_i 오름차순으로 정렬한다.
end
for $i=2$ to n /* 수행 복잡도 : $O(n)$ */
각 P_i 항공기에 대해 P_{i-1}과의 t_s ($i-1$ 행, i 열 셀 값)를 선택한다.
end
for $i=2$ to n /* 수행 복잡도 : $O(n)$ */
$i\rightarrow j\rightarrow k\rightarrow l$ 항공기 순서의 Σt_s에서 j번째 항 공기의 t_s가 주어진 t_s 행렬에서 최대값인 경우, $i\rightarrow k\rightarrow j\rightarrow l\,(j\leftrightarrow k)$로 순서를 변경한 $\Sigma t_s'$에 대해 $\Sigma t_s > \Sigma t_s'$이면 순서를 변경한다.
end
$t_{l(1)}=E_1$, P_n부터 역으로 $t_p<0$이 연속적으로 발생한 경우는 $T_l=T_i$로 설정한다.

Step 2. $m=1$ 최적화 /* 수행 복잡도 : $O(n)$ */
for $i=1$ to n
$t_p(i)<0$인 P_i의 t_l을 Σc_p가 최소값을 갖도록 증가시킨다.
end

Step 3. $m\geq 2$ 최적화 /* 수행 복잡도 : $O(n)$ */
이전의 보다 작은 활주로 개수에 대한 착륙순서 최적화 결과에 대해, $c_{p(i)}>0$인 항공기를 여분의 활주로로 이동시킨다.
for $i=2$ to n /* 수행 복잡도 : $O(n)$ */
각 활주로의 P_i 항공기에 대해 P_{i-1}과의 $t_s(i-1$행, i열 셀 값)를 선택한다.
end
for $i=1$ to n
$t_p(i)<0$인 P_i의 t_l을 Σc_p가 최소값을 갖도록 증가시킨다.
end
$t_p(i)>0$을 갖는 P_i 항공기가 속한 활주로와 다른 활주로의 P_j항공기를 상호 교환하여 $t_p(i)$를 감소시킬 수 있으면 교환을 수행한다. 이 경우 0:1, 1:1 또는 2:2의 교환이 발생할 수 있다. 만약, 항공기 교환이 발생하면 위의 최적화 과정을 다시 수행한다.

Airland1 예제 데이터에 대해 FTFSO를 적용한 결과는 그림 2에 제시되어 있다. 여기서는 $m=1,2,3$ 순서로 적용하였다.

P_i	a_i	Landing time			Penalty cost		Separation time									
		E_i	T_i	L_i	c_E	c_L	1	2	3	4	5	6	7	8	9	10
3	14	89	98	510	30	30	15	15	-	**8**	8	8	8	8	8	8
4	21	96	106	521	30	30	15	15	8	-	**8**	8	8	8	8	8
5	35	110	123	555	30	30	15	15	8	8	-	**8**	8	8	8	8
6	45	120	135	576	30	30	15	15	8	8	8	-	**8**	8	8	8
7	49	124	138	577	30	30	15	15	8	8	8	8	-	**8**	8	8
8	51	126	140	573	30	30	15	15	8	8	8	8	8	-	**8**	8
9	60	135	150	591	30	30	**15**	15	8	8	8	8	8	8	-	8
1	54	129	155	559	10	10	-	3	15	15	15	15	15	15	15	**15**
10	85	160	180	657	30	30	15	**15**	8	8	8	8	8	8	8	-
2	120	195	258	744	10	10	3	-	15	15	15	15	15	15	15	15



(a) Landing sequence

P_i	a_i	Landing time			Penalty cost		Sep. Time	Actual Landing Time	Penalty	
		E_i	T_i	L_i	c_E	c_L			Time	Cost
3	14	89	**98**	510	30	30	8	**98**	0	0
4	21	96	106	521	30	30	8	106	0	0
5	35	110	123	555	30	30	8	**118**	-5	150
6	45	120	135	576	30	30	8	126	-9	270
7	49	124	138	577	30	30	8	134	-4	120
8	51	126	140	573	30	30	8	142	2	60
9	60	135	150	591	30	30	15	150	0	0
1	54	129	155	559	10	10	15	165	10	100
10	85	160	180	657	30	30	15	180	0	0
2	120	195	258	744	10	10		**258**	0	0
										700

(b) Optimal solution for $m=1$

P_i	a_i	Landing time			Penalty cost		Sep. Time	Actual Landing Time	Penalty	
		E_i	T_i	L_i	c_E	c_L			Time	Cost
3	14	89	98	510	30	30	8	98	0	0
4	21	96	106	521	30	30	8	106	0	0
5	35	110	123	555	30	30	8	123	0	0
7	49	124	138	577	30	30	8	138	0	0
9	60	135	150	591	30	30	8	150	0	0
10	85	160	180	657	30	30		180	0	0
										0

P_i	a_i	Landing time			Penalty cost		Sep. Time	Actual Landing Time	Penalty	
		E_i	T_i	L_i	c_E	c_L			Time	Cost
6	45	120	135	576	30	30	8	135	0	0
8	51	126	140	573	30	30	15	143	3	90
1	54	129	155	559	10	10	3	158	3	30
2	120	195	258	744	10	10		258	0	0
										120

P_i	a_i	Landing time			Penalty cost		Sep. Time	Actual Landing Time	Penalty	
		E_i	T_i	L_i	c_E	c_L			Time	Cost
3	14	89	98	510	30	30	8	98	0	0
4	21	96	106	521	30	30	8	106	0	0
6	45	120	135	576	30	30	8	132	-3	90
8	51	126	140	573	30	30	8	140	0	0
9	60	135	150	591	30	30	8	150	0	0
10	85	160	180	657	30	30		180	0	0
										90

P_i	a_i	Landing time			Penalty cost		Sep. Time	Actual Landing Time	Penalty	
		E_i	T_i	L_i	c_E	c_L			Time	Cost
5	35	110	123	555	30	30	8	123	0	0
7	49	124	138	577	30	30	15	138	0	0
1	54	129	155	559	10	10	3	155	0	0
2	120	195	258	744	10	10		258	0	0
										0

(c) Optimal solution for $m=2$

P_i	a_i	Landing time			Penalty cost		Sep. Time	Actual Landing Time	Penalty	
		E_i	T_i	L_i	c_E	c_L			Time	Cost
3	14	89	98	510	30	30	8	98	0	0
4	21	96	106	521	30	30	8	106	0	0
6	45	120	135	576	30	30	8	135	0	0
9	60	135	150	591	30	30	8	150	0	0
10	85	160	180	657	30	30		180	0	0
										0

P_i	a_i	Landing time			Penalty cost		Sep. Time	Actual Landing Time	Penalty	
		E_i	T_i	L_i	c_E	c_L			Time	Cost
5	35	110	123	555	30	30	8	123	0	0
7	49	124	138	577	30	30	15	138	0	0
1	54	129	155	559	10	10	3	155	0	0
2	120	195	258	744	10	10		258	0	0
										0

P_i	a_i	Landing time			Penalty cost		Sep. Time	Actual Landing Time	Penalty	
		E_i	T_i	L_i	c_E	c_L			Time	Cost
8	51	126	140	573	30	30		140	0	0
										0

(d) Optimal solution for $m=3$

그림 2. Airland1 데이터에 대한 FTFSO
Fig. 2. FTFSO for Airland1 data

(a)에서 T_i 오름차순으로 정렬시킨 항공기 착륙순서는 3-4-5-6-7-8-9-1-10-2로 결정되었으며, t_s가 최대값 15를 갖는 P_1에 대해 순서가 변경되지는 않았음을 알 수 있다. (b)는 $m=1$인 경우로, P_3를 $t_l=T_i=98$로 설정한 결과 $\Sigma c_p=1930$을 얻었다. 다음으로, $t_l<0$인 P_5의 $t_l=114$를 P_{10}의 -4를 고려하여 118로 증가시키고, 마지막으로 $t_l<0$인 P_2를 $t_l=T_i=258$로 설정한 결과 $\Sigma c_p=700$으로 최적 해 z를 얻었다. (c)는 $m=2$인 경우로, $m=1$에서 두 번째로 발생한 $c_p>0$인 P_6부터 한 칸씩 건너뛰면서 P_8,P_1,P_2를 활주로 #2(R2)로 이동시켰다. 이 결과 $\Sigma c_p=0+120=120$을 얻었으며, 활주로 #1 (R1)의 P_5,P_7을 활주로 #2 (R2)의 P_6,P_8과 교환한 결과 $\Sigma c_p=90+0=90$으로 최적 해 z를 얻었다. (d)는 $m=3$인 경우로, $m=2$에서 $c_p>0$인 P_8을 활주로 #3 (R3)으로 이동시켜 $\Sigma c_p=0+0+0=0$의 최적 해 z를 얻었다.

IV. 실험 및 결과 분석

본 장에서는 Beasley[11]의 OR-Library에서 인용된 Airland2 ~ Airland8에 대해 제안된 FTFSO를 적용하여 본다. 실험 데이터에 대한 FTFSO를 적용한 결과는 표 2에 제시되어 있다. 예를 들면 Airland1의 항공기 대수 10대에 대해 $m=1$인 활주로 1개로 착륙을 시킬 경우 목표시간 순서는 3,4,5,6,7,8,9,1,10,2 항공기 순서이며, FTFSO 알고리즘 수행 결과인 최적 착륙순서 역시 목표시간 순서와 동일하며, 이 때 R1 (활주로 1번)의 각 항공기에 대한 항공기 번호(실제 착륙시간, 목표시간-지연착륙시간)을 표기하였다. 목표시간-지연착륙시간이 +이면 목표시간 이후에 착륙시킨 결과이며, -이면 보다 일찍 착륙시킴을 의미한다. 0이면 비용이 소요되지 않는 목표시간에 착륙시켰음을 알 수 있다.

실험 데이터에 대한 FTFSO와 기존 알고리즘들[1-10]을 비교한 결과는 표 3과 같다. 여기서 인용된 알고리즘 약어들은 다음을 의미한다.

TS : Linear Programming(LP)→Based Tree Search
DALP-H1 : Dynamically ALP-Heuristic
DALP-OPT : Dynamically ALP-Optimization
EOH : Extremal Optimisation-Hybrid
FCFS : First-Come First-Served (Target time ordering)

표 2. 실험 데이터에 대한 FTFSO 결과
Table 2. The result of FTFSO for experimental data

문제	n	m	$P_i(t_l, T_i - t_l)$	Σc_p
Airland1	10	1	Target time ordering: 3→4→5→6→7→8→9→1→10→2 Optimal sequence : 3→4→5→6→7→8→9→1→10→2 R1: 3(98,0)→4(106,0)→5(118,-5)→6(126,-9)→7(134,-4)→8(142,2)→9(150,0)→1(165,10)→10(180,0)→2(258,0)	**700**
		2	R1: 3→4→6→8→9→10 3(98,0)→4(106,0)→6(132,-3)→8(140,0)→9(150,0)→10(180,0) R2: 5→7→1→2 5(123,0)→7(138,0)→1(155,0)→2(258,0)	**90** **0**
		3	R1: 3→4→6→9→10 3(98,0)→4(106,0)→6(135,0)→9(150,0)→10(180,0) R2: 5→7→1→2 5(123,0)→7(138,0)→1(155,0)→2(258,0) R3: 8 8(140,0)	**0** **0** **0**
Airland2	15	1	Target time ordering: 3→4→5→6→8→7→9→10→1→14→13→2→12→11→15 Optimal sequence : 3→4→5→6→8→7→9→10→1→14→13→2→12→11→15 R1: 3(90,-3)→4(98,0)→5(106,-5)→6(114,-6)→8(122,2)→7(130,9)→9(138,10)→10(151,0)→14(171,0)→13(181,0)→1(196,41)→2(250,0)→12(313,0)→11(341,0)→15(344,2)	**1,480**
		2	R1: 3→5→6→7→10→14→13→2→12→11 3(93,0)→5(111,0)→6(120,0)→7(128,7)→10(151,0)→14(171,0)→13(181,0)→2(250,0)→12(313,0)→11(341,0) R2: 4→8→9→1→15 4(98,0)→8(120,0)→9(128,0)→1(155,0)→15(342,0)	**210** **0**
		3	R1: 3→5→6→10→14→13→2→12→11 3(93,0)→5(111,0)→6(120,0)→10(151,0)→14(171,0)→13(181,0)→2(250,0)→12(313,0)→11(341,0) R2: 4→8→9→1→10 4(98,0)→8(120,0)→9(128,0)→1(155,0)→10(342,0) R3: 7 7(121,0)	**0** **0** **0**
Airland3	20	1	Target time ordering: 1→6→8→4→12→10→9→11→19→20→3→2→7→15→5→18→14→13→17→16 Optimal sequence : 1→6→8→4→12→10→9→11→19→20→3→2→7→15→5→18→14→13→17→16 R1: 1(82,0)→6(101,-5)→8(109,1)→4(117,0)→12(125,-1)→10(133,3)→9(141,9)→11(149,0)→19(160,0)→20(169,0)→3(184,24)→2(197,0)→7(229,0)→15(258,0)→5(261,0)→18(287,0)→14(316,0)→13(336,0)→17(339,1)→16(409,0)	**820**
		2	R1: 1→6→12→9→11→19→20→2→7→15→5→18→14→13→16 1(82,0)→6(106,0)→12(126,0)→9(134,2)→11(149,0)→19(160,0)→20(169,0)→2(197,0)→7(229,0)→15(258,0)→5(261,0)→18(287,0)→14(316,0)→13(336,0)→16(409,0) R2: 8→4→10→3→17 8(108,0)→4(117,0)→10(130,0)→3(160,0)→17(338,0)	**60** **0**
		3	R1: 1→6→12→11→19→20→2→7→15→5→18→14→13→16 1(82,0)→6(106,0)→12(126,0)→11(149,0)→19(160,0)→20(169,0)→2(197,0)→7(229,0)→15(258,0)→5(261,0)→18(287,0)→14(316,0)→13(336,0)→16(409,0) R2: 8→4→10→3→17 8(108,0)→4(117,0)→10(130,0)→3(160,0)→17(338,0) R3: 9 9(132,0)	**0** **0** **0**
Airland4	20	1	Target time ordering: 1→2→5→9→8→6→7→13→12→16→19→18→17→15→3→4→10→14→11→20 Optimal sequence : 1→2→5→9→8→6→7→13→12→16→19→18→17→15→3→4→10→14→11→20 R1: 1(82,-10)→2(90,-3)→5(98,0)→9(106,-6)→8(114,-2)→6(122,5)→7(130,12)→13(138,1)→12(146,-10)→16(154,-2)→19(162,-6)→18(170,1)→17(178,8)→15(186,12)→3(201,18)→ 4(270,0)→10(280,0)→14(291,0)→11(295,0)→20(357,0)	**2,520**
		2	R1: 1→5→8→6→13→12→19→15→4→10→14→11→20 1(92,0)→5(100,2)→8(110,-6)→6(118,1)→13(137,0)→12(156,0)→19(168,0)→15(176,2)→4(270,0)→10(280,0)→14(291,0)→11(295,0)→20(357,0) R2: 2→9→7→16→18→17→3 2(93,0)→9(110,-2)→7(118,0)→16(156,0)→18(164,-5)→17(172,2)→3(187,4)	**330** **310**
		3	R1: 1→9→7→13→12→18→3→4→10→14→11→20 1(92,0)→9(110,-2)→7(118,0)→13(137,0)→12(156,0)→18(169,0)→3(184,1)→4(270,0)→10(280,0)→14(291,0)→11(295,0)→20(357,0) R2: 2→8→16→17 2(93,0)→8(116,0)→16(156,0)→17(170,0) R3: 5→6→19→15 5(98,0)→6(117,0)→19(166,-2)→15(174,0)	**70** **0** **60**
		4	R1: 1→9→13→12→18→4→10→14→11→20 1(92,0)→9(112,0)→13(137,0)→12(156,0)→18(169,0)→4(270,0)→10(280,0)→14(291,0)→11(295,0)→20(357,0) R2: 2→8→16→17 2(93,0)→8(116,0)→16(156,0)→17(170,0) R3: 5→6→19→3 5(98,0)→6(117,0)→19(168,0)→3(183,0) R4: 7→15 7(118,0)→15(174,0)	**0** **0** **0** **0**
Airland5	20	1	Target time ordering: 3→4→5→8→6→7→9→10→14→19→17→13→18→20→1→2→12→15→11→16 Optimal sequence : 3→4→5→8→6→7→9→10→14→19→17→13→18→20→1→2→12→15→11→16 R1: 3(82,-8)→4(90,-4)→5(98,-7)→8(106,-5)→6(114,2)→7(122,10)→9(130,13)→10(138,1)→14(146,-5)→19(154,1)→17(162,3)→13(170,9)→18(178,16)→20(186,4)→1(201,46)→2(246,0)→12(280,0)→15(301,0)→11(307,0)→16(393,0)	**3,100**
		2	R1=3→5→6→9→10→14→17→18→20→2→12→15→11→16 3(90,0)→5(104,-1)→6(112,0)→9(120,3)→10(137,0)→14(151,0)→17(159,0)→18(167,5)→20(182,0)→2(246,0)→12(280,0)→15(301,0)→11(307,0)→16(393,0) R2=4→8→7→1→19→13 4(94,0)→8(104,-7)→7(112,0)→1(138,-17)→19(153,0)→13(161,0)	**270** **380**
		3	R1: 3→5→6→10→14→17→20→2→12→15→11→16 3(90,0)→5(105,0)→6(113,1)→10(137,0)→14(151,0)→17(159,0)→20(182,0)→2(246,0)→12(280,0)→15(301,0)→11(307,0)→16(393,0) R2: 4→8→9→1→18 4(94,0)→8(111,0)→9(119,2)→1(147,-8)→18(162,0) R3: 7→19→13 7(112,0)→19(153,0)→13(161,0)	**30** **140** **0**
		4	R1=3→5→9→10→14→17→20→2→12→15→11→16 3(90,0)→5(105,0)→9(117,0)→10(137,0)→14(151,0)→17(159,0)→20(182,0)→2(246,0)→12(280,0)→15(301,0)→11(307,0)→16(393,0) R2: 4→8→18 4(94,0)→8(111,0)→18(162,0) R3: 7→19→13 7(112,0)→19(153,0)→13(161,0) R4: 6→1 6(112,0)→1(155,0)	**0** **0** **0** **0**

문제	n	m	$P_i(t_l, T_i - t_l)$	Σc_p
Airland6	30	1	Target time ordering: 1→2→3→4→5→6→7→8→9→10→11→12→13→14→15→16→17→18→19→20→21→22→23→24→25→26→27→28→29→30 Optimal sequence : 1→2→3→4→5→6→7→8→9→10→11→12→13→14→15→16→17→18→19→20→21→22→23→24→25→26→27→28→29→30 R1: 1(0,0)→2(96,17)→3(192,48)→4(392,188)→5(464,200)→6(560,240)→7(760,232)→8(832,197)→9(1032,302)→10(1112,346)→11(1184,394)→12(1280,360)→13(1461,415)→14(1591,485)→15(1671,535)→16(1751,585)→17(1831,598)→18(1903,261)→19(1999,284)→20(2180,410)→21(2252,178)→22(2348,180)→23(2576,317)→24(2656,229)→25(2728,247)→26(2928,249)→27(2998,115)→28(3098,116)→29(3170,124)→30(3266,175)	24,442
		2	R1: 1→4→6→7→8→11→13→16→14→18→19→21→22→24→26→27→29 1(0,0)→4(204,0)→6(320,0)→7(528,0)→8(635,0)→11(790,0)→13(1046,0)→16(1166,0)→14(1276,170)→18(1642,0)→19(1738,23)→21(2074,0)→22(2170,2)→24(2427,0)→26(2679,0)→27(2883,0)→29(3046,0) R2: 2→3→5→9→10→12→15→17→20→23→25→28→30 2(79,0)→3(175,31)→5(271,7)→9(730,0)→10(810,44)→12(920,0)→15(1136,0)→17(1233,0)→20(1770,0)→23(2259,0)→25(2481,0)→28(2982,0)→30(3091,0)	270 284
		3	R1: 1→4→6→7→8→11→13→16→18→21→24→26→27→29 1(0,0)→4(204,0)→6(320,0)→7(528,0)→8(635,0)→11(790,0)→13(1046,0)→16(1166,0)→18(1642,0)→21(2074,0)→24(2427,0)→26(2679,0)→27(2883,0)→29(3046,0) R2: 2→9→12→15→17→20→23→25→28→30 2(79,0)→9(730,0)→12(920,0)→15(1136,0)→17(1233,0)→20(1770,0)→23(2259,0)→25(2481,0)→28(2982,0)→30(3091,0) R3: 3→5→10→14→19→22 3(144,0)→5(264,0)→10(766,0)→14(1106,0)→19(1715,0)→22(2168,0)	0 0 0
Airland7	44	1	Target time ordering: 1→2→3→4→5→6→7→8→9→10→11→12→13→14→15→16→17→18→19→20→21→22→23→24→25→26→27→28→29→30→31→32→33→34→35→36→37→38→39→40→41→42→43→44 Optimal sequence : 1→2→3→4→5→6→7→8→9→10→11→12→13→14→15→16→17→18→19→20→21→22→23→24→25→26→27→28→29→30→31→32→33→34→35→36→37→38→39→40→41→42→43→44 R1: 1(0,0)→2(96,-41)→3(296,25)→4(376,25)→5(456,25)→6(528,25)→7(624,-2)→8(720,-7)→9(920,25)→10(992,25)→11(1192,25)→12(1264,25)→13(1464,25)→14(1544,25)→15(1616,25)→16(1816,25)→17(1896,25)→18(1968,25)→19(2168,25)→20(2248,25)→21(2320,25)→22(2416,25)→23(2616,25)→24(2688,25)→25(2888,25)→26(2968,25)→27(3048,25)→28(3128,25)→29(3200,25)→30(3296,25)→31(3496,25)→32(3568,25)→33(3768,25)→34(3840,25)→35(3936,25)→36(4136,25)→37(4216,25)→38(4296,25)→39(4368,25)→40(4464,25)→41(4560,25)→42(4656,25)→43(4752,25)→44(4952,25)	1,550
		2	R1: 1→2→7→8 1(0,0)→2(137,0)→7(626,0)→8(727,0) R2: 3→4→5→6→9→10→11→12→13→14→15→16→17→18→19→20→21→22→23→24→25→26→27→28→29→30→31→32→33→34→35→36→37→38→39→40→41→42→43→44 3(271,0)→4(351,0)→5(431,0)→6(503,0)→9(895,0)→10(967,0)→11(1167,0)→12(1239,0)→13(1439,0)→14(1519,0)→15(1591,0)→16(1791,0)→17(1871,0)→18(1943,0)→19(2143,0)→20(2223,0)→21(2295,0)→22(2391,0)→23(2591,0)→24(2663,0)→25(2863,0)→26(2943,0)→27(3023,0)→28(3103,0)→29(3175,0)→30(3271,0)→31(3471,0)→32(3543,0)→33(3743,0)→34(3815,0)→35(3911,0)→36(4111,0)→37(4191,0)→38(4271,0)→39(4343,0)→40(4439,0)→41(4535,0)→42(4631,0)→43(4727,0)→44(4927,0)	0 0
Airland8	50	1	Target time ordering: 1→6→8→4→12→10→9→11→3→19→20→2→7→15→5→24→18→14→23→13→17→50→26→25→43→16→35→22→27→44→45→49→28→32→29→33→47→34→37→38→48→21→30→39→46→31→36→40→41→42 Optimal sequence : 1→6→8→4→12→10→9→11→3→19→20→2→7→15→5→24→18→14→13→23→17→50→26→16→25→43→35→22→27→44→45→49→28→32→29→33→47→34→37→38→48→21→30→39→46→31→36→40→41→42 R1: 1(82,0)→6(100,-6)→8(108,0)→4(116,-1)→12(124,-2)→10(132,2)→9(135,3)→11(149,0)→3(152,-8)→19(160,0)→20(175,6)→2(197,0)→7(229,0)→15(258,0)→5(261,0)→24(269,0)→18(287,0)→14(316,0)→13(325,-11)→23(333,0)→17(341,3)→50(345,0)→26(378,0)→16(390,-19)→25(398,0)→43(401,0)→35(410,-2)→22(425,0)→27(428,2)→44(441,-9)→45(456,0)→49(471,3)→28(498,0)→32(512,0)→29(516,-5)→33(524,0)→47(527,0)→34(556,-6)→37(564,0)→38(569,-3)→48(577,0)→21(619,-9)→30(634,0)→39(658,4)→46(673,0)→31(688,14)→36(717,0)→40(725,0)→41(740,8)→42(763,0)	1,945
		2	R1: 1→6→10→9→11→19→14→23→17→25→16→22→45→29→37→48→30→31→41 1(82,0)→6(106,0)→10(130,0)→9(133,1)→11(149,0)→19(160,0)→14(316,0)→23(333,0)→17(341,3)→25(398,0)→16(409,0)→22(425,0)→45(456,0)→29(521,0)→37(564,0)→48(577,0)→30(634,0)→31(674,0)→41(732,0) R2: 8→4→12→3→20→2→7→15→5→24→18→13→50→26→43→35→27→44→49→28→32→33→47→34→38→21→39→46→36→40→42 8(108,0)→4(117,0)→12(126,0)→3(154,-6)→20(169,0)→2(197,0)→7(229,0)→15(258,0)→5(261,0)→24(269,0)→18(287,0)→13(336,0)→50(345,0)→26(378,0)→43(401,0)→35(411,-1)→27(426,0)→44(450,0)→49(468,0)→28(498,0)→32(512,0)→33(524,0)→47(527,0)→34(562,0)→38(572,0)→21(628,0)→39(654,0)→46(673,0)→36(717,0)→40(725,0)→42(763,0)	60 75
		3	R1: 1→6→12→11→19→14→23→25→16→22→45→29→37→48→30→31→41 1(82,0)→6(106,0)→12(126,0)→11(149,0)→19(160,0)→14(316,0)→23(333,0)→25(398,0)→16(409,0)→22(425,0)→45(456,0)→29(521,0)→37(564,0)→48(577,0)→30(634,0)→31(674,0)→41(732,0) R2: 8→4→10→20→2→7→15→5→24→18→13→50→26→43→27→44→49→28→32→33→47→34→38→21→39→46→36→40→42 8(108,0)→4(117,0)→10(130,0)→20(169,0)→2(197,0)→7(229,0)→15(258,0)→5(261,0)→24(269,0)→18(287,0)→13(336,0)→50(345,0)→26(378,0)→43(401,0)→27(426,0)→44(450,0)→49(468,0)→28(498,0)→32(512,0)→33(524,0)→47(527,0)→34(562,0)→38(572,0)→21(628,0)→39(654,0)→46673,0)→36(717,0)→40(725,0)→42(763,0) R3: 9→3→17→35 9(132,0)→3(160,0)→17(338,0)→35(412,0)	0 0 0

SCS : Scatter Search
BA : Bionomic Algorithm
HBA : Hybrid Bat Algorithm
PSA : Awasthi et al.[5]
GRASP : Greedy Randomized Adaptive Search Procedure
ACA : Ant Colony Algorithm
IACA : Improved Ant Colony Algorithm
CA : Cellular Automation
CAO : Cellular Automation-based Optimization
GA : Genetic Algorithm
ACGA : Ant Colony optimization Genetic Algorithm
FTFSO : First-Target First-Served (Target time ordering) Optimization

표 3에서는 각 실험 데이터에 대해 $m=1,2,3,4$의 활주로 개수에 대한 실험을 수행하여 3+3+3+4+4+3+2+3=25개 활주로 실험 데이터로 구성되어 있다. 이들 실험 데이터에 대해 기존의 알고리즘들과 제안된 FTFSO에 대해 모든 항공기의 목표시간 대비 선 착륙 비용 c_E와 후착륙비용 c_L 의 합 Σc_p를 비교한 결과이다.

FTFSO는 8개 데이터의 25개 활주로 모든 데이터에 대해 최적 해를 구하였음을 알 수 있다. 특히, 기존의 선형계획법, 휴리스틱이나 메타휴리스틱의 어떠한 알고리즘으로도 최적 해를 구하지 못한 Airland8의 $m=1$ 활주로 데이터에 대해서는 기존에 알려진 최적 해 z_{opt}를 개선하는 효과를 얻었다.

표 3. 알고리즘 성능 비교
Table 3. Comparison of algorithm performance

문제	n	m	z_{opt}	TS [1]	DALP-H1 [2]	DALP-OPT [2]	EOH [4,6]	FCFS [1,7]	SCS [5,7]	BA [5,7]	HBA [7]	PSA [5]	GRASP [3]	CA [9]	ACA [8]	IACA [7,8]	CAO [9]	GA [10]	ACGA [10]	FTFSO
Airland1	10	1	**700**	**700**	740	740	-	1790	**700**	**700**	-	**700**	**700**	1260	1150	**700**	**700**	820	**700**	**700**
		2	**90**	**90**	120	**90**	-	120	**90**	**90**	**90**	**90**	-	-	120	**90**	-	**90**	**90**	**90**
		3	**0**	**0**	**0**	**0**	-	**0**	**0**	**0**	**0**	**0**	-	-	**0**	**0**	-	0	**0**	**0**
Airland2	15	1	**1480**	**1480**	1870	1730	-	2610	**1480**	**1480**	-	**1480**	**1480**	2500	1840	**1480**	**1480**	1720	1720	**1480**
		2	**210**	**210**	**210**	**210**	-	**210**	**210**	**210**	**210**	**210**	-	-	**210**	**210**	-	220	**210**	**210**
		3	**0**	**0**	**0**	**0**	-	**0**	**0**	**0**	**0**	**0**	-	-	**0**	**0**	-	10	**0**	**0**
Airland3	20	1	**820**	**820**	1440	940	-	2930	**820**	822.3	-	**820**	**820**	2150	2540	**820**	**820**	1750	850	**820**
		2	**60**	**60**	**60**	**60**	-	**60**	**60**	**60**	**60**	**60**	-	-	**60**	**60**	-	570	**60**	**60**
		3	**0**	**0**	**0**	**0**	-	**0**	**0**	**0**	**0**	**0**	-	-	**0**	**0**	-	320	**0**	**0**
Airland4	20	1	**2520**	**2520**	2670	2700	-	6290	2544	**2520**	-	**2520**	2520	3300	4820	2520	2520	6580	4480	**2520**
		2	**640**	640	680	680	-	1560	640	640	640	640	-	-	680	640	-	1770	680	**640**
		3	**130**	**130**	**130**	**130**	-	330	**130**	**130**	**130**	130	-	-	130	130	-	600	130	**130**
		4	**0**	**0**	**0**	**0**	-	60	**0**	**0**	**0**	0	-	-	0	0	-	330	0	**0**
Airland5	20	1	**3100**	3100	6130	3810	-	8370	3290	3100	-	3100	3100	4300	6260	3100	3680	5800	4800	**3100**
		2	**650**	650	1070	680	-	1440	650	650	890	650	-	-	1210	730	-	1650	729	**650**
		3	**170**	170	240	240	-	240	170	170	170	170	-	-	330	170	-	560	240	**170**
		4	**0**	0	0	0	-	0	0	0	0	0	-	-	0	0	-	440	0	**0**
Airland6	30	1	**2444**	24442	24442	24442	-	2444	2444	24442	-	2444	24442	2444	64001	24442	24442	-	24442	**24442**
		2	**2**	554	882	809	-	2	2	554	636	2	-	-	2394	837	-	-	554	**554**
		3	**554**	0	0	0	-	882	554	0	0	554	-	-	240	0	-	-	0	**0**
			0					0	0			0								
Airland7	44	1	**1550**	1550	3974	3974	-	1550	1550	1550	-	1550	1550	6247	5342	1550	1550	-	1550	**1550**
		2	**0**	0	0	0	-	0	0	0	0	0	-	-	160	0	-	-	200	**0**
Airland8	50	1	1950	1950	2915	2000	2320	2683	2964.	2041	-	1995	1950	7040	13840	2185	2635	-	3240	**1945**
		2	**135**	135	255	135	-	5	97	135	180	135	-	-	835	165	-	-	160	**135**
		3	**0**	0	0	0	-	1014	135	0	0	0	-	-	195	15	-	-	0	**0**
								0	0											
								4825												

V. 결론

공항에 접근하는 항공기들의 시간 간격이 불규칙한 다수의 항공기들을 최소의 비용으로 착륙시키는 ALP는 대해 최적 해를 구하기 어려워 다양한 메타휴리스틱 방법들이 제안되고 있다.

본 논문에서는 ALP에 대해 $O(n\log n)$의 다항시간으로 최적 해를 구하는 휴리스틱 알고리즘을 제안하였다. 제안된 알고리즘은 단지 착륙 목표시간 오름차순으로 정렬시키고, 항공기들 간의 분리 시간 간격과 목표시간과의 차이에 따른 소요비용을 고려하여 착륙순서를 변경시킨 최적화 과정을 수행하는 방법을 적용하였다. ALP에 대한 예제 데이터인 Airland1 ~ Airland8에 대해 소요비용 $\Sigma c_p = 0$가 되는 활주로 개수 m까지 25개 데이터를 실험한 결과 모든 데이터에 대해서 최적 해를 구하였다. 또한, Airland8의 $m = 1$활주로 데이터에 대해서는 기존에 알려진 최적 해 z_{opt}를 개선하였다.

참고문헌

[1] J. E. Besley, M. Krishnamoorthy, Y. M. Sharaiha, and D. Abramson, "Scheduling Aircraft Landings - The Static Case," Transportation Science, Vol. 34, No. 2, pp. 180-197, May 2000.

[2] J. E. Besley, M. Krishnamoorthy, Y. M. Sharaiha, and D. Abramson, "Displacement Problem and Dynamically Scheduling Aircraft Landings," Journal of the Operational Research Society, Vol. 55, No. 1, pp. 54-64, Jan. 2004.

[3] B. C. Fiss, "GRASP Applicado ao Problema de Aterrissagem de Aviões," Universidade Federal do Rio Grande do Sul, pp. 1-10, 2010.

[4] I. Moser and T. Hendtlass, "Solving Dynamic Single-Runway Aircraft Landing Problems with Extremal Optimisation," Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling, pp. 206-211, Apr. 2007.

[5] A. Awasthi, O. Kramer, and J. Lässig, "Aircraft Landing Problem: Efficient Algorithm for a Given Landing Sequence," 16th IEEE International Conference on Computational Science and Engineering in Data Structures and Algorithms, pp. 1-16, Oct. 2013.

[6] I Moser, "Scheduling Aircraft Landings Dynamically Using Stochastic and Deterministic Elements," International Journal of Information Technology and Intelligent Computing, Vol. 2, No. 1, pp. 1-21, Jan.

2007.

〔7〕 J. Xie, Y. Zhou, and H. Zheng, "A Hybrid Metaheuristic for Multiple Runways Aircraft Landing Problem Based on Bat Algorithm," Journal of Applied Mathematics, Vol. 2013, pp. 1-8, Jul. 2013.

〔8〕 G. Bencheikh, J. Boukachour, and A. E. H. Alaoui, "Improved Ant Colony Algorithm to Solve the Aircraft Landing Problem," International Journal of Computer Theory and Engineering, Vol. 3, No. 2, pp. 224-233, Apr. 2011.

〔9〕 S. Yu, X. Cao, M. Hu, W. Du, and J. Zhang, "A Real-time Schedule Method for Aircraft Landing Scheduling Problem Based on Cellular Automation," Applied Soft Computing, Vol. 11, No. 4, pp. 3485-3493, Jun. 2011.

〔10〕 G. Bencheikh, J. Boukachour, A. E. H. Alaoui, and F. E. Khoukhi, "Hybrid Method for Aircraft Landing Scheduling Based on a Job Shop Formulation," International Journal of Computer Science and Network Security, Vol. 9, No. 8, pp. 78-88, Aug. 2009.

〔11〕 J. E. Beasley, "OR-Library: Equitable Partitioning Problem," http://people.brunel.ac.uk/~mastjjb/jeb/orlib/eppinfo. html, 2013.

〔12〕 X. B. Hu and E. A. D. Paolo, "A Ripple- Spreading Genetic Algorithm for the Aircraft Sequencing Problem," Evolutionary Computation, Vol. 19, No. 1, pp. 77-106, Feb. 2011.

〔13〕 K. Kahn and A. Raith, "The Multicriteria Aircraft Landing Problem," The 21st International Conference on Multiple Criteria Decision Making (MCDM), pp. 1-32, Jun. 2011.

저 자 소 개

이 상 운(Sang-Un, Lee)
1983년 ~ 1987년 :
한국항공대학교 항공전자공학과 (학사)
1995년 ~ 1997년 :
경상대학교 컴퓨터과학과 (석사)
1998년 ~ 2001년 :
경상대학교 컴퓨터과학과 (박사)
2003.3 ~ 현 재 :
강릉원주대학교 멀티미디어공학과 부교수
관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 개발 방법론, 소프트웨어 신뢰성, 그래프 알고리즘
e-mail : sulee@gwnu.ac.kr