Relational Time-Space Data Structure To Speed Up Conflict Detection Under Heavy Traffic Conditions

Sergio Ruiz, Miquel A. Piera and Catya A. Zúñiga
Dept. Telecommunication and System Engineering
Universidad Autónoma de Barcelona - Technical Aeronautical Innovation Cluster on Management
Sabadell (Barcelona), Spain
Sergio.ruiz@uab.cat, MiquelA.Piera@uab.cat, CatyaA.Zúñiga@uab.cat

***Abstract*—In this paper, an innovative technique called Relational Time-Space Data Structures is presented for Conflict Detection (CD) between 4D trajectories, improving the performance of the widely-used *pairwise* CD algorithms and also overcoming some of the shortages of previous Spatial Data Structures CD algorithm. WP-E project STREAM aims to coordinate the whole European ATM at strategic and tactical levels, which means (particularly at strategic level) having to process big amounts of trajectories under heavy traffic conditions. This new and efficient CD algorithm may contribute to achieve some of the targets of the STREAM project.** *(Abstract)*

Keywords-component; ATM, 4D Trajectories, Conflict Detection, MTCD, Spatial Data Structure, linear complexity algorithm

I. INTRODUCTION

One of the most important challenges of SESAR with respect the current ATM is the introduction of the Trajectory Based Operations (TBOs), which implies the use of 4D trajectories (trajectories defined in the 3 spatial dimensions together with a time-stamp), also known as Business Trajectories (BT) in the SESAR's terminology for civil flights [1], [2].

In SESAR, Conflict Detection and Resolution (CD&R) systems are decision support tools that use TBOs concepts in order to help controllers managing air traffic flows at *tactical level* (within a foreseen time-window of 20-30 minutes) and at *strategic level* (generally, several hours in advance), providing with information about possible conflicts (understanding a conflict as a loss of due separation between two or more aircrafts [3]) and providing if possible with a new set of conflict-free trajectories [4].

In general, conflict detection and resolution algorithms implemented in CD&R systems can be designed separately as two different but coordinated subsystems, one of them in charge of the conflict detection (CD), and another one in charge of the conflict resolution (CR). Both subsystems can be classified according to the way they handle detection/resolution when multiple conflicts among two or more trajectories happen.

It is referred as a *pairwise strategy* when the algorithms detect/solve conflicts by considering the minimum safety distances between each pair of trajectories, or it is referred as a *global strategy* when the entire traffic situation is examined simultaneously [5].

The STREAM project, a SESAR WP-E project in progress, aims to fill the current existing gap between the strategic and the tactical planning in the ATM, by designing CD&R tools that re-organize the air traffic (i.e. the flight plans) at strategic level (thus, diminishing the amount of conflicts at tactical level), while generating useful network information in order to improve the process of decision taking when conflicts appear at tactical level.

Currently, most CD algorithms that are implemented in operational applications (CTAS, FASTI, iFACTS, ERATO or VAFORIT, among others) are mainly based on pairwise strategies [6–8]. Automated CR tools are currently under development, but early operational applications are also based on pairwise strategies [9], [10].

Unfortunately, it is well known that CD algorithms, when they are based on pairwise strategies, have a temporal complexity of quadratic order, $O(n^2)$ [11]. It is also broadly accepted that CR algorithms, when based on pairwise strategies, belong to a set of non-polynomial problems (NP-Hard) having at least a temporal complexity of exponential order, $O(2^n)$ [4].

Taking into account that it is forecasted an increment of the current air traffic flows in a factor of x2 by 2030 [12], [13] and, on the other hand, that the new ATM air sectors (FABs) will become bigger than the current ones [14], it can be expected that the amount of trajectories to be processed by CD&R systems will increase considerably in next decades.

Thus, more efficient algorithms are needed to achieve the targets of STREAM, since in these incoming scenarios with such a high-density of traffic the computational performance of

CD&R systems based on pairwise strategies could be dramatically mitigated because of the quadratic-growth complexity of their CD algorithms.

Within the framework of the ATLANTIDA project (leaded by Boeing) [15], [16], an efficient CD algorithm which is based on Spatial Data Structures (SDS) (and that do not use pairwise strategies) was developed, showing excellent computational efficiency in the simulation tests. A formal proof of the algorithm having linear temporal complexity, O(n), can be found in [17].

Nevertheless, this SDS-based algorithm also presented some shortages in the scalability of the algorithm due to a huge growth of the required computer-memory, and also due to the limitation of being only a "spatial" data structure (i.e., limited to the representation of 3D spaces).

In this paper is presented an innovative technique that has been named Relational SDS (RSDS) and that allows to reduce the growth-rate of memory of the original SDS algorithm approximately in a 98% whilst keeping the computational time performance (and the rest of the advantages) of the original algorithm.

Moreover, a new concept of data structure is also presented, the Time-Space Data Structure (TSDS), which adds a fourth dimension to the original concept of the SDS.

Early results using synthetic traffic are promising, opening the possibility to efficiently detect conflicts taking into consideration the whole European ATM and analyzing periods of several hours at strategic level, all at the cost of a reasonable amount of memory and processing-time.

RSDSs and TSDSs will be explored under the WP-E project STREAM, since this project pretends to manage the air traffic of the whole European ATM coordinating the strategic and tactical phases of the planning.

II. STATE OF THE ART

A. *PAIRWISE CD ALGORITHMS*

Pairwise CD algorithms basically consist on distance calculations between each different pair of 4D trajectories, or more specifically, consist on distance calculations between the point-mass positions occupied by the aircrafts in a given time. In [11], a well-known pairwise CD algorithm, the CTAS system, was properly described, taking in this case a time-interval between samples equal to 10 seconds.

Fig. 1 shows an example of two pairwise-processed trajectories. It can be observed that points A and C, corresponding to trajectories of aircrafts A1 and A2 respectively, are in conflict. Note that points B and C are geographically closer than points A and C, however, as they belong to different time instants, no conflict happens (conflict occurs in 4 dimensions). Dotted lines indicate the 4D coordinates which belong to the same time instant.

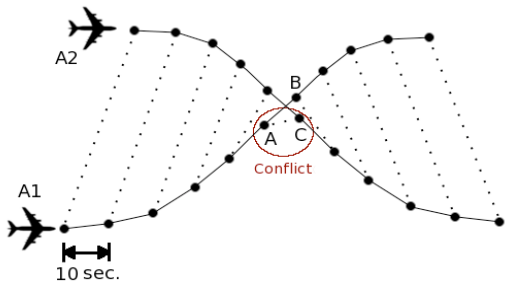

Figure 1. Representation of a pairwise algorithm.

B. *SDS-BASED CD ALGORITHM*

A SDS (Spatial Data Structure) is a database that represents a spatial region (e.g., an air sector) by using individual memory positions to represent each of the discrete (3D) coordinates of the sector. Such memory positions are sorted in a way that, given a certain coordinate, the information stored inside the SDS (associated with such a coordinate) is easily recoverable applying simple mathematical formulas [15], [16].

Fig. 2 shows a graphical representation of a SDS. In particular, the SDS can be thought as a mesh of discrete points distributed along the space region that is being used in the conflict detection process. Note that inside this three-dimensional SDS (the cube of the figure) there is a discretized 4D trajectory (different 3D positions of an aircraft in different discrete time steps).

Granularity or *resolution* of a SDS is the distance between discrete points of the SDS. To determine the optimal separation between SDS points is not an easy matter, and there is no a general method to do that. Factors as the size of the physical airspace to model, the size of the objects to be stored in the database, the speed at what these objects move, the quantity of memory available in the computer, and the speed of execution of the algorithms, among other factors, should be considered to determine the granularity of the SDS [15], [16]. Note that the excess of resolution may lead to a loss of computer performance as well as to an inoperable amount of memory requirements, whereas a lack of resolution may lead to lose some important objects of the space.

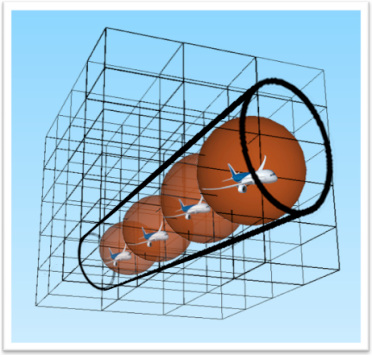

Figure 2. SDS conceptual representation

SDSs are highly configurable and they allow to be set for its use in different applications. For instance, find 2 different conflict detection algorithms using SDSs in the following already-tested applications:

1) Time-distance CD algorithm based on a SDS

Regarding to the targets defined for the SESAR Service Level 5 [14], a SDS-based CD algorithm that checks time-distance separations between aircrafts according to the turbulences generated by other aircrafts (wake vortex) was described in [15], [16], [18]. The idea behind of this algorithm is similar to take a "4D snapshot" of the scenario in where the aircrafts execute their trajectories and the vortexes are generated. Note that vortexes can be modeled with 4D tubes, with an inner radius and a temporal duration defined according to the dynamic behavior of the turbulence [18–20]. A discretized version of these tubes is built and stored in the SDS, where the conflict detection process is performed.

Note that in such a particular algorithm every discrete point of the SDS is treated as a single resource, i.e. a spatial resource that only can be used by one aircraft at a given time. Thus, those spatial resources that are going to be used need to be reserved by aircrafts during a certain time window: the time window of utilization (time windows depend on the wake vortex generated by the aircraft, typically 120 seconds for Heavy and Medium aircrafts in TMA) [3], [16].

Fig. 3 illustrates an example of a SDS content for this particular CD algorithm, which stores the reservations of resources (spatial discrete cells) booked by aircrafts. Note that a SDS can be conceptually drawn as a table containing as many rows as discrete coordinates exist in the modeled airspace and as many columns as aircrafts/trajectories will be processed. Rows are sorted sequentially to ease the access to the database.

To detect conflicts, at the moment of storing a tube-point the algorithm reads the first column. If its value is empty (i.e. equal to zero) it means that no other aircrafts intend to use such a coordinate, so this spatial resource can be booked without conflict. If the first column is not empty, then the algorithm compares the (explicit or implicit) time windows. If their time windows are overlapping, then a conflict is detected and the CR system is informed. If the time windows are not in conflict, it means that the coordinate might be booked in the following column. In next columns applies sequentially the same procedure.

Coords	Reservation 1			Reservation 2			Reservation 3			Reservation n		
	Aircraft	TWon	TWoff	Aircraft	TWon	TWoff	Aircraft	TWon	TWoff	Aircraft	TWon	TWoff
(0,0,0)	0	0	0	0	0	0	0	0	0	0	0	0
(0,0,1)	0	0	0	0	0	0	0	0	0	0	0	0
(0,0,2)	15	50	170	0	0	0	0	0	0	0	0	0
(0,0,3)	0	0	0	4	76	196	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
(0,1,0)	8	99	219	0	0	0	0	0	0	0	0	0
(0,1,1)	**3**	**34**	**154**	**4**	**32**	**152**	7	879	999	0	0	0
(0,1,2)	0	0	0	0	0	0	0	0	0	0	0	0
(0,1,3)	0	0	0	0	0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
(0,2,0)	6	565	685	15	233	353	0	0	0	0	0	0
(0,2,1)	0	0	0	0	0	0	0	0	0	0	0	0
(0,2,2)	76	12320	12440	78	800	920	0	0	0	0	0	0
(0,2,3)	0	0	0	0	0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
(1,0,0)	0	0	0	0	0	0	0	0	0	0	0	0
(1,0,1)	**4**	**590**	**710**	15	88	208	**19**	**680**	**800**	0	0	0
(1,0,2)	1	79	199	2	800	920	3	550	670	n	10098	10118
(1,0,3)	0	0	0	0	0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 3. SDS content example

A spatial granularity of 100 meters between points of the discrete mesh has been considered, having taken into account the size of the aircrafts and their safety envelope, as well as the aircrafts' speed (generally over 200 m/sec.) and the restrictions imposed by the quantity of RAM memory available. Tubes convexity property has been used to ensure that all the possible conflicts will be detected on the surface of the tubes (important computational time savings are possible by only processing the surface of the tubes).

Simulations of different TMA scenarios have proved excellent performance of the SDS-based CD algorithm [16]. A formal demonstration of the linear temporal complexity O(n) of this algorithm, in contrast with the quadratic temporal complexity $O(n^2)$ of the pairwise-based CD algorithms can be found in [17].

2) Efficient spatial-distance CD algorithm based on a SDS

In [21], [22], an efficient collision detection algorithm used in computer games field can be found, consisting of simple pairwise computations among trajectories, but reducing the amount of these pairwise computations by assuming that each *boid* (i.e. a mobile agent defining a trajectory) only can collide against another boid when both of them are at a certain short distance. Formally, the algorithm uses the fact that collisions can only occur between agents that are geographically correlated. The intent is to keep the characters "pre-sorted" in the SDS, based on their location in space, in a way that it can be quickly found which of them are in a given neighborhood at a given time-step without having to examine the entire population (see Fig.4).

In this case, instead of storing information about the boids that visit certain discrete coordinates of the scenario, the SDS was configured to store the information of the boids located over the surface of the spatial subdivisions of the SDS (the

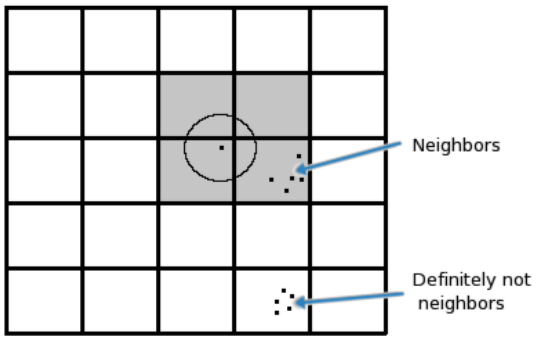

Figure 4. Neighbors search to filter some pairwise comparisons

volume in case of a 3D scenario).

The detection of conflicts is performed by, first, identifying which are the neighbors of each boid at each time step (which is also called spatial prune) and, later, by checking the spatial distances among those still-remaining pairs of trajectories.

Some excellent results of the combined pairwise-SDS algorithm were found in [18], although this algorithm need to be reconfigured for the application to the detection of strategic and tactical conflicts in the European airspace.

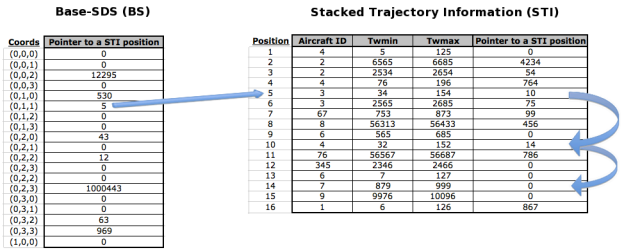

Figure 5. Relational SDS architecture

III. RELATIONAL SDS FOR REDUCING THE MEMORY REQUIREMENTS

One of the most important shortages observed in the original SDS-based CD algorithm was the immense growth rate of the memory required by the algorithm when the number of trajectories to be processed is increased [15], [16], [18]. This problem, given a certain amount of memory, drastically bounds the size of the modeled sector and/or the granularity/resolution of the SDS. Therefore, a more efficient use of the memory is desirable.

Additionally, the memory growth rate of previous SDS designs also limited the possibility of considering aircrafts using a same coordinate more than once (for example during holding trajectories).

Such a huge memory growth rate is due to the construction of the structure of the SDS, which has as many columns as bookings are possible for a certain coordinate (i.e., as many columns as trajectories to be processed). So, each time a new trajectory is added to the problem, the SDS increases its positions in an amount equal to the number of rows required to represent the air sector (which usually means a huge amount of memory positions).

However, it happens that most of the memory positions of the SDS are not being used for storing aircraft information (i.e. SDS memory positions are set to zero), as shown in Figure 3. It is because the set of coordinates belonging to the modeled sector (rows of the SDS) is much greater than the set of coordinates used by all the aircrafts.

Therefore, a more efficient way to manage the information and to minimize the growth of memory of the SDS is by creating different databases, one optimized to store the structure of the SDS (i.e., creating the memory positions that models the airspace), and another one optimized to store the information of the trajectories. The content information of those databases can relate each other through pointers (i.e. identification fields), just like in relational databases.

To implement a RSDS, 2 different databases are required (see Figure 5). The Base SDS (BS) is a database built similarly to the one of the original SDS-based CD algorithm, with one row for each discrete coordinates of the airspace, but with only 1 column (usually occupying 32 bits) instead of n columns for n trajectories (32*n bits). The content of this unique column may be zero or may store a pointer to a position of the second database.

The second database, named Stacked Trajectory Information (STI), will store all the coordinates going to be used by all the trajectories. The particularity of this database is that it stores all the information about trajectories in a stack (in FIFO order), which allows optimizing the storage of the information because no empty memory positions are present at this database (saving lot of memory with respect the non-relational SDS).

The structure of the STI is in general configurable regarding the requirements of the CD and CR algorithms, but always requires a column used to store a pointer to another STI position (set to zero if no pointer is stored). In the example shown in Fig. 5 the STI consist of 4 columns: 3 of them used to store the information of a booking, just containing the same information as in the non-relational SDS (in this case the *id* of the aircraft, the *time window begin* and the *time window end*). The 4th column allows storing a pointer to another STI position, if necessary.

Note that the same information is stored in the SDS of Fig. 3 and in the RSDS of Fig. 5 with regards to the reservations made over the coordinate (0,1,1). If the pointer to STI is not zero (as in the example of Figure 5), it means that at least one booking was done for this coordinate. The value of the pointer indicates the position of the STI where the previous booking is stored (in the example, for coordinate (0,1,1) a pointer is stored to position 5 of STI). By accessing this position in the STI, it is possible to check if there is a conflict between the current and the previous booking. Once determined whether there is a conflict with such a previous booking, the fourth column of the STI current position is checked to search if another pointer to STI is present. If the pointer is set to zero, it means that no more previous booking for such a coordinate exists, so the current booking can be stored in the last free position of the STI and a pointer to that position is stored in the fourth column of the current position of the STI. If the pointer is not zero, it sequentially proceeds with the same algorithm until a free position is found to complete the booking (in the example, positions 10 and 14 has to be sequentially checked).

The amount of memory positions required for the construction of the BS is only related with the size of the

airspace sector to model, so its size does not increase with the amount of trajectories considered in the problem. On the other hand, the amount of memory positions needed for the STI is calculated by NxLxC, being N the number of trajectories to be processed, L the average amount of time-steps per trajectory and C the amount of columns to store bookings and pointers. Thus, the total memory space needed to store the STI grows at a linear rate with respect the amount of trajectories n, what means a spatial complexity of linear order, O(n).

Note that, since only the information being useful for detecting (and solving) conflicts is stored in the RSDS, the STI will not contain too much empty memory positions. This efficient use of the memory has implied important reductions in the use of memory, needing about a 95-99% less memory (in most practical cases) than using non-relational SDS for the same scenario.

Moreover, the characteristic of the SDS of storing the state space of the problem is still available, being possible to extract and summarize useful information about the state-space as feed for new CR algorithms that may take advantage of the state-space exploration in order to find efficient and optimal conflict-free trajectories. CR algorithms based on Coloured Petri Nets may be able to explore the state-space generated by the RSDS.

IV. TIME-SPACE DATA STRUCTURES

The kind of SDS introduced in Section II.B presents the ability of storing the information of the trajectories according to the spatial position they occupy within a certain space, allowing posterior spatial queries that reduce the amount of trajectories to compare for detecting conflicts. Nevertheless, when the amount of trajectories willing to use same spatial resources considerably increases (e.g. in most demanded en-route or TMA sectors), the benefits of using SDS decreases. It is due to the fact that making comparisons with previous trajectories has a computational cost.

By adding the 4th dimension to the structure of the SDS, i.e. the temporal dimension, it is possible to reduce the cost of comparing with previous trajectories since the reservations that where made for the same spatial resources but booked for different times are treated as belonging to different time-space regions. Thus, the reduction of the pairwise comparisons is done by time-spatial queries, which is more powerful than only using spatial queries.

The new data structure can be named as Time-Space Data Structure (TSDS). Conceptually, a TSDS can be thought as a set of T different SDSs, one for each discrete portion of time (see Fig. 6). Note that the discrete portions of time do not necessarily must be thought as time-instants, since they could be also defined as time-windows (for instance, Fig. 6 is showing a set of T different SDSs, each one storing 4 time-steps of different 4D trajectories living in different time-windows). The amount of discrete portions of time and the order of the temporal dimension define the granularity of the temporal dimension of the RSDS.

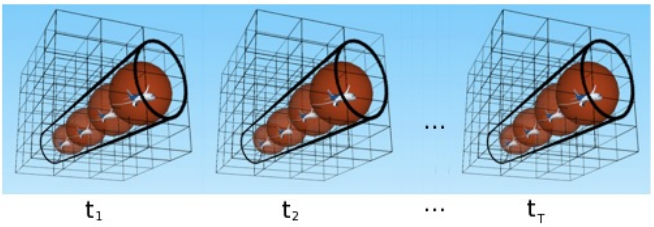

Figure 6. TSDS conceptual representation

The logical structure of the TSDS is similar to the one of the SDS shown in Fig. 2, i.e. each 4D coordinate is represented by a single row of a table, sorted in a way that it is easy to access to the stored content. The method to easily access the content given a coordinate was presented in [15], and here it is extended to take into account the 4th dimension:

$$SDSpos = x*Y*Z*T + y*Z*T + z*T + t \quad (1)$$

Where SDSpos is an univocal memory position inside the TSDS that stores the information relative to the given 4D coordinate (x,y,z,t), and X, Y, Z and T are the total amount of different discrete values that the variables x, y, z, t of a certain 4D coordinate can adopt, according to the order (i.e. size) of each respective dimension.

The combination of the concepts of the RSDS and TSDS is also possible (RTSDS), reducing the memory needs to support the TSDS.

V. NEW CD ALGORITHM USING RTSDS TO REDUCE THE AMOUNT OF PAIRWISE COMPARISONS

In order to build a strategic CD tool for the STREAM project it has been tested the following configuration using the above concepts of RTSDS: the granularity of the RTSDS has been set with bins of 20Km x 20Km x 600m. These dimensions attend to be the double of the minimum safety separation defined in the current ATM [3], which are defined with 5NM (~9.3Km) in the horizontal plane and 1000ft (~300m) in the vertical plane (typical values). The temporal dimension has been set with a resolution of 1 second, since the second is the same time unit used to discretize the 4D trajectories of the aircrafts and it eases the construction and manipulation of the RTSDS.

The definition of 20Km x 20Km for the horizontal-plane grid of the RTSDS allows reducing the granularity of the BS, thus optimizing the usage of memory, whilst at the same time it optimizes the time needed for searching neighbors (smaller bins would require searching for neighbors in more bins) and also reduces the amount of trajectories to compare in pairs (bigger bins means a less powerful “trajectory pruning”). As seen in Fig. 7, with a 20Km x 20Km bin-size it is geometrically ensured that only 4 bins of the horizontal plane needs to be accessed to complete the search for neighbor aircrafts. Similarly occurs in the vertical plane using a bin-size of 600m: only 2 bins need to be accessed to ensure the detection of any

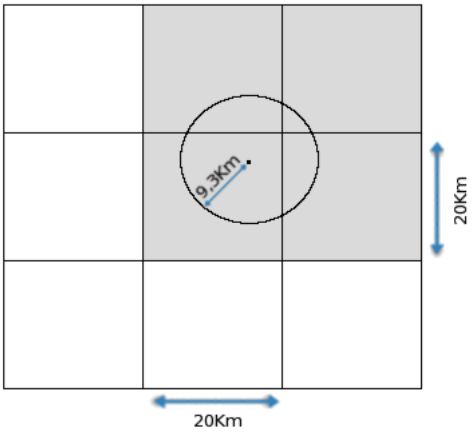

Figure 8. Neighborhood defined by geometrical arguments

conflict. Therefore, in total, the algorithm checks 4x2=8 adjacent bins, looking for neighbors at each time-step of a given trajectory.

If a neighbor aircraft is found inside the neighborhood formed by these 8 bins, and in the same time-instant, since neighborhoods are time-spatial regions when using TSDS, then a direct distance comparison between the positions of the 2 aircrafts is performed to check if they are in conflict.

As an example of the memory requirements of an RTSDS, let consider the modeling of an airspace sector of 5.000x5.000Km2 of planar surface (for example, to cover the whole European ATM), with 20 flight levels (6000m.), and let consider a strategic look ahead for conflict detection of 5 hours with temporal resolution of 1 second. Then, the memory size occupied by the BS is (consider 4 bytes per row):

$$X=Y=5000Km/20Km=250 \quad (2)$$

$$Z=6000m/600m=10 \quad (3)$$

$$T=5h*3600s/h=18000 \quad (4)$$

$$BS=X*Y*Z*T*4=250*250*10*18000*4=45GB \quad (5)$$

Let consider a maximum of 30.000 different trajectories living within the 5 hours look-ahead of the scenario, with average flight duration of 2 hours (and resolution of 1 second). Then, the STI will occupy (consider 8 bytes per row):

$$STI = N*L*8=30000*2*3600*8=1.8GB \quad (6)$$

In total the RTSDS required amount of memory would occupy less than 47GB, amount that could be supported with the internal RAM memory of a computer, instead of using external hard drives that are much more slower at reading and writing the information.

Figure 7. Simulation scenario (100 trajectories)

VI. PERFORMANCE ANALYSIS

A set of different scenarios has been generated with the purpose of measuring the performance of the RTSDS algorithm. An airspace sector with dimensions of 400x400Km2 and maximum height of 30.000 feet has been considered.

Different traffic loads with 35, 100, 200, 400, 500, 800, 1000, 1500, 2000, 3000 and 5000 trajectories were generated with a random entry point located in one of the sides of the surface square and ending in a random exit position of the opposite side. All the trajectories last exactly 30 minutes, which results in an average ground speed of 450 knots and an average covered distance per trajectory of 417,6 Km. Figure 8 shows a visual representation of the 100 trajectories scenario.

ALGORITHM I. PAIRWISE CD ALGORITHM

```
FOR i < N
    j=i+1
    FOR j <= N
        WHILE time < trajectoriesDuration
            P1 = vAircraft[i].vTrajectory[time]
            P2 = vAircraft[j].vTrajectory[time]
            distance2D = calculate2DDistance(P1, P2)
            diffHeight = calculateHeightDistance(P1, P2)
            IF ((distance2D < 9300)&&(diffHeight < 300))
                A conflict has been detected
            END IF
            time++
        END WHILE
    END FOR
END FOR
```

TABLE I. PERFORMANCE RESULTS

n	Pairwise T [ms]	RTSDS T [ms]
35	23	22
100	194	52
200	785	97
400	3153	213
500	4925	287
800	12620	579
1000	19738	837
1500	44447	1675
2000	79074	3200
3000	178008	6837
5000	493802	17513

All the traffic loads have been processed with the RTSDS algorithm and also with a simple pairwise algorithm that does not use any type of SDS or prefilter (see Algorithm I), in order to compare the differences in performance.

The measured performance of both algorithms is shown in Table 1 and in Fig. 9. Clearly, the use of RTSDS presents important advantages over the use of simple pairwise algorithms, presenting processing times up to 28 times faster in the case of 5000 concurrent trajectories.

VII. CONCLUSIONS

STREAM project requires finding efficient conflict detection algorithms in order to process a huge amount of trajectories of the European ATM at strategic and tactical levels.

Pairwise-based CD algorithms, being currently widely used on major real CD systems, may be improved with the use of SDS-based CD algorithms, since these algorithms run at linear time (linear temporal complexity, O(n)) and since they store the state-space of the problem, which may be useful for CR systems.

The main shortage with the use of original SDS-based CD algorithms was the exponential increasing of the memory requirements, which was a hitch to detect conflicts in large sectors and considering a huge amount of 4D trajectories.

An innovative technique called Relational SDS has been presented in this paper, and it has been designed to reduce the huge memory exponential growth of SDSs with up to 95-99% reductions.

Another technique called Time-Space Data Structure has been also presented in this paper, which means a qualitative shift in the concept of SDSs by introducing a temporal dimension in the data structure.

Simulations with several congested scenarios have illustrated that the combination of RSDS and TSDS, called RTSDS, have all the ingredients to become a firm candidate as a CD algorithm solution for the STREAM project.

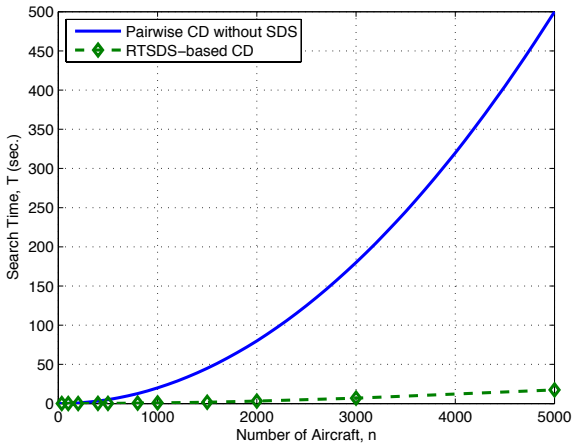

Figure 9. Performance comparison

More research is required to extend the benefits of using RTSDS for CD in STREAM, as for example to find CR algorithms able to explore the state-space generated by the RTSDS. CR algorithms using Coloured Petri Nets are candidates to be integrated with RTSDS and are currently under study.

REFERENCES

[1] Andrew Cook, «The Fourth Dimension: Implementing 4D Aircraft Trajectories», *Navigation News, The magazine of the Royal Institute of Navigation*, nº. January/February, págs. p.17-20, 2010.

[2] SESAR Consortium, «The SESAR Master Plan - SESAR Definition Phase, Deliverable 5». Abr-2008.

[3] ICAO, *Procedures for Air Navigation Services - Air Traffic Management*, 15o ed. (Doc 4444), 2007.

[4] María Prandini y Oliver J. Watkins, «Probabilistic Aircraft Conflict Detection». May-2005.

[5] James K. Kuchar y Lee C. Yang, «A Review of Conflict Detection and Resolution Modeling Methods», *IEEE Transactions on Intelligent Transportation Systems*, vol. Vol.I, nº. 4, págs. pp.179-189, Dic. 2000.

[6] EUROCONTROL, «FASTI Baseline Description». 2007.

[7] EUROCONTROL, «MTCD Algorithms Overview». 2002.

[8] EUROCONTROL, «Specification for Medium-Term Conflict Detection». 2010.

[9] EUROCONTROL, «FASTI Operational Concept». 2007.

[10] M. Kupfer, T.Farley, Y. Chu, y H. Erzberger, «Automated Conflict Resolution - A simulation-based sensitivity study of airspace and demand», in *Proc. 26th International Congress of the Aeronautical Sciences (ICAS)*, 2008.

[11] Isaacson, D.R. y Erzberger, H, «Design of a conflict detection algorithm for the Center/TRACON automation system», in *Digital Avionics Systems Conference*, Irvine, CA, 1997.

[12] EUROCONTROL, «Challenges of Growth 2008». 2008.

[13] EUROCONTROL, «Long-Term Forecast: Flight Movements 2007-2030». 2008.

[14] SESAR Consortium, «European ATM Master Plan». Mar-2009.

[15] Sergio Ruiz y Miquel A. Piera, «Spatial Data Structure based Algorithm for Improving Conflict Detection/Conflict Resolution algorithms», in *Proceedings of Unmaned Aerial Vehicles Conferences (UAV Conferences 2009)*, Reno, Nevada (USA), 2009.

[16] Sergio Ruiz y Miquel A. Piera, «A TMA Simulation Model for Efficient Conflict Detection and resolution Based on Spatial Data Structures», in *In Proc. of WAMS 2010*, Búzios (Rio de Janerio), Brasil, 2010.

[17] S. Ruiz, M. A. Piera, C. A. Zúñiga, y I. del Pozo, «Medium Term Conflict Detection System Using Time-Distance Separations With An Algorithm Of Linear Complexity», *IEEE Transactions on Intelligent Transportation Systems*, unpublished.

[18] S. Ruiz, M. A. Piera, C. A. Zúñiga, y I. del Pozo, «A Medium Term Conflict Detection and Resolution system for TMA Based on Spatial Data Structures and 4D Trajectories», *Elsevier Transportation Research: part C*, unpublished.

[19] O. Desenfans, T. Lonls, G. Winckelmans, y F. Holzapfel, «Description of probabilistic wake predictors adapted to cruise flight», Universite catholique de Louvain (UCL) and DLR, 2007.

[20] Group for Research in Turbulence and Vertical Flows,, «UCL operational tools for predicting aircraft wake vortex transport and decay: The deterministic/Probabilistic wake vortex Models (DVM/PVM) and the WAKE4D platform», v. 2.2, 2010.

[21] Craig W. Reynolds, «Flocks, Herds, and Schools: A distributed Behavioural Model», in *SIGGRAPH 87 Conference Proceedings*, 1987, págs. 25-34.

[22] Craig W. Reynolds, «Big Fast Crowds on PS3», in *Sandbox Symposium*, 2006.