SO6C: Compressed Trajectories in Air Traffic Management

Sebastian Wandelt, Department of Computer Science, Humboldt-Universität zu Berlin, Germany

Xiaoqian Sun, Institute of Air Transportation Systems, German Aerospace Center, Germany

Outline

1) Motivation:

   Why do we need **data science** in **aviation**?

   Why **compression** is important in data science?

2) Standard compression techniques

3) SO6C: Engineering 4D-Trajectories Data Compression

4) Conclusions

Data Science

http://blogs.informatica.com/perspectives/

- Data science is like teenage sex:
  - Everyone talks about it
  - Nobody really knows how to do it
  - Everyone thinks everyone else is doing it
  - Everyone claims they are doing it

Bart Goethals @ 2nd Workshop on Data Science in Aviation

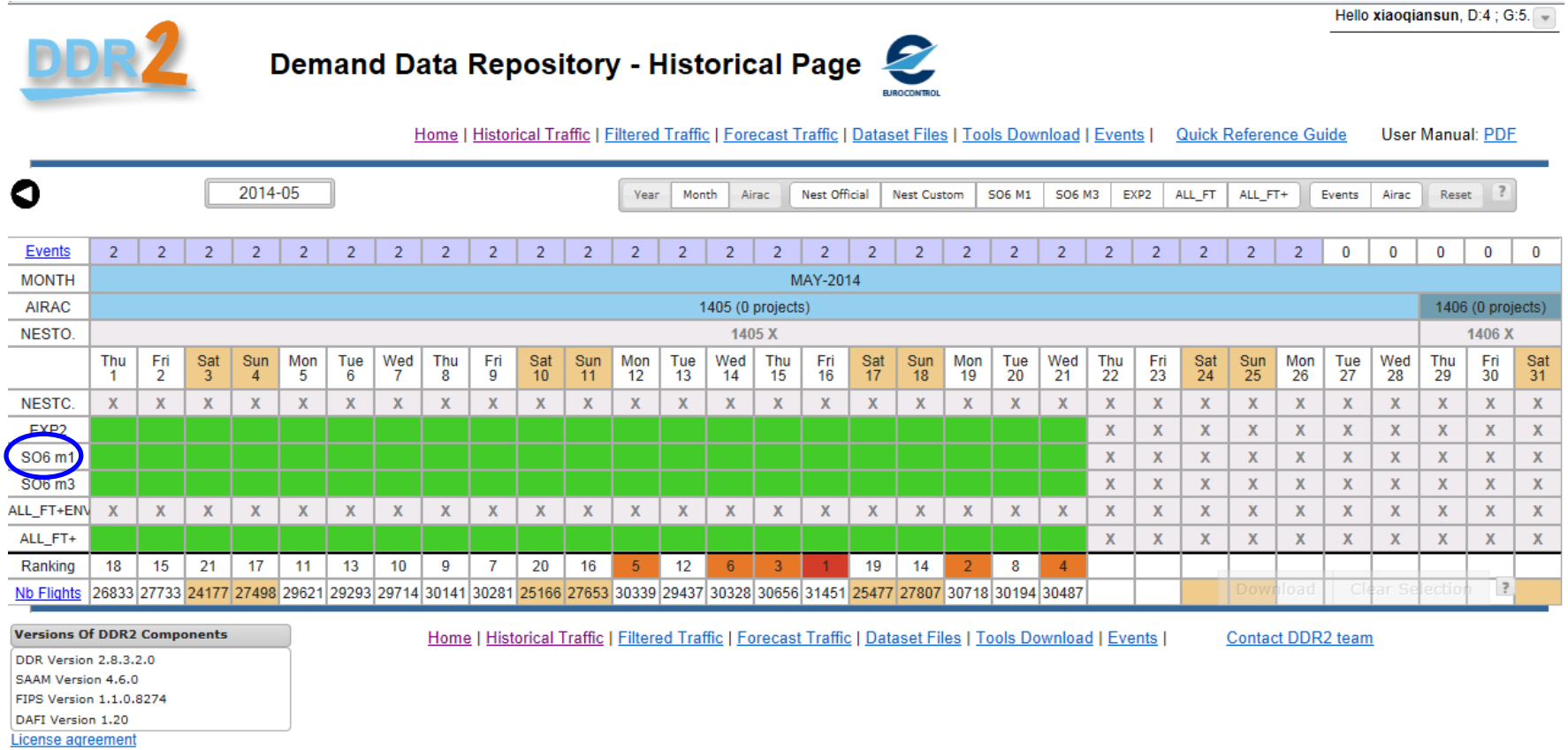
(originally by Dan Ariely, Duke University)

- We do *not* (claim to) do data science, but address a challenging problem towards **data science** in **aviation**!
  - Managing large amounts of **4D-trajectories data**

Major challenge:
Scalable data management in aviation

- Aviation is facing a tremendous increase in (air) traffic data, for example, 4D-trajectories data
- Managing, storing, and analyzing data
  - needs **large disk arrays** for storage
  - and **computing clusters** for analysis
  - Both are **very expensive**
- Data storage and processing in the cloud?
  - Data has to be shipped to the cloud first.
  - Major bottleneck: slow and expensive!

Database DDR2 from Eurocontrol



Hello **xiaoqiansun**, D:4 ; G:5.

DDR2

Demand Data Repository - Historical Page

EUROCONTROL

Home | Historical Traffic | Filtered Traffic | Forecast Traffic | Dataset Files | Tools Download | Events | Quick Reference Guide User Manual: PDF

2014-05

Year | Month | Airac | Nest Official | Nest Custom | SO6 M1 | SO6 M3 | EXP2 | ALL_FT | ALL_FT+ | Events | Airac | Reset

Events	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0	
MONTH	MAY-2014																														
AIRAC	1405 (0 projects)																												1406 (0 projects)		
NESTO.	1405 X																												1406 X		
	Thu 1	Fri 2	Sat 3	Sun 4	Mon 5	Tue 6	Wed 7	Thu 8	Fri 9	Sat 10	Sun 11	Mon 12	Tue 13	Wed 14	Thu 15	Fri 16	Sat 17	Sun 18	Mon 19	Tue 20	Wed 21	Thu 22	Fri 23	Sat 24	Sun 25	Mon 26	Tue 27	Wed 28	Thu 29	Fri 30	Sat 31	
NESTC.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
EXP2																						X	X	X	X	X	X	X	X	X	
SO6 m1																						X	X	X	X	X	X	X	X	X	
SO6 m3																						X	X	X	X	X	X	X	X	X	
ALL_FT+ENV	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
ALL_FT+																						X	X	X	X	X	X	X	X	X	
Ranking	18	15	21	17	11	13	10	9	7	20	16	5	12	6	3	1	19	14	2	8	4										
Nb Flights	26833	27733	24177	27498	29621	29293	29714	30141	30281	25166	27653	30339	29437	30328	30656	31451	25477	27807	30718	30194	30487			Download	Clear Selection						

Versions Of DDR2 Components

DDR Version 2.8.3.2.0
SAAM Version 4.6.0
FIPS Version 1.1.0.8274
DAFI Version 1.20

License agreement

Home | Historical Traffic | Filtered Traffic | Forecast Traffic | Dataset Files | Tools Download | Events | Contact DDR2 team

AIRAC (Aeronautical Information Regulation And Control) **cycle**:

- ICAO defines a series of common dates and an associated standard aeronautical information publication procedure.
- Each year has 13 AIRAC cycles, each AIRAC cycle has 28 days

SO6 m1 file: 4D traffic flight plan trajectories (1)

#	Name	Type	Size	Comment
1	Segment Identifier	Char		Name of first segment _ name of last segment
2	Origin of flight	Char	4	ICAO code of origin airport, e.g. EDDF for Frankfurt
3	Destination of flight	Char	4	ICAO code of destination airport, e.g. EDDH for Hamburg
4	Aircraft type	Char	4	ICAO code of aircraft type, e.g. A388 for Airbus A380
5	Time begin segment	Num	6	Time of entering the segment, format HHMMSS, padded with 0's from the left
6	Time end segment	Num	6	Time of leaving the segment, format HHMMSS, padded with 0's from the left
7	FL begin segment	Num	1-3	Flight level (hundreds of feet) entering the segment, e.g. 250 for 7.620 m
8	FL end segment	Num	1-3	Flight level (hundreds of feet) leaving the segment, e.g. 260 for 7.924,8 m
9	Status	Char	1	0=climb, 1=descent, 2=cruise
10	Callsign	Char		Call sign with ICAO code for airline, e.g. DLH6PH for a Lufthansa flight
11	Date begin segment	Num	6	Date of entering the segment, format HHMMSS, padded with 0's from the left
12	Date end segment	Num	6	Date of leaving the segment, format HHMMSS, padded with 0's from the left
13	Latitude begin segment	Float		Latitude in minute decimale, e.g. 3002 for 50°1'60'' N
14	Longitude begin segment	Float		Longitude in minute decimale, e.g. 514.233 for 8°34'14'' E
15	Latitude end segment	Float		Latitude in minute decimale
16	Longitude end segment	Float		Longitude in minute decimale
17	Flight identifier	Num		Unique identifier for the flight, e.g. 172874110
18	Sequence	Num		Increment at each route segment, e.g. 3 for the third route segment on a flight
19	Segment length	Float		Length of the route segment in nautical miles
20	Segment parity/color	Num		SAAM-specific color encoding (values 0-9)

- The 4D-trajectory of a flight in SO6 consists of 20 fields
  - Route segments: date/time entering segment, flight level, …
  - Meta data: origin, destination, aircraft type, flight identifier, …

SO6 m1 file: 4D traffic flight plan trajectories (2)

```
EDDF_$GHFY,EDDF,EDDH,A319,121700,121716,4,10,0,DLH6PH,131212,131212,3002,514.233333,3001.65,513.616667,172874110,1,0.528604,0
$GHFY_$GHFZ,EDDF,EDDH,A319,121716,121847,10,50,0,DLH6PH,131212,131212,3001.65,513.616667,2998.816667,508.666667,172874110,2,4.260279,0
$GHFZ_ROXAP,EDDF,EDDH,A319,121847,122007,50,93,0,DLH6PH,131212,131212,2998.816667,508.666667,2994.916667,501.85,172874110,3,5.869474,0
ROXAP_$GHFb,EDDF,EDDH,A319,122007,122043,93,110,0,DLH6PH,131212,131212,2994.916667,501.85,2997.883333,499.9,172874110,4,3.2212,0
$GHFb_LISKU,EDDF,EDDH,A319,122043,122136,110,133,0,DLH6PH,131212,131212,2997.883333,499.9,3002.85,496.633333,172874110,5,5.392189,0
LISKU_$GHFc,EDDF,EDDH,A319,122136,122141,133,135,0,DLH6PH,131212,131212,3002.85,496.633333,3003.35,496.9,172874110,6,0.528506,0
$GHFc_$GHFd,EDDF,EDDH,A319,122141,122219,135,135,2,DLH6PH,131212,131212,3003.35,496.9,3006.883333,498.75,172874110,7,3.727401,0
$GHFd_TABUM,EDDF,EDDH,A319,122219,122409,135,177,0,DLH6PH,131212,131212,3006.883333,498.75,3017.45,504.3,172874110,8,11.147823,0
TABUM_LIKSI,EDDF,EDDH,A319,122409,122552,177,210,0,DLH6PH,131212,131212,3017.45,504.3,3028.4,510.266667,172874110,9,11.592171,0
LIKSI_$GHFg,EDDF,EDDH,A319,122552,122704,210,230,0,DLH6PH,131212,131212,3028.4,510.266667,3035.966667,513.9,172874110,10,7.911215,0
$GHFg_LORPA,EDDF,EDDH,A319,122704,122812,230,247,0,DLH6PH,131212,131212,3035.966667,513.9,3043.533333,517.55,172874110,11,7.912495,0
LORPA_MARUN,EDDF,EDDH,A319,122812,122901,247,257,0,DLH6PH,131212,131212,3043.533333,517.55,3049.266667,520.316667,172874110,12,5.994352,0
...
$GHFm_$GHFn,EDDF,EDDH,A319,125039,125148,240,240,2,DLH6PH,131212,131212,3181.166667,603.2,3188.716667,604.9,172874110,23,7.618741,0
$GHFn_$GHFo,EDDF,EDDH,A319,125148,125301,240,210,1,DLH6PH,131212,131212,3188.716667,604.9,3196.8,606.716667,172874110,24,8.156213,0
$GHFo_NOLGO,EDDF,EDDH,A319,125301,125412,210,182,1,DLH6PH,131212,131212,3196.8,606.716667,3203.816667,608.3,172874110,25,7.080069,0
NOLGO_$GHFq,EDDF,EDDH,A319,125412,125443,182,170,1,DLH6PH,131212,131212,3203.816667,608.3,3207.066667,609.05,172874110,26,3.280584,0
$GHFq_$GHFr,EDDF,EDDH,A319,125443,125535,170,150,1,DLH6PH,131212,131212,3207.066667,609.05,3211.933333,610.166667,172874110,27,4.911802,0
$GHFr_$GHFs,EDDF,EDDH,A319,125535,125628,150,130,1,DLH6PH,131212,131212,3211.933333,610.166667,3217.35,611.416667,172874110,28,5.467275,0
$GHFs_HAM,EDDF,EDDH,A319,125628,125710,130,114,1,DLH6PH,131212,131212,3217.35,611.416667,3221.133333,612.3,172874110,29,3.819382,0
HAM_$GHFt,EDDF,EDDH,A319,125710,125817,114,90,1,DLH6PH,131212,131212,3221.133333,612.3,3220.633333,602.483333,172874110,30,5.835624,0
$GHFt_$GHFu,EDDF,EDDH,A319,125817,130128,90,50,1,DLH6PH,131212,131212,3220.633333,602.483333,3219.533333,581.05,172874110,31,12.745985,0
$GHFu_LBE,EDDF,EDDH,A319,130128,130224,50,41,1,DLH6PH,131212,131212,3219.533333,581.05,3219.25,575.7,172874110,32,3.183176,0
LBE_$GHFw,EDDF,EDDH,A319,130224,130444,41,20,1,DLH6PH,131212,131212,3219.25,575.7,3218.416667,589.316667,172874110,33,8.114253,0
$GHFw_$GHFx,EDDF,EDDH,A319,130444,130615,20,10,1,DLH6PH,131212,131212,3218.416667,589.316667,3218.1,594.766667,172874110,34,3.24673,0
$GHFx_EDDH,EDDF,EDDH,A319,130615,130742,10,1,1,DLH6PH,131212,131212,3218.1,594.766667,3217.816667,599.3,172874110,35,2.702977,0
```

- Comma-separated value file
- For a computer: bunch of unstructured text
  - Compressing such representations efficiently is hard!

SO6 m1 file: 4D traffic flight plan trajectories (3)

- Statistics for uncompressed air traffic in SO6:

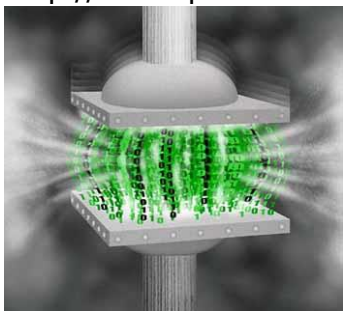
Date	AIRAC cycle	Entries	Uncompressed size (MB)
Thursday, March 08, 2012	0312	1,018,262	141.5
Wednesday, March 14, 2012	0312	991,732	137.7
Thursday, April 05, 2012	0412	1,115,076	155
Thursday, December 12, 2013	1313	1,085,218	150.8
Saturday, December 14, 2013	1313	911,842	126.9

- Storage per day: approx. 142 MB
- Storage per year: approx. 51.8 GB
- Storage per decade: > 0.5 TB
  - And this is only data for Europe!

$\Rightarrow$ How to **store** and **process** such large amounts of data?

Solution: Data compression

http://www.hpcwire.com



- In computer science / information theory:
  - Data compression involves encoding information with less bits than the original representation
- Compression can be either
  - **Lossless**: Original data can be reconstructed completely
  - **Lossy**: Original data can be only reconstructed partially/approx.
- Space-/Time-complexity **tradeoff**
  - Degree of compression VS. amount of loss VS. computational resources required for compression/decompression
- **Compression ratio**:
  - |original input| / |compressed representation|

Standard compression techniques by example

- List of aircraft types as input (1 byte=8 bits)
  - A320, A319, A320, B738, A321, A320, B738, E190, B738, A319
- Uncompressed storage: 10*4 bytes=40 bytes (=320 bits)

1. Naive Bit-manipulation

- Using 8 bits (2^8=256 different states) to encode five different aircraft obviously constitutes a waste of space
- A straight-forward compression technique for these five aircraft types is the encoding with 3 bits (2^3=8 possible states)
  - We assign the codes as follows: A320->000, B738->001, A319->010, A321->011, E190->100
  - Result:

A320	A319	A320	B738	A321	A320	B738	E190	B738	A319
000	010	000	001	011	000	001	100	001	010

  - Only needs 10*3 bits (=30 bits) plus size of data structure which keeps track about mapping aircraft types to bit code

Standard compression techniques by example

Input: A320, A319, A320, B738, A321, A320, B738, E190, B738, A319

2. Dictionary-based compression

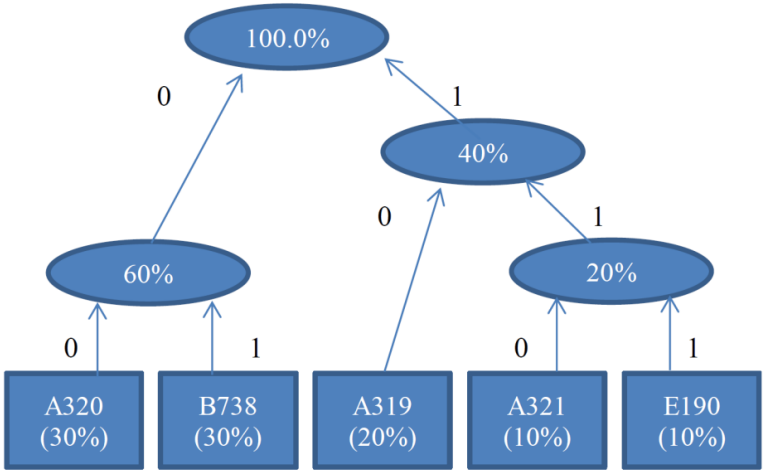
- Keep previously occurred subtexts in a dictionary
- Works well for any kind of (long) text, especially natural language and highly-repetitive text
- Not really applicable for this aircraft example, because the dictionary is larger than the input for short text.

3. Statistical compression

- Create a statistical model of the input
- Shorter codes for frequent items

A320	A319	A320	B738	A321	A320	B738	E190	B738	A319
00	10	00	01	110	00	01	111	01	10

- Only uses 22 (8*2+2*3) bits, instead of 30 bits.



Huffman tree

Standard compression techniques by example

4. Referential compression

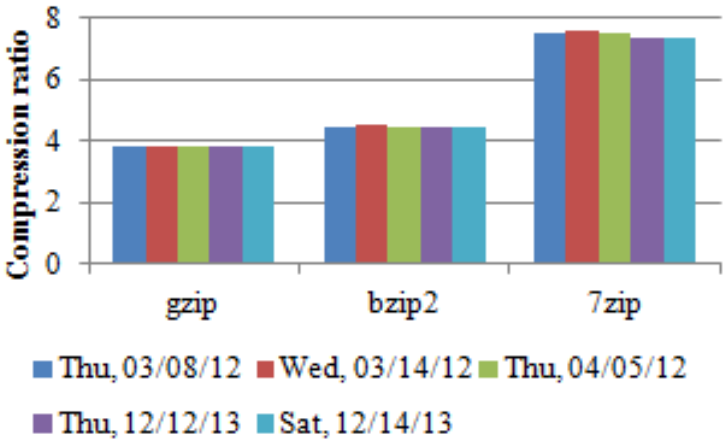
- Encode entries referentially against a previous entry
- Not applicable in our example, but assume a sequence of numbers: S=1,2,3,4,5,6,7,8 ….
- This can be encoded as
  - S(0), S(1)-S(0) , S(2)-S(1) , S(3)-S(2), …, , S(n)-S(n-1)
  - 1,1,1,1,1,1,1,1,1 …
- If the difference is small (here it is fixed at 1), the encoding of the difference entries needs less space than the original sequence

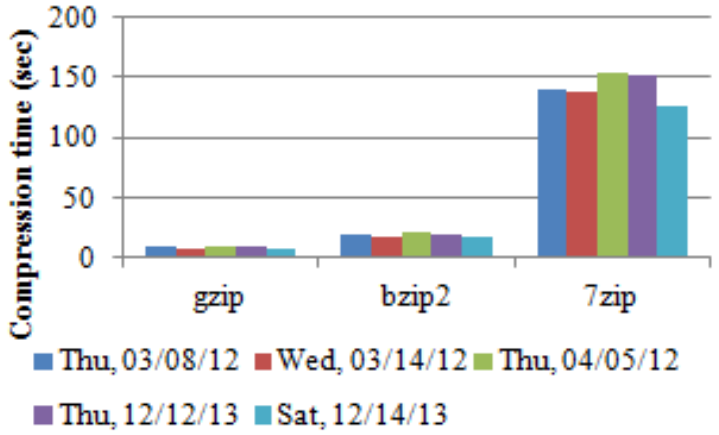
5. Run-length encoding

- Captures frequent number of occurrences of the same element
- E.g. 1,1,1,1,1,1,1,1,1 … is encoded as n*1
- For long sequences, this can save a lot of space

Baseline evaluation

- Three standard compression programs
  - **gzip**: dictionary-based compression
  - **bzip2**: statistical compression
  - **7zip**: combination of dictionary-based and statistical compression
    (Note that 7zip is currently used by Eurocontrol)
- Results:
  - Compression ratio (|input| / |compressed|): 4-8
  - Compression time: few seconds to several minutes

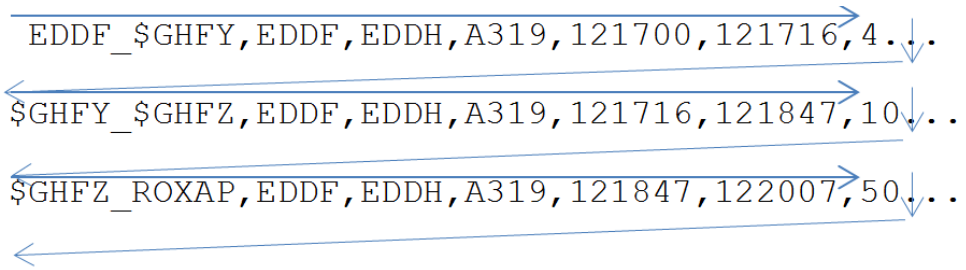




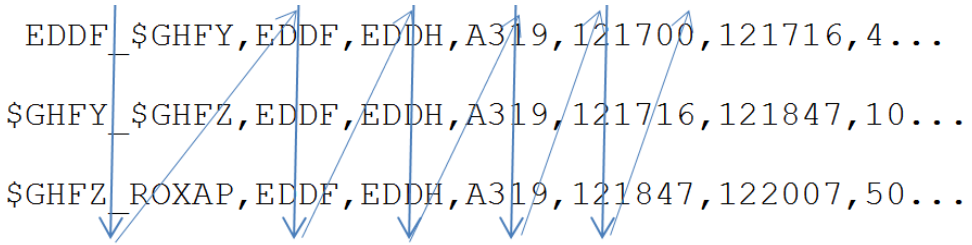
- Can we do better?

Strategy of traversal evaluation

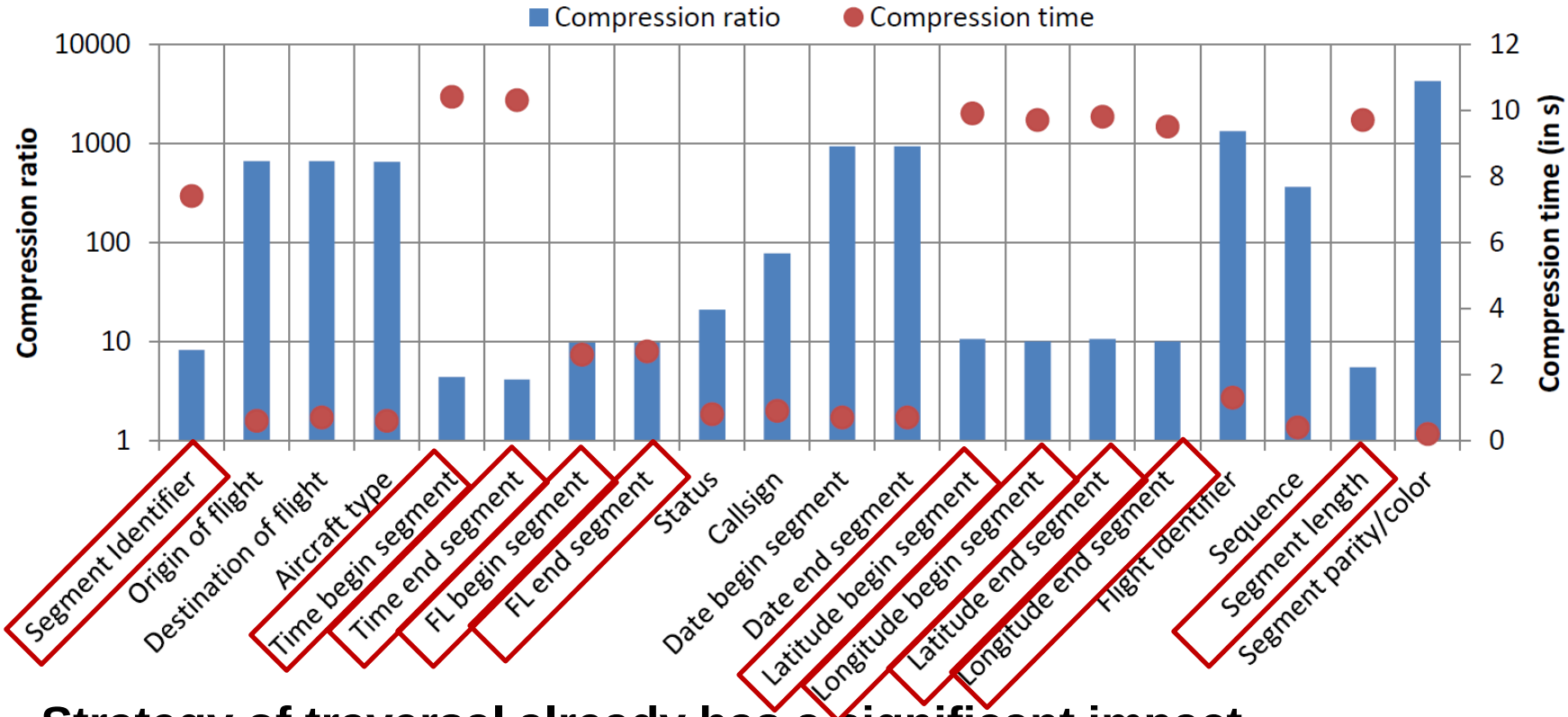
- Standard row-wise compression
  - Mixture of content models
  - Limited window size, i.e. cannot remember items seen much earlier



- How about **column-wise traversal**?
  - Separated content models => Similar types of items stay together
  - Widely used in Bioinformatics



Stream splitting: column-wise traversal



- **Strategy of traversal already has a significant impact**
  - We can compress the SO6 file of a single day
    - Compression ratio of **11.8** (7zip: 7.5) in **88** seconds (7zip: 150 seconds)
  - Stream splitting already identified the **hard-to-compress** fields!
- **Hypothesis**: Further optimization on each field in SO6 should further increase compression ratio

SO6 field 1: Segment identifier

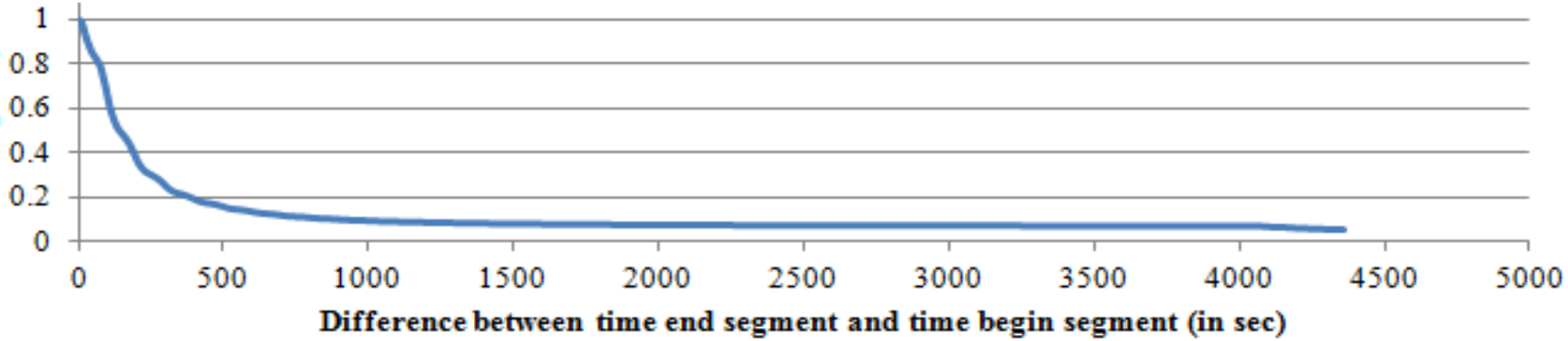
- Unique identifier for the segment at hand
- Concatenation of begin route point and end route point
- Examples
  - EDDF_$GHFY
  - $GHFY_$GHFZ
- Problem: Randomly generated (?) descriptions of temporary places, e.g. $GHFY
- For named locations (airports, fixed route points) we could use a lookup table, but there are too many of these randomly generated segment identifiers (hard to compress)
- Main questions:
  - Are these segment identifiers used?
  - What are their formal semantics?

SO6 field 5: Time begin segment

- Reports the time an aircraft enters the segment
- This field is the same as time end segment for the previous entry of the flight (if a previous entry exists)
  - Storage is redundant in many cases
- We apply **referential compression** of time begin segment to the previous time end segment and often (approx. in 97.6% of all cases) obtain 0
  - 0 can be efficiently encoded using only one bit
- Thus, we often have 0,0,0,0,0,….
  - On top, we apply **run-length encoding**, which further reduces the storage requirements
- Compression ratio is increased significantly
  - from 4 to 72.4

SO6 field 6: Time end segment

- Reports the time an aircraft enters leaves the segment
- Encode referentially against **current** time begin segment
  - Often the difference can be measured in seconds
- Distribution:



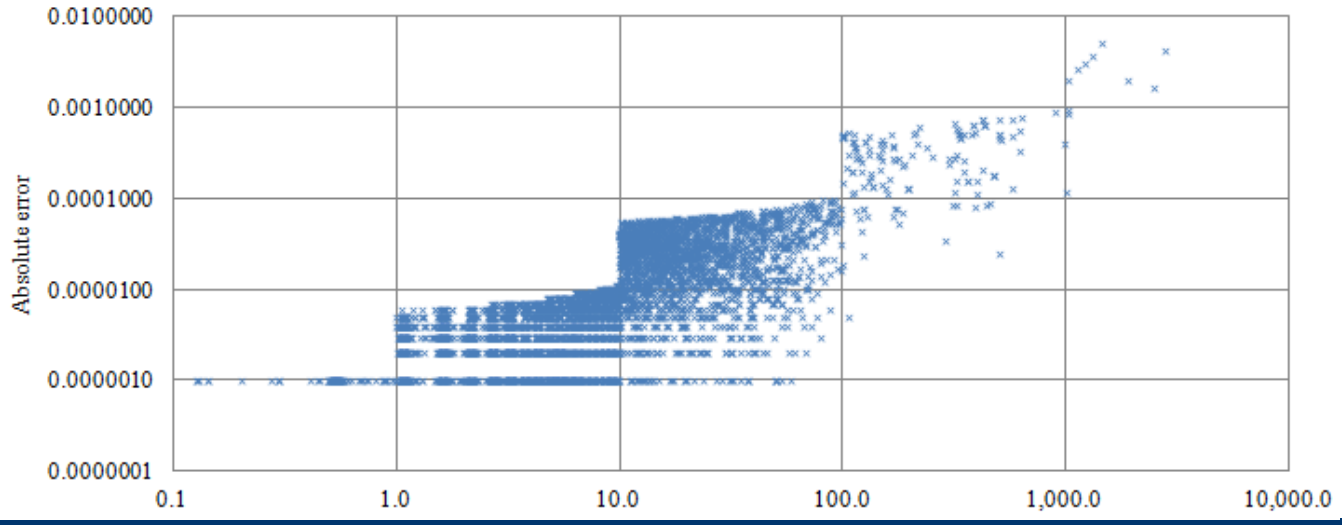
- Small values (we only need exact seconds!) can be encoded efficiently using **a Huffman encoding**
  - Compression ratio increased from 4 to 8

SO6 field 7: FL begin/end segment

- Flight level when entering/leaving a segment
- This field is often the same as FL end segment for the previous entry of the flight (if a previous entry exists)
- FL begin segment is referentially encoded against previous FL end segment
  - Compression ratio increased from 10 to 73.6
- FL end segment is encoded referentially against current FL begin segment
  - No improvement for compression, since the difference is not stable
  - Even taking into account flight status (2=cruise mode), did not help us here

SO6 field 19: Segment length

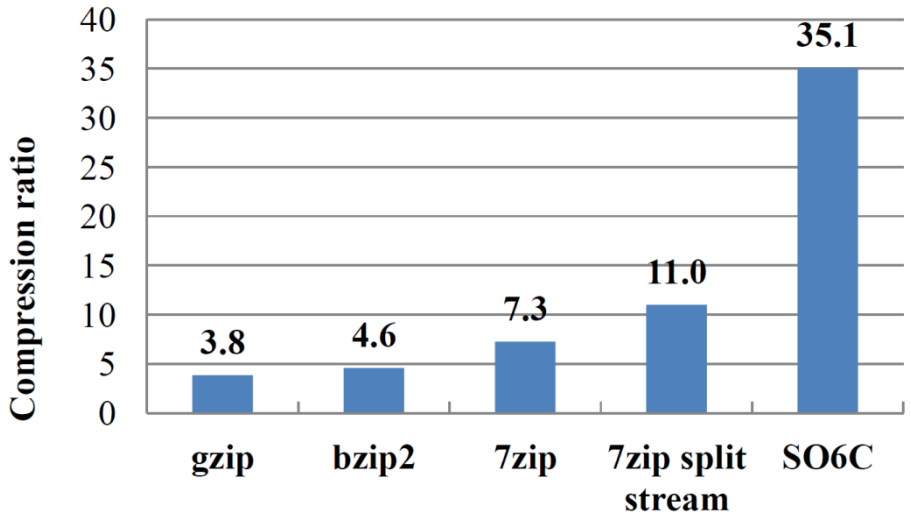
- Length of the current route segment in nautical miles
- A functional dependency great circle distance between lat/lon begin segment and lat/lon end segment
  - Can be computed using Haversine formula
- Redundant?
  - Depends on the use case
  - Computed values are slightly different from values in SO6. Why?
    - Error increases with distance; always smaller than 0.01 nm



SO6C Overall evaluation

Stream	Raw	7zip splitstream	SO6C
1	12.055	1.484	0.360
2	5.176	0.094	0.072
3	5.176	0.094	0.072
4	5.160	0.082	0.057
5	7.246	1.746	0.103
6	7.246	1.762	1.426
7	3.820	0.410	0.052
8	3.820	0.410	0.319
9	2.070	0.129	0.000
10	7.586	0.141	0.142
11	7.246	0.008	0.003
12	7.246	0.008	0.003
13	12.414	1.184	0.360
14	11.773	1.246	0.360
15	12.414	1.180	0.360
16	11.773	1.242	0.360
17	10.355	0.090	0.079
18	2.898	0.078	0.026
19	9.754	1.824	0.000
20	2.070	0.004	0.000
SUM	147.301	13.215	4.155

- We apply our compression techniques to **a set of 91 days** (the first 7 days of each AIRAC cycle in 2013)
- We compare our compression techniques against standard compression methods.



Conclusions

- We propose **a new technique** for **4D-trajectory data compression**
  - We achieve **compression ratios** of **35:1**, compared to 7:1 (7zip) as state-of-the-art
  - Compressing **a single day** takes **14.2 seconds**, compared to 10 (gzip) – 150 (7zip) seconds as state-of-the-art
- We believe that our work is one important step towards **data science** in aviation
- Compression will become very likely a prerequisite for **scalable data processing** in aviation (it is already in Bioinformatics)
- Future work should address **indexing** of 4D-trajectories.
  (What are typical queries for trajectory data?)

Thank you for your attention!

Sebastian Wandelt
Department of Computer Science, Humboldt-University Berlin, 10099 Berlin, Germany
wandelt@informatik.hu-berlin.de

Xiaoqian Sun
Institute of Air Transportation Systems, German Aerospace Center, 21079 Hamburg, Germany
xiaoqian.sun@dlr.de