dataintegrity

January 31, 2022

The Graph Structure of Public Software Development

Antoine Pietri(1), Guillaume Rousseau(2) and Stefano Zacchiroli(3)

1. Inria, Paris, France. antoine.pietri@inria.fr
2. Universit'e de Paris, Paris, France. guillaume.rousseau@u-paris.fr
3. LTCI, Tlcom Paris, Institut Polytechnique de Paris, Paris, France. stefano.zacchiroli@telecom-paris.fr

TODO: Add citation string

1 Replication Package : Quality and Data Integrity

[4]:
```
from pathlib import Path
import matplotlib.pyplot as plt
import numpy as np
import tabulate
from IPython.display import HTML, display
import json

import common


DATASET = Path('../experiments')

def d(p):
    x, y = common.load_text_distribution(p)
    return common.Distribution(x, y, '', '', '')

distributions = {
    'In-degrees': [
        ("Full", d(DATASET / 'inout/full_in.txt')),
        ("Filesystem", d(DATASET / 'inout/dir+cnt_in.txt')),
        ("Commit", d(DATASET / 'inout/rev_in.txt')),
        ("History", d(DATASET / 'inout/rel+rev_in.txt')),
        ("Hosting", d(DATASET / 'inout/ori+snp_in.txt')),
    ],
```

```
    'Out-degrees': [
        ("Full", d(DATASET / 'inout/full_out.txt')),
        ("Filesystem", d(DATASET / 'inout/dir+cnt_out.txt')),
        ("Commit", d(DATASET / 'inout/rev_out.txt')),
        ("History", d(DATASET / 'inout/rel+rev_out.txt')),
        ("Hosting", d(DATASET / 'inout/ori+snp_out.txt')),
    ],
    'Connected components': [
        ("Full", d(DATASET / 'connectedcomponents/full/distribution.txt')),
        ("Filesystem", d(DATASET / 'connectedcomponents/dir+cnt/distribution.
 ↪txt')),
        ("Commit", d(DATASET / 'connectedcomponents/rev/distribution.txt')),
        ("History", d(DATASET / 'connectedcomponents/rel+rev/distribution.txt')),
        ("Hosting", d(DATASET / 'connectedcomponents/ori+snp/distribution.txt')),
    ],
    'Clustering coefficient': [
        ("Full", d(DATASET / 'clusteringcoeff/distribution-full.txt')),
        ("Filesystem", d(DATASET / 'clusteringcoeff/distribution-dircnt.txt')),
        ("Commit", d(DATASET / 'clusteringcoeff/distribution-rev.txt')),
        ("History", d(DATASET / 'clusteringcoeff/distribution-relrev.txt')),
        # ("Hosting", d(DATASET / 'clusteringcoeff/distribution-orisnp.txt')),
    ],
    'Shortest path': [
        ("Filesystem", d(DATASET / 'shortestpath/dir+cnt/distribution.txt')),
        ("Commit", d(DATASET / 'shortestpath/rev/distribution.txt')),
    ]
}
```

1.1 Graph layer statistics

Statistics of the graph layers and their associated distributions, as reported in the article.

[5]:
```
# it can take few minutes to process
headers = ["Algorithm", "Layer", "Number of objects", "Scaling parameter", "X␣
 ↪decades", "Y decades"]
table = []
for algo_name, algo_distributions in distributions.items():
    for name, distribution in algo_distributions:
        row = [
            algo_name,
            name,
            f'{int(np.sum(distribution.y)):,}',
            distribution.fitted_power(),
            np.log10(np.max(distribution.x)),
            np.log10(np.max(distribution.y)),
        ]
```

```
        table.append(row)

display(HTML(tabulate.tabulate(table, headers=headers, tablefmt='html')))
```

```
<IPython.core.display.HTML object>
```

1.2 Data integrity: in and out degrees

This data helps getting an overview of the graph properties and check whether it is consistent to our expectations as a way to perform data integrity checks.

1.2.1 Node and edge statistics of the studied graph corpus.

It corresponds to https://annex.softwareheritage.org/public/dataset/graph/2020-12-15/compressed/ (same as Table 1)

TODO Confirm that these numbers do not come from a calculation based on the distributions but from the raw data, and provide script to generate them from raw data. https://forge.softwareheritage.org/source/swh-dataset/browse/master/swh/dataset/exporters/edges.py$150-230 ? Add extra link

Layer	Node type	Nodes	%
hosting	origins	147 453 557	0.76%
	snapshots	139 832 772	0.72%
history	releases	16 539 537	0.09%
	commits	1 976 476 233	10.22%
filesystem	directories	7 897 590 134	40.86%
	contents	9 152 847 293	47.35%
	Total	19 330 739 526	100%

Layer	Edge type	Edges	%
hosting	origin → snapshot	776 112 709	0.35%
	snapshot → commit	1 358 538 567	0.61%
	snapshot → release	70 0823 546	0.32%
history	release → commit	16 492 908	0.01%
	commit → commit	2 021 009 703	0.91%
	commit → directory	1 971 187 167	0.89%
filesystem	directory → directory	64 584 351 336	29.16%
	directory → commit	792 196 260	0.36%
	directory → content	149 267 317 723	67.39%
	Total	221 488 073 659	100%

[152]:
```
#raw stats from the compress dataset 2021-12-15
# (before any statistical processing)
short2longname={"ori":"origin","snp":"snapshot","rel":"release",
                "rev":"commit","dir":"directory","cnt":"content"}
rawstats={"nodes":
          {
              "origin":147453557,
              "snapshot":139832772,
              "release":16539537,
              "commit":1976476233,
              "directory":7897590134,
              "content":9152847293
          },
          "edges":{
              "origin":{"snapshot":776112709},
              "snapshot":{"commit":1358538567,"release":700823546},
              "release":{"commit":16492908},
              "commit":{"commit":2021009703,"directory":1971187167},
              "directory":{"directory":64584351336,"commit":792196260,"content":
 ↪149267317723}
          }
         }
```

1.2.2 Criteria list

Here are a few examples of criteria that can be checked on the table:

1. The number of nodes computed from the distributions (= the sum of the second column) is always the same in all distributions starting from the same node type. For instance, `dir_in_*` and `dir_out_*` all have the same number of directory nodes which have to be equals to the number of directory nodes in the raw swh dataset (namely 7 897 590 134).
2. The number of edges computed from a source type to a destination type (src_out_dest) equals the number of edges from the raw dataset.
3. The total (or average) in/outdegree of a given object type is consistent when each neighbor type is looked independently and when they are all aggregated together (e.g. the average degree of `dir_out_all` is a weighted average of the average degrees of the `dir_out_{cnt,dir,rev}` distributions).
4. The number of objects with a total indegree of 0 should be small in all types of objects that are supposed to be reachable from the upper layers of the graph.
5. The number of objects with a total outdegree of 0 should be small in specific types of objects that are supposed to reach the lower layers of the graph.
6. Some specific per-layer indegrees are expected to be relatively small compared to the total number of objects (e.g. most revisions do not have an associated release)

[153]:
```
inout_per_type = [
    'cnt_in_dir',
```

```
    'dir_in_all',
    'dir_in_dir',
    'dir_in_rev',
    'dir_out_all',
    'dir_out_cnt',
    'dir_out_dir',
    'dir_out_rev',
    'ori_out_snp',
    'rel_in_snp',
    'rev_in_all',
    'rev_in_dir',
    'rev_in_rel',
    'rev_in_rev',
    'rev_in_snp',
    'rev_out_rev',
    'snp_in_ori',
    'snp_out_all',
    'snp_out_rel',
    'snp_out_rev',
]

headers = ["Node type", "Direction", "Neighbor type", "# Nodes", "# Edges", "Avg␣
 ↪degree", "# (Lowest degree)", "# (Second-lowest)"]
table2 = []
for name in inout_per_type:
    dist = d(DATASET / f'inout/per_type/{name}.txt')
    src, direction, dst = name.split('_')
    row = [
        common.types_verbose[src],
        ("âĘŘ in " if direction == 'in' else "âĘŠ out "),
        common.types_verbose[dst],
        f'{int(np.sum(dist.y)):,}',
        f'{int(np.sum(dist.x * dist.y)):,}',
        np.sum(dist.x * dist.y) / np.sum(dist.y),
        f'{int(dist.y[0]):,} ({int(dist.x[0]):,})',
        f'{int(dist.y[1]):,} ({int(dist.x[1]):,})',
    ]
    table2.append(row)

display(HTML(tabulate.tabulate(table2, headers=headers, tablefmt='html')))
```

```
<IPython.core.display.HTML object>
```

1.2.3 Control scripts

- control script according to criterion 1

[154]:
```
print("Control script according to criterion 1")
print()
DATASET = Path('../experiments')
error=False
for name in inout_per_type:
    dist = d(DATASET / f'inout/per_type/{name}.txt')
    src, direction, dst = name.split('_')
    sy=np.sum(dist.y)
    sxy=np.sum(dist.x*dist.y)
    if dist.x[0]==0:
        s0=dist.y[0]
    else:
        s0=0
    if sy==rawstats["nodes"][short2longname[src]]:
        print(f'{src} {direction} {dst} OK,', end=" ")
    else:
        print()
        print(f'{src} {direction} {dst} ERROR {int(sy):,} {int(sxy):,}')
        error=True
print("end")
print()
if error:
    print("Control status : Failed")
else:
    print("Control status : Successful")
```

```
Control script according to criterion 1

cnt in dir OK, dir in all OK, dir in dir OK, dir in rev OK, dir out all OK, dir
out cnt OK, dir out dir OK, dir out rev OK, ori out snp OK, rel in snp OK, rev
in all OK, rev in dir OK, rev in rel OK, rev in rev OK, rev in snp OK, rev out
rev OK, snp in ori OK, snp out all OK, snp out rel OK, snp out rev OK, end

Control status : Successful
```

- control script according to criterion 2

[155]:
```
print("Control script according to criterion 2")
print()
DATASET = Path('../experiments')

error=False

for name in inout_per_type:
    dist = d(DATASET / f'inout/per_type/{name}.txt')
    src, direction, dst = name.split('_')
    sy=np.sum(dist.y)
    sxy=np.sum(dist.x*dist.y)
```

```
    if dist.x[0]==0:
        s0=dist.y[0]
    else:
        s0=0
    if direction=="out" and dst!="all":
        rs=rawstats["edges"][short2longname[src]][short2longname[dst]]
        if sxy==rs:
            print(f'{src} {direction} {dst} OK')
        else:
            ds=(rs-sxy)/rs
            print(f'{src} {direction} {dst} ERROR {int(sxy):17,} (derived) {rs:
↪17,}(raw) {ds:<}')
            error=True
    if direction=="in" and dst!="all":
        rs=rawstats["edges"][short2longname[dst]][short2longname[src]]
        if sxy==rs:
            print(f'{src} {direction} {dst} OK')
        else:
            ds=(rs-sxy)/rs
            print(f'{src:3} {direction:3} {dst:3} ERROR {int(sxy):17,} (derived)␣
↪{rs:17,}(raw) {ds:<}')
            error=True

if error:
    print("Control status : Failed")
else:
    print("Control status : Successful")
```

```
Control script according to criterion 2

cnt in  dir ERROR   143,786,784,566 (derived)   149,267,317,723(raw)
0.036716229919602335
dir in  dir ERROR    63,229,213,027 (derived)    64,584,351,336(raw)
0.020982455981479086
dir in rev OK
dir out cnt ERROR   143,786,781,408 (derived)   149,267,317,723(raw)
0.03671625107627647
dir out dir ERROR    63,229,213,027 (derived)    64,584,351,336(raw)
0.020982455981479086
dir out rev ERROR       789,473,873 (derived)       792,196,260(raw)
0.0034365057466946387
ori out snp ERROR       189,314,705 (derived)       776,112,709(raw)
0.7560731800875586
rel in  snp ERROR       700,135,072 (derived)       700,823,546(raw)
0.000982378522995573
rev in  dir ERROR       789,473,873 (derived)       792,196,260(raw)
0.0034365057466946387
```

```
rev in rel OK
rev in  rev ERROR     2,019,963,947 (derived)     2,021,009,703(raw)
0.0005174423450059012
rev in  snp ERROR     1,146,176,123 (derived)     1,358,538,567(raw)
0.15631683130568055
rev out rev ERROR     2,019,963,947 (derived)     2,021,009,703(raw)
0.0005174423450059012
snp in  ori ERROR       189,320,602 (derived)       776,112,709(raw)
0.7560655819643329
snp out rel ERROR       700,096,853 (derived)       700,823,546(raw)
0.0010369129350000464
snp out rev ERROR     1,146,176,123 (derived)     1,358,538,567(raw)
0.15631683130568055
Control status : Failed
```

Subject to further investigation, we observe here the deduplication due to compression, which means that the statistics measured are not those of the original graph (see section dedicated to internal threat to validity). TODO(TBC).

- control script according to criterion 3

[156]:
```
print("Control script according to criterion 3")
print()
DATASET = Path('../experiments')

error=False
statsC3={}
for name in inout_per_type:
    dist = d(DATASET / f'inout/per_type/{name}.txt')
    src, direction, dst = name.split('_')
    sy=np.sum(dist.y)
    sxy=np.sum(dist.x*dist.y)
    if dist.x[0]==0:
        s0=dist.y[0]
    else:
        s0=0
    if direction=="out":
        if dst!="all":
            try:
                statsC3[src]["*"]+=sxy
            except:
                try:
                    statsC3[src]["*"]=sxy
                except:
                    statsC3[src]={"*":sxy}
        else:
            try:
                statsC3[src]["all"]=sxy
```

```
            except:
                statsC3[src]={"all":sxy}

for src in statsC3:
    if "all" in statsC3[src]:
        if statsC3[src]["all"]!=statsC3[src]["*"]:
                print(f'{src} OK')
        else:
                print(f'{src}  ERROR {statsC3[src]["all"]:,} {statsC3[src]["*"]:
 ↪,}')
                error=True

if error:
    print("Control status : Failed")
else:
    print("Control status : Successful")
```

```
Control script according to criterion 3

dir OK
snp OK
Control status : Successful
```

- control script according to criterion 4

[157]:
```
print("Control script according to criterion 4")
print()
DATASET = Path('../experiments')

error=False
for name in inout_per_type:
    dist = d(DATASET / f'inout/per_type/{name}.txt')
    src, direction, dst = name.split('_')
    sy=np.sum(dist.y)
    sxy=np.sum(dist.x*dist.y)
    if dist.x[0]==0:
        s0=dist.y[0]
    else:
        s0=0
    if ((dst=="all" and direction!="out") or (direction=="in" and src!="dir" and␣
 ↪src!="rev")) and s0!=0:
    #if (dst=="all" or (direction=="in")) and s0!=0:
        rs=rawstats["nodes"][short2longname[src]]
        ds=s0/rs
        print(f'{src:3} {direction:3} {dst:3} ERROR {int(s0):17,} {ds:<}')
        error=True
print("end")
print()
```

```
if error:
    print("Control status : Failed")
else:
    print("Control status : Successful")
```

```
Control script according to criterion 4

dir in  all ERROR         1,343,830 0.00017015696904992107
rel in  snp ERROR           427,531 0.025849030719541907
rev in  all ERROR        21,591,750 0.010924366121634006
snp in  ori ERROR            53,736 0.0003842875974739312
end

Control status : Failed
```

- control script according to criterion 5

[158]:
```
print("Control script according to criterion 5")
print()
DATASET = Path('../experiments')

error=False
for name in inout_per_type:
    dist = d(DATASET / f'inout/per_type/{name}.txt')
    src, direction, dst = name.split('_')
    sy=np.sum(dist.y)
    sxy=np.sum(dist.x*dist.y)
    if dist.x[0]==0:
        s0=dist.y[0]
    else:
        s0=0
    if ((dst=="all" and direction=="out")) and s0!=0:
    #if (dst=="all" or (direction=="in")) and s0!=0:
        rs=rawstats["nodes"][short2longname[src]]
        ds=s0/rs
        print(f'{src:3} {direction:3} {dst:3} ERROR {int(s0):17,} {ds:<}')
        error=True
print("end")
print()
if error:
    print("Control status : Failed")
else:
    print("Control status : Successful")
```

```
Control script according to criterion 5

dir out all ERROR           557,087 7.053885939226939e-05
snp out all ERROR            43,567 0.0003115650171048601
```

```
end

Control status : Failed
```

- control script according to criterion 6

[159]:
```
print("Control script according to criterion 6")
print()
DATASET = Path('../experiments')

error=False
threshold=0.01 # 5% : threshold value meaning "small compared to the number of␣
 ↪nodes"
for name in ['rev_in_rel']:
    dist = d(DATASET / f'inout/per_type/{name}.txt')
    src, direction, dst = name.split('_')
    sy=np.sum(dist.y)
    sxy=np.sum(dist.x*dist.y)
    if dist.x[0]==0:
        s0=dist.y[0]
    else:
        s0=0
    ds=(sy-s0)/sy
    if ds>threshold:
        print(f'{src:3} {direction:3} {dst:3} ERROR {int(sy-s0):17,} / {int(sy):
 ↪<17,} = {ds:<}')
        error=True
    else:
        print(f'{src:3} {direction:3} {dst:3} OK {int(sy-s0):17,} / {int(sy):
 ↪<17,} = {ds:<} < {threshold} (threshold)')

print("end")
print()
if error:
    print("Control status : Failed")
else:
    print("Control status : Successful")
```

```
Control script according to criterion 6

rev in  rel OK        11,722,969 / 1,976,476,233     = 0.0059312471378450415 <
0.01 (threshold)
end

Control status : Successful
```

1.3 Example of Data integrity failures ecountered during this study

1.3.1 Error in the in/out distributions processing (switch fallthrough bug)

Experiments based on dataset 2020-05-20 and an incorrect computation of the distributionsincorrect computation of the distributions

Control script corresponding to criterion 1 failed

Issue has been investigated and fixed (https://forge.softwareheritage.org/rDGRPH6ef89157db57834ad94607f369

[160]:
```
# ! raw stats from distribution after the bug fix
# and not from the raw stats of the compress graph
#
rawstats20210403={"nodes":
          {
              "origin":108109058,
              "snapshot":121696833,
              "release":14386337,
              "commit":1734773279,
              "directory":6914748995,
              "content":8181993787
          }
         }
print("Control script according to criterion 1")
print()
#DATASET = Path('../experiments/deprecated/20201019/')
DATASET = Path('../experiments/deprecated/20210317/')  # latest before bugfix
#DATASET = Path('../experiments/deprecated/20210403/') # after bugfix
error=False
for name in inout_per_type:
    dist = d(DATASET / f'inout/per_type/{name}.txt')
    src, direction, dst = name.split('_')
    sy=np.sum(dist.y)
    sxy=np.sum(dist.x*dist.y)
    if dist.x[0]==0:
        s0=dist.y[0]
    else:
        s0=0
    rs=rawstats20210403["nodes"][short2longname[src]]
    if sy==rs:
        print(f'{src} {direction} {dst} OK,', end=" ")
    else:
        print(f'{src} {direction} {dst} ERROR {int(sy):,} (derived) {int(rs):,}␣
 ↪(raw)')
        error=True
print("end")
print()
```

```
if error:
    print("Control status : Failed")
else:
    print("Control status : Successful")
```

```
Control script according to criterion 1

cnt in dir ERROR 16,363,987,574 (derived) 8,181,993,787 (raw)
dir in all ERROR 15,096,742,782 (derived) 6,914,748,995 (raw)
dir in dir ERROR 15,096,742,782 (derived) 6,914,748,995 (raw)
dir in rev ERROR 15,096,742,782 (derived) 6,914,748,995 (raw)
dir out all ERROR 15,096,742,782 (derived) 6,914,748,995 (raw)
dir out cnt ERROR 15,096,742,782 (derived) 6,914,748,995 (raw)
dir out dir ERROR 15,096,742,782 (derived) 6,914,748,995 (raw)
dir out rev ERROR 15,096,742,782 (derived) 6,914,748,995 (raw)
ori out snp ERROR 17,075,708,289 (derived) 108,109,058 (raw)
rel in snp ERROR 16,845,902,398 (derived) 14,386,337 (raw)
rev in all ERROR 16,831,516,061 (derived) 1,734,773,279 (raw)
rev in dir ERROR 16,831,516,061 (derived) 1,734,773,279 (raw)
rev in rel ERROR 16,831,516,061 (derived) 1,734,773,279 (raw)
rev in rev ERROR 16,831,516,061 (derived) 1,734,773,279 (raw)
rev in snp ERROR 16,831,516,061 (derived) 1,734,773,279 (raw)
rev out rev ERROR 16,831,516,061 (derived) 1,734,773,279 (raw)
snp in ori ERROR 16,967,599,231 (derived) 121,696,833 (raw)
snp out all ERROR 16,967,599,231 (derived) 121,696,833 (raw)
snp out rel ERROR 16,967,599,231 (derived) 121,696,833 (raw)
snp out rev ERROR 16,967,599,231 (derived) 121,696,833 (raw)
end

Control status : Failed
```

After bug fix

[161]:
```
print("Control script according to criterion 1")
print()
DATASET = Path('../experiments/deprecated/20210403/') # after bugfix
error=False
for name in inout_per_type:
    dist = d(DATASET / f'inout/per_type/{name}.txt')
    src, direction, dst = name.split('_')
    sy=np.sum(dist.y)
    sxy=np.sum(dist.x*dist.y)
    if dist.x[0]==0:
        s0=dist.y[0]
    else:
        s0=0
    rs=rawstats20210403["nodes"][short2longname[src]]
    if sy==rs:
```

```
        print(f'{src} {direction} {dst} OK,', end=" ")
    else:
        print(f'{src} {direction} {dst} ERROR {int(sy):,} (derived) {int(rs):,}␣
 ↪(raw)')
        error=True
print("end")
print()
if error:
    print("Control status : Failed")
else:
    print("Control status : Successful")
```

```
Control script according to criterion 1

cnt in dir OK, dir in all OK, dir in dir OK, dir in rev OK, dir out all OK, dir
out cnt OK, dir out dir OK, dir out rev OK, ori out snp OK, rel in snp OK, rev
in all OK, rev in dir OK, rev in rel OK, rev in rev OK, rev in snp OK, rev out
rev OK, snp in ori OK, snp out all OK, snp out rel OK, snp out rev OK, end

Control status : Successful
```

1.3.2 Data integrity: nodes without ancestors / Compression Pipline

Control script corresponding ti criteria 5 on dataset 2020-05-20 (https://annex.softwareheritage.org/public/dataset/graph/2020-05-20/compressed/) lead to the following result

[162]:
```
print("Control script according to criterion 4 : Nodes without ancestors")
print()
DATASET = Path('../experiments/deprecated/20210403/') # after bugfix
error=False
for name in inout_per_type:
    dist = d(DATASET / f'inout/per_type/{name}.txt')
    src, direction, dst = name.split('_')
    sy=np.sum(dist.y)
    sxy=np.sum(dist.x*dist.y)
    if dist.x[0]==0:
        s0=dist.y[0]
    else:
        s0=0
    if ((dst=="all" and direction!="out") or (direction=="in" and src!="dir" and␣
 ↪src!="rev")) and s0!=0:
    #if (dst=="all" or (direction=="in")) and s0!=0:
        rs=rawstats["nodes"][short2longname[src]]
        ds=s0/rs
        print(f'{src:3} {direction:3} {dst:3} ERROR {int(s0):17,} {ds:<}')
```

```
        error=True
print("end")
print()
if error:
    print("Control status : Failed")
else:
    print("Control status : Successful")
```

```
Control script according to criterion 4 : Nodes without ancestors

cnt in  dir ERROR       302,918,865 0.033095588214573306
dir in  all ERROR         1,968,810 0.0002492925014586481
rel in  snp ERROR           428,698 0.02591958892198736
rev in  all ERROR        17,852,476 0.009032476941502388
snp in  ori ERROR        51,899,904 0.3711569416645763
end

Control status : Failed
```

37% of the snaphost nodes were not connected to an upstream origins

It appears that a more recent export (2020-12-15) did not show the same problem.

https://annex.softwareheritage.org/public/dataset/graph/2020-12-15/compressed/

[163]:
```
print("Control script according to criterion 4 : Nodes without ancestors")
print()
DATASET = Path('../experiments/deprecated/20210602/') # after bugfix
error=False
for name in inout_per_type:
    dist = d(DATASET / f'inout/per_type/{name}.txt')
    src, direction, dst = name.split('_')
    sy=np.sum(dist.y)
    sxy=np.sum(dist.x*dist.y)
    if dist.x[0]==0:
        s0=dist.y[0]
    else:
        s0=0
    if ((dst=="all" and direction!="out") or (direction=="in" and src!="dir" and␣
 ↪src!="rev")) and s0!=0:
    #if (dst=="all" or (direction=="in")) and s0!=0:
        rs=rawstats["nodes"][short2longname[src]]
        ds=s0/rs
        print(f'{src:3} {direction:3} {dst:3} ERROR {int(s0):17,} {ds:<}')
        error=True
print("end")
print()
if error:
    print("Control status : Failed")
```

```
else:
    print("Control status : Successful")
```

```
Control script according to criterion 4 : Nodes without ancestors

dir in  all ERROR         1,343,830 0.00017015696904992107
rel in  snp ERROR           427,531 0.025849030719541907
rev in  all ERROR        21,591,750 0.010924366121634006
snp in  ori ERROR            53,736 0.0003842875974739312
end

Control status : Failed
```

It appears that now less than 0,04% of the snapshot do not have ancestors. Directory nodes without ancestors decrease from 3.3% to less than 0.02%

1.3.3 Data integrity: nodes without ancestors / Raw Dataset

Due to the update mechanisms of the software heritage project base, having parents without ancestors can have several origins. One of them is the atomicity that is not guaranteed during an update in the bottom-up direction. That is, if there is a problem in the indexing process of new software artifacts, the nodes of the filesystem layer, for example, may have been injected without the nodes of the history or hosting layers having been injected. In most cases, when crawlers return to an origin whose last visit was an error, the injection process based on intrinsic identifiers corrects the problem.

Similarly, the process, used up to now, to export the graph and build a compressed version is not atomic. So there may be a time shift of the same type with some objects missing at the frontier of the current injection processes.

In both cases, the problems should be only temporary and it is possible to check whether this explains all or some of the nodes without ancestors we have found, by - comparing two exports separated by a time guaranteeing that the crawlers have returned to the failed visits, and checking that most nodes missing an ancestor in the frirst export, have one in the second export. - comparing the 2020-12-15 export with the information contained in the Software Heritage project database. In the case of revisions not linked to an ancestor, it is sufficient to check whether these revisions were seen in visits much older than the export date, or on the contrary close to the limit represented by the export date.

We did this on a sample of 1000 revisions without ancestors, identifying for 98.5% of them (see file *rev1000.txt*) the oldest visit in which it was seen (without having to go back in the chain of revisions).

A small proportion of these revisions have been seen recently. This invalidates the hypothesis according to which nodes without ancestors are primarily caused by the non-atomicity of the crawling process and the export process.

Further investigation is needed. An anomaly report has been filed https://forge.softwareheritage.org/T3660.

At this point, we have no evidence that these anomalies have a significant impact on the results presented in this study. Nevertheless, this is one of the limitations of this study, and will need further investigation.

[ ]: