Here, We explains the Algorithm logic used in Skylents to detect and resolve aircraft conflicts.

1. Input Data

each flight provides:

* Departure and arrival airports
* A route (ordered list of lat/long waypoints)
* Departure time (UTC)
* Aircraft speed (knots)
* Altitude

From this information, each flight defines a planned trajectory in space and time.

2. From routes to Segments

Since we have waypoints between airport departure and arrival, we separate the routes into segment.

each route : $P_0 \longrightarrow P_1 \longrightarrow P_2 \cdots \longrightarrow P_n$

For every segment, we compute:

* Distance
* Travel time (distance/speed)
* Time interval $[t_{start}, t_{end}]$

3. Time Simulation (Δt)

To know where each aircraft is at a given instant, we simulate time using a fixed step: $\Delta t = 15s$.

We iterate: t = earliest departure → end of the longest flight.

At each time step:

- Aircraft that have not yet departed are ignored.
- Aircraft that have already arrived are ignored
- Active aircraft have position computed

The simulation ends when the last segment of the longest flight is completed.

4. Aircraft position at Time t

For each aircraft at time t ($\Delta t.k$):

a. Compute time since departure

b. Compute the traveled distance with that time.

c. Identify the current segment

d. Place position on that segment

Which produces a 4D trajectory point: (lat, long, alt, time)

5. Conflict detection

At each simulation step t, we compare all aircraft present at the same time.

For each of them we compute:

- Horizontale distance (Haversine formula)
- Vertical separation

LOS Rule: { H.d < 5NM and V.S < 2000ft } if one of the condition is false, there is no conflict.

Conflicts depends on the time, not just route intersections.

6. Why Time simulation is mandatory ?

Whitout it:

- Two aircraft could cross the same location at different times: It will incorrectly appear as a conflict

7. Conflict resolution strategy (greedy Approach)

For a given conflict, we generate a finite list of resolution candidates.

Candidates types:

* Delay time * Speed change, respecting the speed limits
* Altitude change ; while respecting constraints: limites

8. Cost fonction:

General form:

Cost = $W\_time \cdot |\Delta t| + W\_alt \cdot |\Delta alt| + W\_speed \cdot |\Delta Speed|$

9. Avoiding New Conflicts

. Local verification window

When testing a candidate solution, we only re-simulate locally. For a conflict occuring in $[t\_start, t\_end]$ we simulate only $[t\_start - \Delta, t\_end + \Delta]$ to minimize side effects

Note: The local verification does not depend on type of the solution candidate.

Conflicts resolution steps:

for each conflict:

1, Generate resolution candidates (delay, alt, speed)

2, Compute cost for each candidate

3. Sort by cost:

4. Test values (finites values) one by one:

- Apply Candidate temporarily
- Re-simulate locally
- Check for Conflits

5. Apply first valide value
6. lock the modification
7. Move to the new conflict.

The process Stops when : ALL conflicts are resolved or No valid resolution exists for a remaining conflict