AIRCRAFT LANDING PROBLEM (ALP)

CS 254 -Project Report

Submitted By:-

Shubham Nimesh(180001054)

AnmolGomra(180001007)

Submitted To:-

Dr. Kapil Ahuja

Associate prof. CSE(IIT Indore)

INDEX:-

Introduction:-

The air traffic (AT) flow planning and control systems are designed to ensure safe, ordered and accelerated AT, including in airport and airline hubs, reduce the costs of airlines and passengers, and reduce the negative environmental impact, taking into account the uncertainty of AT forecasting. The complexity of planning and regulating AT flows in the vicinity of an aerodrome and the workload of dispatchers increases with increasing AT intensity. At the same time, management efficiency drops, aircraft delays and fuel consumption increases. To prevent such situations, special tools are developed to support the work of the dispatcher in the planning and regulation of AT flows, which perform part of the dispatcher's functions and allow more efficient use of the capacity of the runway and airspace. In Europe, such tools are called Arrival Manager (AMAN) and Departure Manager (DMAN).

The capacity of a runway system represents a bottleneck at many international airports. The current practice at airports is to land approaching aircraft on a first-come, first-served basis. An active rescheduling of aircraft landing times increases runway capacity or reduces delays. The problem of finding an optimal schedule for aircraft landings is referred to as the "aircraft landing problem". The objective is to minimize the total delay of aircraft landings or the respective cost. The necessary separation time between two operations must be met. Due to the complexity of this scheduling problem, recent research has been focused on developing heuristic solution approaches.

History:-

The aircraft landing problem described inthis paperwas first introduced and studied by Beasley inthe mid-nineties.Since then,ithasbeenstudiedby several researchers using different metaheuristics, hybrid metaheuristics, linear programming, variants of exact branch and bound algorithms etc., for both the staticand dynamic cases of the problem.In 1995, Beasley et al.presented a mixed-integer zero-one formulation oftheproblem forthesingle runway case and later extended it tothe multiple runway case.The ALP was studied for up to 50 aircraft with multiple runways using linear programming based tree search and an effective heuristic algorithm for the problem.Again in 1995, Abela et al. proposed a genetic algorithm and a branch and bound algorithm to solve the problem of scheduling aircraft landings.Ernst et al.presented a simplex algorithm which evaluated the landing times based on somepartialordering information.This method was used in a problem space search heuristic as well as a branch-and-bound methodfor both, the single andmultiple runwaycase, for again up to 50 aircraft.Beasley et al.adopted similar methodologiesand presented extended results.In 1998, Ciesielski et al.developed areal time algorithm for the aircraft landingsusinga genetic algorithm andperformed experiments onlanding data for the Sydneyairport on thebusiestday of theyear.In2001, Beasley et al.developed a population heuristic and implemented it on actual operational data related toaircraft landings at the LondonHeathrow airport.The dynamic case of theALP was studied again by Beasley et al.by expressing itas a displacement problem and using heuristics and linear programming.In 2006, Pinol and Beasley presented two heuristic techniques, Scatter Search and the Bionomic Algorithm and published results forthe available test problems involving up to 500 aircraft and 5 runways.The dynamic case of the problem for thesingle-runway case was again studied byMoser et al.in 2007.They used extremal optimization along with a deterministic algorithm to optimizea landing sequence.In 2008 Tang et al.implemented a multi-objective evolutionary approach to simultaneously minimize the total scheduled time of arrival and the total cost incurred.In 2009, Bencheikhet al. approached the ALP using hybrid methods combining genetic algorithms and ant colony optimization by formulating the problem as a job shop scheduling problem.The same authors presented an ant colony algorithm along with a newheuristic toadjust thelanding times of theaircraft in a given landing sequencein order to reduce thetotal penalty cost, in 2011.In 2012, a hybrid meta-heuristicalgorithm was suggested using simulated annealing with variable neighbourhood search and variable neighbourhood descent.

Related Work:

The aircraft landing problem described in this paper was first introduced and studied by Beasley in the mid-nineties. Since then, it has been studied by several researchers using different metaheuristics, hybrid metaheuristics, linear programming, variants of exact branch and bound algorithms etc., for both the static and dynamic cases of the problem. In 1995, Beasley et al. presented a mixed-integer zero-one formulation of the problem for the single runway case and later extended it to the multiple runway case. The ALP was studied for up to 50 aircraft with multiple runways using linear programming based tree search and an effective heuristic algorithm for the problem. Again in 1995, Abela et al. proposed a genetic algorithm and a branch and bound algorithm to solve the problem of scheduling aircraft landings. Ernst et al. presented a simplex algorithm which evaluated the landing times based on some partial ordering information. This method was used in a problem space search heuristic as well as a branch-and-bound method for both, the single and multiple runway case, for again up to 50 aircraft. Beasley et al. adopted similar methodologies and presented extended results. In 1998, Ciesielski et al. developed a real time algorithm for the aircraft landings using a genetic algorithm and performed experiments on landing data for the Sydney airport on the busiest day of the year. In 2001, Beasley et al. developed a population heuristic and implemented it on actual operational data related to aircraft landings at the London Heathrow airport. The dynamic case of the ALP was studied again by Beasley et al. by expressing it as a displacement problem and using heuristics and linear programming. In 2006, Pinol and Beasley presented two heuristic techniques, Scatter Search and the Bionomic Algorithm and published results for the available test problems involving up to 500 aircraft and 5 runways. The dynamic case of the problem for the single-runway case was again studied by Moser et al. in 2007. They used extremal optimization along with a deterministic algorithm to optimize a landing sequence. In 2008 Tang et al. implemented a multi-objective evolutionary approach to simultaneously minimize the total scheduled time of arrival and the total cost incurred. In 2009, Bencheikh et al. approached the ALP using hybrid methods combining genetic algorithms and ant colony optimization by formulating the problem as a job shop scheduling problem. The same authors presented an ant colony algorithm along with a new heuristic to adjust the landing times of the aircraft in a given landing sequence in order to reduce the total penalty cost, in 2011 .In 2012, a hybrid meta-heuristic algorithm was suggested using simulated annealing with variable neighbourhood search and variable neighbourhood descent

Problem Formulation

In this section we give the mathematical formulation of the static aircraft landing problem based on [3]. We also define some new parameters which are later used in the presented algorithm in the next sections.

Let,

N = the number of aircraft,

E_i=theearliestlandingtimeforaircraft i,i=1,2,...,N,

L_i=thelatestlandingtimeforaircraft$i$,i =1,2,...,N,

T_i=thetargetlandingtimeforaircraft$i$,i =1,2,...,N,

ST_i= the scheduled landing time for aircraft$i$,

$S_{i,j}$= the required separation time between planes $i$and j, where plane $i$lands before plane j on the same runway, $i \neq j$,

$s_{i,j}$= the required separation time between planes $i$and j, where plane $i$lands before plane j on different runways, $i \neq j$,

g_i= the penalty cost per time unit associated with plane $i$for landing before T_i,

h_i= the penalty cost per time unit associated with plane $i$for landing after T_i,

α_i= earliness(time) of plane i from T_i,

$\alpha_i=\max\{0,T_i-ST_i\}$,$i$=1,2,...,$N$,

β_i= tardiness (time) of plane i fromT_i,

$\beta i=\max\{0,ST_i-T_i\}$, i=1,2,...,N.

The total penalty corresponding to any aircraft $i$is then expressed as $\alpha_i g_i+\beta_i h_i$. If aircraft $i$lands at its target landing time then both $\alpha_i$and $\beta_i$are equal to zero and the cost incurred by its landing is equal to zero. However, if aircraft $i$does not land at T_i, either $\alpha_i$or $\beta_i$is non-zero and there is a strictly positive cost incurred. The objective function of the problem can now be defined as

N

$$\min \sum_{i=1} (\alpha_i g_i + \beta_i h_i). \quad (1)$$

The Solution to single runway problem:

1. Assign a start time (s)

2. The assumption is the first plane will land at that the start time (s) and it takes a second for landing.

The situation can be relatively changed when being in applied in a real life scenario.

3. Each plane will have a landing window gap and a minimum collision prevention gap which can be modified.

Earliest Landing Time (Ei) . This will be provided to each plane depending upon starting time and the assumption is that no plane can reach before this time. After the first plane lands the earliest landing time for other planes will be given while takin in the consideration the minimum collision gap.

Latest Landing Time(Li). This Time will depend on **Ei** and cetail value will be assigned depending on the landing window gap

Collision Prevention Gap: C

4. An Optimal time for each plane to be landed is the mid time between **Ei** and **Li** This time is given by **O(t)**.

5. The cost associated with a plane landing before **Ti**, is given by **Gi** and the cost associated with the plane landing after **O(t)** is given by hi .

6. Obviously in real life scenarios chances are that the plane will not land at the optimal time **O(t)**. The time at which it will be landing is the **STi** = the scheduled landing time.

This time will be taken as input for each plane and depending upon this input new landing windows , minimum collision gaps and Optimal time for next planes will be created.

7. There are also chances that a plane might miss it's landing window. The plane will then be provided a new scheduled time inbetween the the landing window gaps of next flights.

8. If a plane lands before/after its target time and no previous planes are in the air then nothing changes and the minimum collision gap is still maintained.

9. If a plane lands before/after its target time and some previous plane is in the air then it will be landed and it will be made sure due to this landing the minimum collision gap is still maintainedas for example the plane which was in the air is landed just at latest time (**Li**) of the previously landed plane than the minimum landing window gap and the Earliest Landing time (**Ei**) of the next plane is also affected and that must not happen.

The Solution to multiple runway problem:

The solution is similar to that of single runway but in this case , more runways so things become quite faster and more number of planes in a given interval of time. Planes do still land in a sequential manner.

Like previously we were maintaining a gap to avoid collision between 2 planes back to back on a single runway ,

Now we also maintain a gap between 2 planes on 2 different runways so as to avoid collision sideways.

The interval for this gap can be similar to that of the gap on single runways. Time complexity will also increase as we now also have to assign planes to all of these runways and thus create a new schedule.

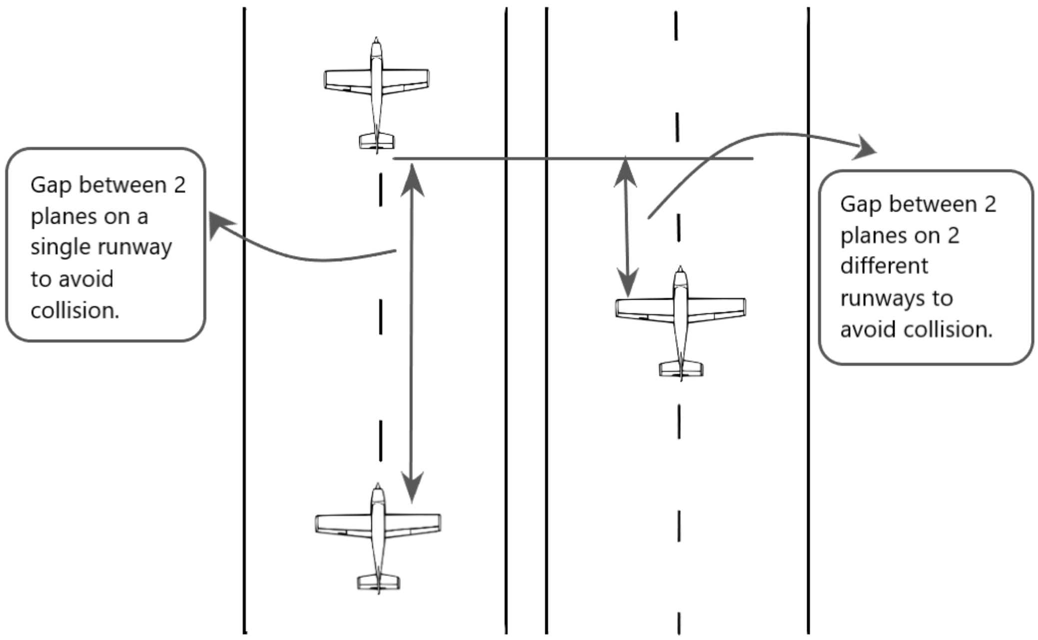
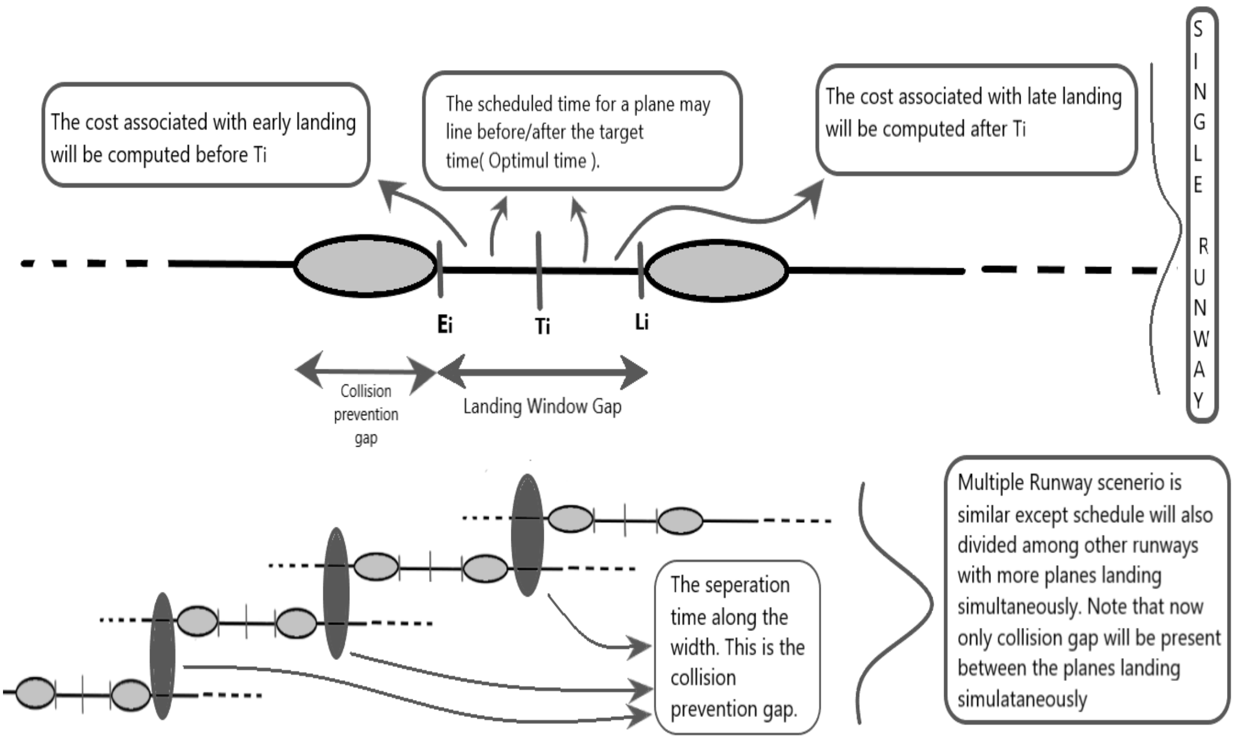


Diagram To display collision gap in single runway and multiple runway.

Flow Diagram of algorithm used:

Flow diagram for the algorithm used is as follows:



Landing Problem implementation in C++:

```cpp
#include <iostream>
#include <ctime>
using namespace std;
int main()
{
    time_t curr_time;
    curr_time = time(NULL);
//
    tm *tm_local = localtime(&curr_time);
//  cout << "Current local time : " << tm_local->tm_hour << ":" << tm_local->tm_min << ":" << tm_local->tm_sec;
//  return 0;

 int total_planes;
 float flight_gap, window_gap;

 string s ;
 cout<<"Enter total number of planes to be landed: ";
 cin>>total_planes;
 cout<<"Enter the min required landing gap in secs: ";
 cin>>flight_gap;
 cout<<"Enter the min required window gap in secs: ";
 cin>>window_gap;
 cout<<"Enter start to give the plane schedule for the day: ";
 cin>>s;


 if(s == "start" || s == "s")
 {
   float hour = tm_local->tm_hour;
   float minute =  tm_local->tm_min;
   float secs =  tm_local->tm_sec;

   float secs_early =  tm_local->tm_sec;
   float mins_early =  tm_local->tm_min;
   float hours_early =  tm_local->tm_hour;

   float secs_latest = tm_local->tm_sec;
   float mins latest = tm local->tm min:
```

```cpp
    float secs_latest = tm_local->tm_sec;
    float mins_latest = tm_local->tm_min;
    float hours_latest = tm_local->tm_hour;

    float mainArray[total_planes][6];

    for(int i = 1;i<=total_planes;i++){
     time_t curr_time;
     curr_time = time(NULL);

     tm *tm_local = localtime(&curr_time);

     if(i == 1){
     cout<<i<<" number plane will be landing at: "<< hour << ":" << minute << ":" << secs;
     cout<<endl;
     cout<<endl;
     }
     else{
     secs_early =  secs_latest + flight_gap;
     mins_early = mins_latest;
     hours_early = hours_latest;

     secs_latest = secs_early + window_gap;

     if(secs_early >=60){
         secs_early = secs_early - 60;
         mins_early += 1;
     }
     if(mins_early >= 60){
         mins_early = mins_early - 60;
         hours_early += 1;
     }
     if(hours_early >= 24){
         hours_early = hours_early - 24;
     }

     if(secs_latest >=60){
         secs_latest = secs_latest - 60;
```


```cpp
     if(secs_latest >=60){
         secs_latest = secs_latest - 60;
```


```cpp
         mins_latest += 1;
     }
     if(mins_latest >= 60){
         mins_latest = mins_latest - 60;
         hours_latest += 1;
     }
     if(hours_latest >= 24){
         hours_latest = hours_latest - 24;
     }

     mainArray[i-1][0] = hours_early;
     mainArray[i-1][1] = mins_early;
     mainArray[i-1][2] = secs_early;
     mainArray[i-1][3] = hours_latest;
     mainArray[i-1][4] = mins_latest;
     mainArray[i-1][5] = secs_latest;
     cout<<i<<" number plane will have earliest landing window at: "<< hours_early << ":" << mins_early << ":" << secs_early<<" latest landing window
     cout<<endl;
     cout<<endl;
     }
    }

    cout<<"The schedule time of plane 1 is booked at: "<< hour << ":" << minute << ":" << secs<<endl;

    for(int i = 1;i<total_planes;i++){
     cout<<"Enter the scheduled time of plane "<<i+1<<" in {hrs -> mins ->secs}: "<<endl;
     int hrs,minutes,secs;
     cin>>hrs>>minutes>>secs;
     if(hrs >= 0 || hrs <= 23){
       if(minutes >= 0 || minutes <= 59){
         if(secs >= 0 || secs <= 59){
             cout<<"The entered values are "<<hrs<<":"<<minutes<<":"<<secs<<endl;
         }
         else{
             cout<<"wrong seconds entered!! (Enter value in [0,60])";
         }
```

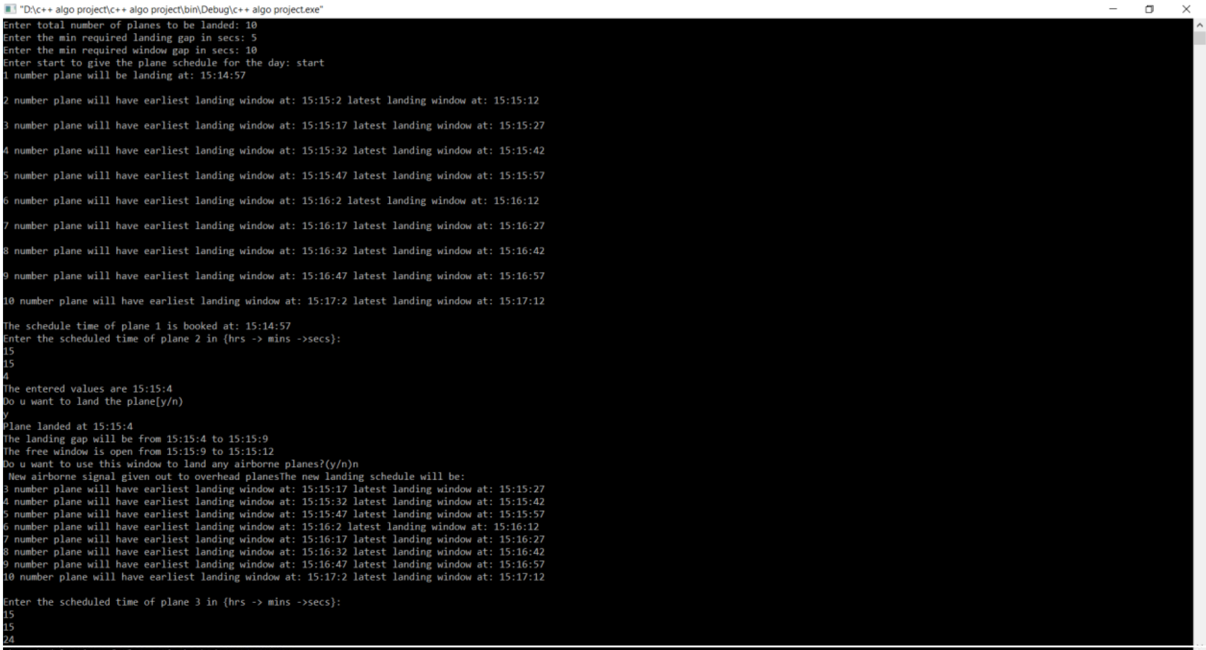
```cpp
                cout<<"wrong seconds entered!! (Enter value in [0,60])";
            }
        }
        else{
            cout<<"wrong minutes entered!! (Enter value in [0,60])";
        }
    }
    else{
        cout<<"wrong hours entered!! (Enter value in [0,23])";
    }

    if(hrs >= mainArray[i][0] && hrs <= mainArray[i][3]){
//              cout<<hrs<<endl;
        if(minutes >= mainArray[i][1] && minutes <= mainArray[i][4]){
//              cout<<minutes<<endl;
            if(secs >= mainArray[i][2] && secs <= mainArray[i][5]){
//                  cout<<secs<<endl;
                cout<<"Do u want to land the plane??(y/n)"<<endl;
                char ans;
                cin>>ans;
                if(ans == 'y'){

                cout<<"Plane landed at "<<hrs<<":"<<minutes<<":"<<secs<<endl;
                int delayed_secs = secs + 5;
                int delayed_minutes = minutes;
                int delayed_hrs = hrs;

                if(delayed_secs>=60){
                    delayed_secs = delayed_secs - 60;
                    delayed_minutes += 1;
                }
                if(delayed_minutes >= 60){
                    delayed_minutes = delayed_minutes - 60;
                    delayed_hrs += 1;
                }
                if(delayed_hrs >= 24){
                    delayed_hrs = delayed_hrs - 24;
                }
```


```cpp
                    delayed_hrs = delayed_hrs - 24;
                }
```

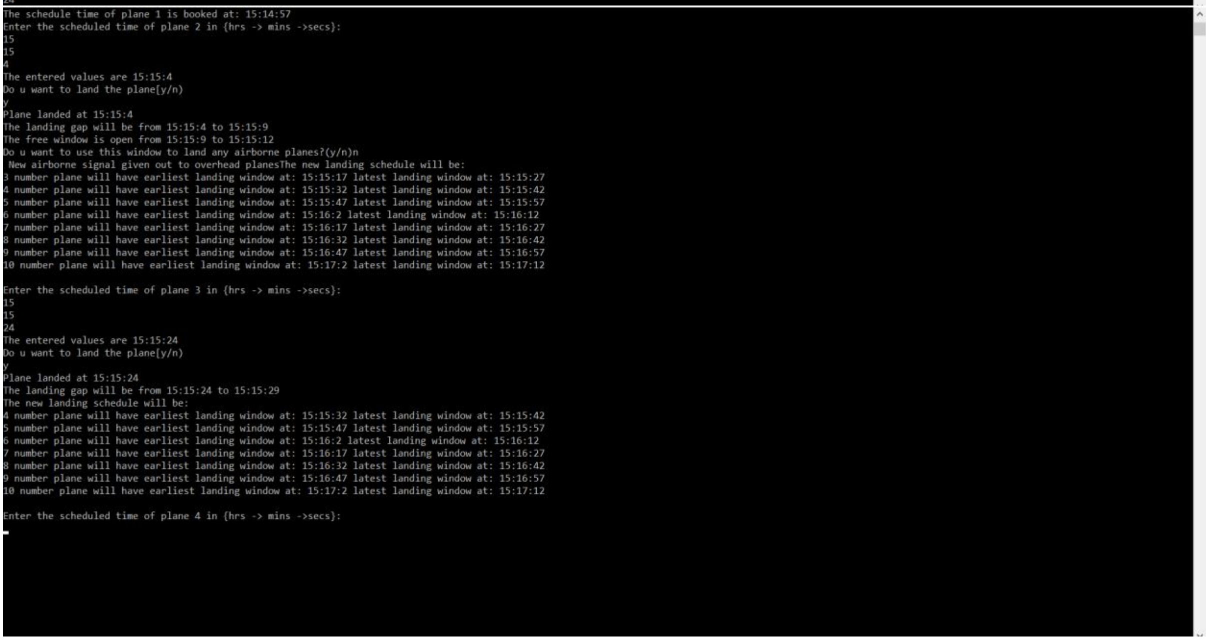

```cpp
                cout<<"The landing gap will be from "<<hrs<<":"<<minutes<<":"<<secs<<" to "<<delayed_hrs<<":"<<delayed_minutes<<":"
                <<delayed_secs<<endl;

                int next_plane_hrs = mainArray[i+1][0];
                int next_plane_minutes = mainArray[i+1][1];
                int next_plane_Secs = mainArray[i+1][2]-5;

                if(next_plane_Secs<0){
                    next_plane_Secs = next_plane_Secs + 60;
                    next_plane_minutes -= 1;
                }
                if(next_plane_minutes < 0){
                    next_plane_minutes = next_plane_minutes + 60;
                    next_plane_hrs += 1;
                }
                if(next_plane_hrs < 0){
                    next_plane_hrs = next_plane_hrs + 24;
                }

                if(next_plane_hrs == delayed_hrs){
                    if(next_plane_minutes == delayed_minutes){
                            if(next_plane_Secs - delayed_secs > 0){
                                cout<<"The free window is open from "<<delayed_hrs<<":"<<delayed_minutes<<":"<<delayed_secs<<" to "
                                <<next_plane_hrs<<":"<<next_plane_minutes<<":"<<next_plane_Secs<<endl;
                                cout<<"Do u want to use this window to land any airborne planes?(y/n)";
                                char ans;
                                cin>>ans;
                                if(ans == 'y'){
                                    cout<<"Airborne plane landed!!";
                                }
                                else if(ans == 'n'){
                                    cout<<" New airborne signal given out to overhead planes";
                                }


                        }
                    }
```

```
                        }
                    }
                }


                }
                else if(ans == 'n'){

                cout<<"Plane window missed from "<<mainArray[i][0]<<":"<<mainArray[i][1]<<":"<<mainArray[i][2]<<" to "<<mainArray[i][3]
                <<":"<<mainArray[i][4]<<":"<<mainArray[i][5]<<endl;
                cout<<"The vague landing gap will be from "<<mainArray[i][3]<<":"<<mainArray[i][4]<<":"<<mainArray[i][5]<<" to "
                <<mainArray[i+1][0]<<":"<<mainArray[i+1][1]<<":"<<mainArray[i+1][2]<<endl;
              }


                cout<<"The new landing schedule will be: "<<endl;

                for(int j = i+1;j<total_planes;j++){
                    cout<<j+1<<" number plane will have earliest landing window at: "<< mainArray[j][0] << ":" << mainArray[j][1] << ":"
                    << mainArray[j][2]<<" latest landing window at: "<< mainArray[j][3] << ":" << mainArray[j][4] << ":" << mainArray[j][5]<<endl;
                }
                cout<<endl;

            }

        }
      }

      //loop ends here
    }

  }
}
```

Output:



```
"D:\c++ algo project\c++ algo project\bin\Debug\c++ algo project.exe"
Enter total number of planes to be landed: 10
Enter the min required landing gap in secs: 5
Enter the min required window gap in secs: 10
Enter start to give the plane schedule for the day: start
1 number plane will be landing at: 15:14:57

2 number plane will have earliest landing window at: 15:15:2 latest landing window at: 15:15:12

3 number plane will have earliest landing window at: 15:15:17 latest landing window at: 15:15:27

4 number plane will have earliest landing window at: 15:15:32 latest landing window at: 15:15:42

5 number plane will have earliest landing window at: 15:15:47 latest landing window at: 15:15:57

6 number plane will have earliest landing window at: 15:16:2 latest landing window at: 15:16:12

7 number plane will have earliest landing window at: 15:16:17 latest landing window at: 15:16:27

8 number plane will have earliest landing window at: 15:16:32 latest landing window at: 15:16:42

9 number plane will have earliest landing window at: 15:16:47 latest landing window at: 15:16:57

10 number plane will have earliest landing window at: 15:17:2 latest landing window at: 15:17:12

The schedule time of plane 1 is booked at: 15:14:57
Enter the scheduled time of plane 2 in {hrs -> mins ->secs}:
15
15
4
The entered values are 15:15:4
Do u want to land the plane[y/n)
y
Plane landed at 15:15:4
The landing gap will be from 15:15:4 to 15:15:9
The free window is open from 15:15:9 to 15:15:12
Do u want to use this window to land any airborne planes?(y/n)n
 New airborne signal given out to overhead planesThe new landing schedule will be:
3 number plane will have earliest landing window at: 15:15:17 latest landing window at: 15:15:27
4 number plane will have earliest landing window at: 15:15:32 latest landing window at: 15:15:42
5 number plane will have earliest landing window at: 15:15:47 latest landing window at: 15:15:57
6 number plane will have earliest landing window at: 15:16:2 latest landing window at: 15:16:12
7 number plane will have earliest landing window at: 15:16:17 latest landing window at: 15:16:27
8 number plane will have earliest landing window at: 15:16:32 latest landing window at: 15:16:42
9 number plane will have earliest landing window at: 15:16:47 latest landing window at: 15:16:57
10 number plane will have earliest landing window at: 15:17:2 latest landing window at: 15:17:12

Enter the scheduled time of plane 3 in {hrs -> mins ->secs}:
15
15
24
```



```
The schedule time of plane 1 is booked at: 15:14:57
Enter the scheduled time of plane 2 in {hrs -> mins ->secs}:
15
15
4
The entered values are 15:15:4
Do u want to land the plane[y/n)
y
Plane landed at 15:15:4
The landing gap will be from 15:15:4 to 15:15:9
The free window is open from 15:15:9 to 15:15:12
Do u want to use this window to land any airborne planes?(y/n)n
 New airborne signal given out to overhead planesThe new landing schedule will be:
3 number plane will have earliest landing window at: 15:15:17 latest landing window at: 15:15:27
4 number plane will have earliest landing window at: 15:15:32 latest landing window at: 15:15:42
5 number plane will have earliest landing window at: 15:15:47 latest landing window at: 15:15:57
6 number plane will have earliest landing window at: 15:16:2 latest landing window at: 15:16:12
7 number plane will have earliest landing window at: 15:16:17 latest landing window at: 15:16:27
8 number plane will have earliest landing window at: 15:16:32 latest landing window at: 15:16:42
9 number plane will have earliest landing window at: 15:16:47 latest landing window at: 15:16:57
10 number plane will have earliest landing window at: 15:17:2 latest landing window at: 15:17:12

Enter the scheduled time of plane 3 in {hrs -> mins ->secs}:
15
15
24
The entered values are 15:15:24
Do u want to land the plane[y/n)
y
Plane landed at 15:15:24
The landing gap will be from 15:15:24 to 15:15:29
The new landing schedule will be:
4 number plane will have earliest landing window at: 15:15:32 latest landing window at: 15:15:42
5 number plane will have earliest landing window at: 15:15:47 latest landing window at: 15:15:57
6 number plane will have earliest landing window at: 15:16:2 latest landing window at: 15:16:12
7 number plane will have earliest landing window at: 15:16:17 latest landing window at: 15:16:27
8 number plane will have earliest landing window at: 15:16:32 latest landing window at: 15:16:42
9 number plane will have earliest landing window at: 15:16:47 latest landing window at: 15:16:57
10 number plane will have earliest landing window at: 15:17:2 latest landing window at: 15:17:12

Enter the scheduled time of plane 4 in {hrs -> mins ->secs}:
```

Time complexity Analysis:

For the single runway case:

Best case: This is the case when all the planes arrive on Target time. No plane gets delayed, comes in early or misses its landing window. The time complexity will be Ω(n).

Average Case: This is the case when not all but some of the planes arrive on Target time. Some planes may get delayed, came in early or missed their landing window. The time complexity will be θ(nlogn).

Worst Case: This is the case when mostly all of the planes do not arrive on Target time. Planes may get delayed, came in early or missed their landing window. The time complexity will be O(n^2). This happens because the schedule rechanged for all of the planes and also new scheduled times have to be given to those planes that missed their landing window. Some if/else checks have to be performed.

For the multiple runway case:

It is difficult to determine the complexity in this case as multiple flights will be landing simultaneously with just collision gap between them so as to avoid collision during simultaneous landing. Overall the complexity will shoot up to O(n^3) since now we also have take in consideration all the planes landing on all of the runways and also along with all of that of sinfle runway. The complexity will fluctuate between O(n) and O(n^3) with a closed bound.

<u>Factors affecting the result of algorithm:</u>

1. Number of planes landing.
2. Number of runways.
3. Number of planes who missed their window.
4. Different landing window gap.
5. Collision avoidance gap.
6. Plane’s scheduled time of arrival.

<u>Results:</u>

N	R	Z_{best}	SCS			BA			PSA		
			Z_{SCS}	T_{run}	G_{best}	Z_{BA}	T_{run}	G_{best}	Z_{PSA}	T_{run}	G_{best}
100	1	5611.70	7298.57	11.9	30.06	6425.95	55.4	14.51	5703.54	14.294	1.637
	2	452.92	478.6	34.2	5.67	700.80	48.7	54.73	**444.1**	10.78	*
	3	75.75	75.75	39	0	142.00	46.6	87.46	75.75	0.868	0
	4	0	0	33.6	0	NA	43.9	n/d	0	0.027	0
150	1	12329.31	17872.56	22.7	44.96	16508.94	92.5	33.90	13515.68	31.411	9.62
	2	1288.73	1390.15	60.8	7.87	1623.15	84.5	25.95	**1203.76**	29.090	*
	3	220.79	240.39	66.8	8.88	653.27	80.3	195.88	**205.21**	19.010	*
	4	34.22	39.94	64.7	16.74	134.27	78.8	292.40	34.22	3.532	0
	5	0	0	60.7	0	NA	76.2	n/d	0	0.0171	0
200	1	12418.32	14647.40	25.6	17.95	14488.45	141.7	16.67	13401.57	27.782	7.92
	2	1540.84	1682.44	95.9	9.19	2134.67	128.7	38.54	**1400.64**	43.77	*
	3	280.82	341.44	102.1	21.59	1095.45	120.3	290.09	**253.15**	11.125	*
	4	54.53	56.04	99.3	2.77	313.25	116.8	474.47	54.53	0.0245	0
	5	0	0	95.6	0	NA	115.8	n/d	0	0.0230	0
250	1	16209.78	19800.24	38.1	22.15	20032.04	201.1	23.58	17346.45	34.93	7.01
	2	1961.39	2330.13	126.6	18.80	2945.61	183.5	50.18	**1753.67**	47.24	*
	3	290.04	340.73	145.4	17.48	864.34	171	198.01	**233.49**	16.271	*
	4	3.49	12.96	144.5	271.63	464.76	168.8	13216.91	**2.44**	1.324	*
	5	0	0	138.6	0	NA	166.2	n/d	0	0.0308	0
500	1	44832.28	46284.84	123.7	3.24	45294.15	585.2	1.03	**43052.04**	52.717	*
	2	5501.96	5706.63	383.6	3.72	7563.54	537.9	37.47	**4593.77**	48.223	*
	3	1108.51	1130.45	456	1.98	3133.64	515.8	182.69	**712.81**	45.168	*
	4	188.46	231.76	441.3	22.98	2425.12	497.7	1186.81	**89.95**	48.6	*
	5	7.35	7.35	442.1	0	1647.02	488.7	22308.44	**0**	0.0554	*
Average				135.5	22.0		197.8	1936.5		20.263	1.091

NA: Results not available.

These are expected results from MATLAB and may differ from actual world results.

CONCLUSION:-

The Aircraft landing problem has mostly been approached using linear programming, meta-heuristic approaches or branch and bound algorithms in the last two decades. This Project was built as an experiment so as to visualize the problem on how it has the capability to perform on real life basis.
Time complexity analysis was done and was done so as to make to minimize it in best way possible.
Obviously in real life scenerios many other factors will also be considered and time complexity will increase. But using various sorting and management algorithms can be used so as to minimize it.

REFERENCES:-

1. https://link.springer.com/content/pdf/10.1134/S0005117919070099.pdf
2. Kumkov, S.I. and Pyatko, S.G., Problem of Detecting and Resolving the Conflict Situations in Automated Air Traffic Control System, Nauchn. Vestn. GosNII “Aeronavigatsiya,” 2013, no. 12, pp. 35–46.
3. 2. Degtyarev, O.V., Sikachev, B.Yu., and Lyashchenko, A.V., Optimal Management of Aircraft Arriving at the Airport, Tr. 9 Mezhd. Simp. (Proc. 9th Int. Simp.), Moscow: GosNIIAS, 2010, pp. 430–435.
4. 3. Aleshin, A.V., Aleshin, V.I., Babaev, N.V., and Kryzhanovskii, G.A., Methods of Simulation of the Processes of Management of the Flow of Arriving and Departing Aircraft at Air Traffic Control in the Area of the Airport, Nauchn. Vest. MGTU GA, 2011, vol. 171, no. 9, pp. 121–128.
5. Aleshin, A.V., Algorithms of Formation of Sequence of Landing Approach When Maneuvering in an Expectation Zone, Vestn. S.-Peterburg. Gos. Univ. GA, 2013, vol. 1(5), pp. 73–84.

6. Furini, F., Kidd, M.P., Persiani, C.A., and Toth, P., Improved Rolling Horizon Approaches to the Aircraft Sequencing, J. Scheduling, 2015, vol. 18, no. 5, pp. 435–447.
7. Ernst, A.T., Krishnamoorthy, M., and Storer, T.H., Heuristic and Exact Algorithms for Scheduling Aircraft Landings, Networks, 1999, vol. 34, no. 3, pp. 229–241.
8. Milan, J., The Flow Management Problem in Air Traffic Control: A Model of Assigning Priorities for Landings at a Congested Airport, Transport. Planning Techn., 1997, vol. 20, no. 2, pp. 131–162.