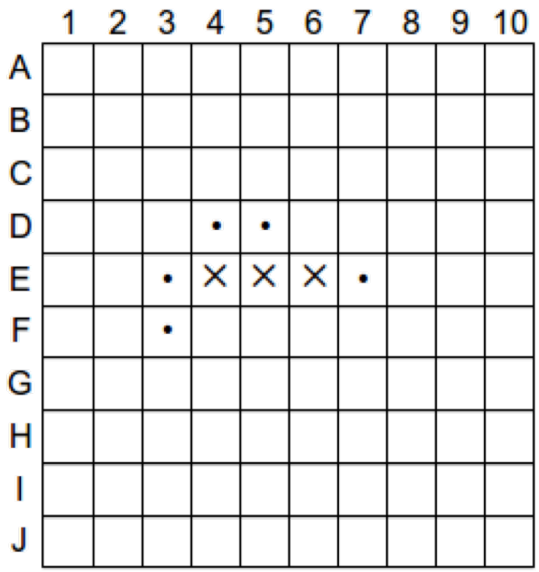

# Decision-Theoretic File Carver For Triage Situations

Pavel Gladyshev, Joshua James

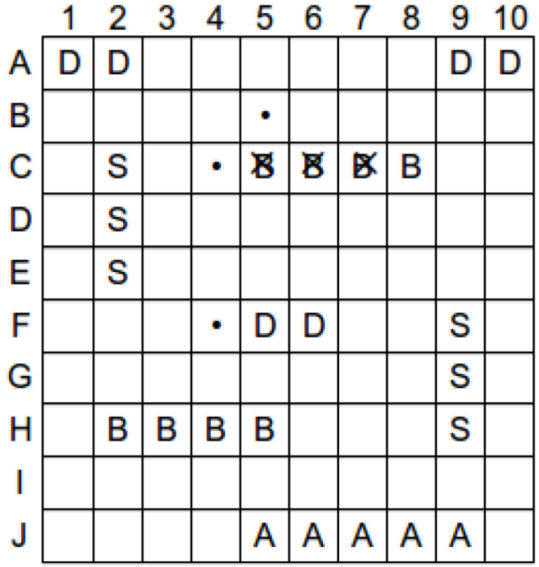

http://dfire.ucd.ie/

# Battleship game

Opponent's Ships



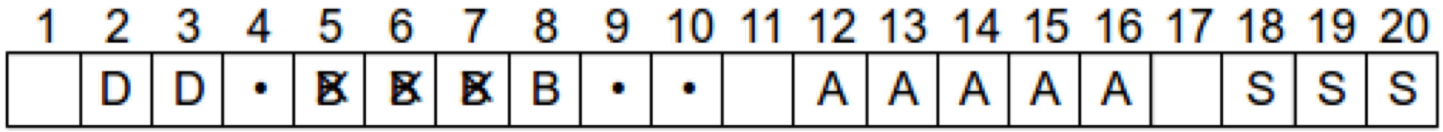

My Ships




Fleet:

| | |
|---|---|
| 1x Aircraft Carrier: | AAAAA |
| 2x Battleship: | BBBB |
| 2x Submarine: | SSSS |
| 3x Destroyer: | DD |

# File carving as a Battleship game




Fleet:

1x Aircraft Carrier: AAAAA

1x Battleship: BBBB

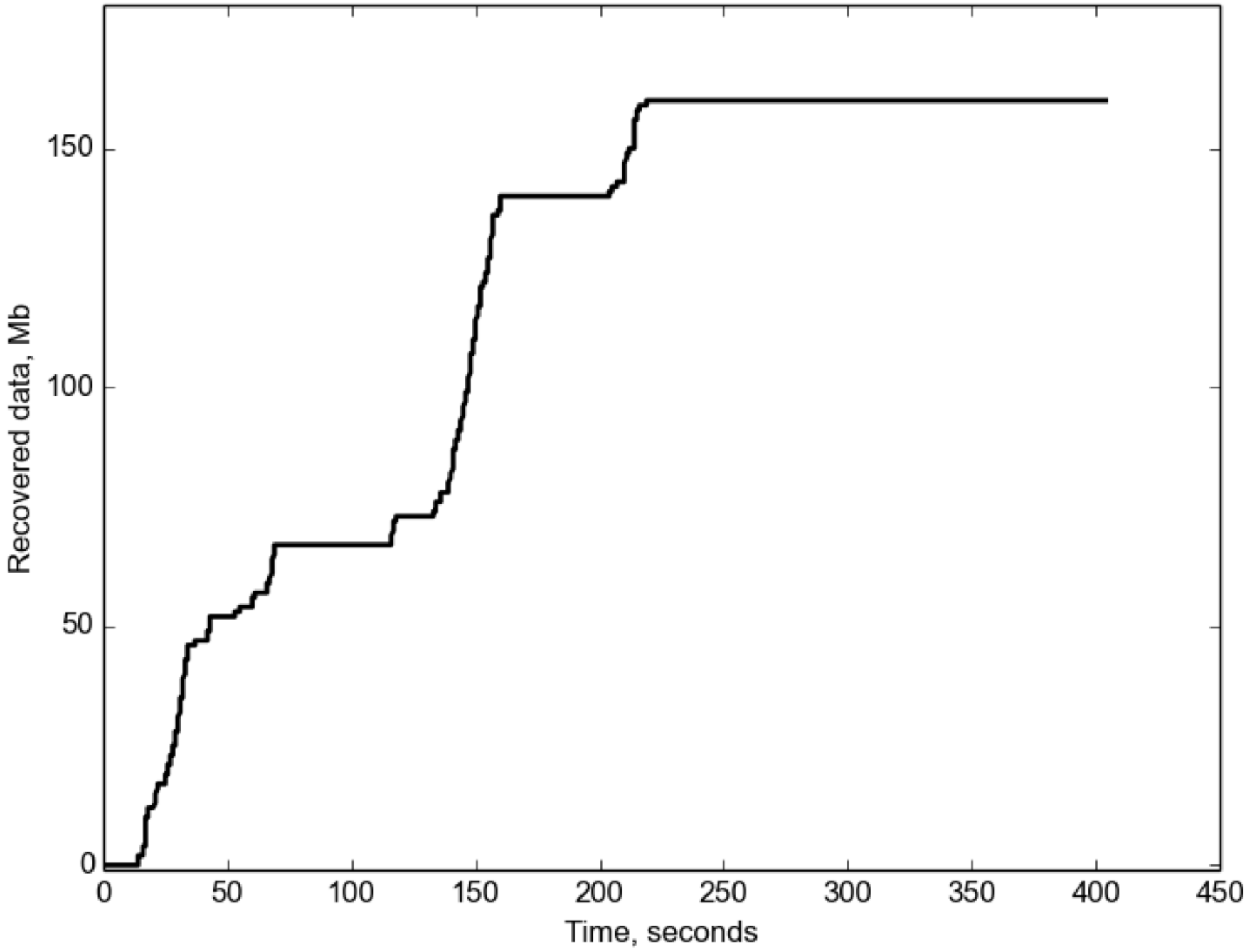
1x Submarine: SSS

1x Destroyer: DD

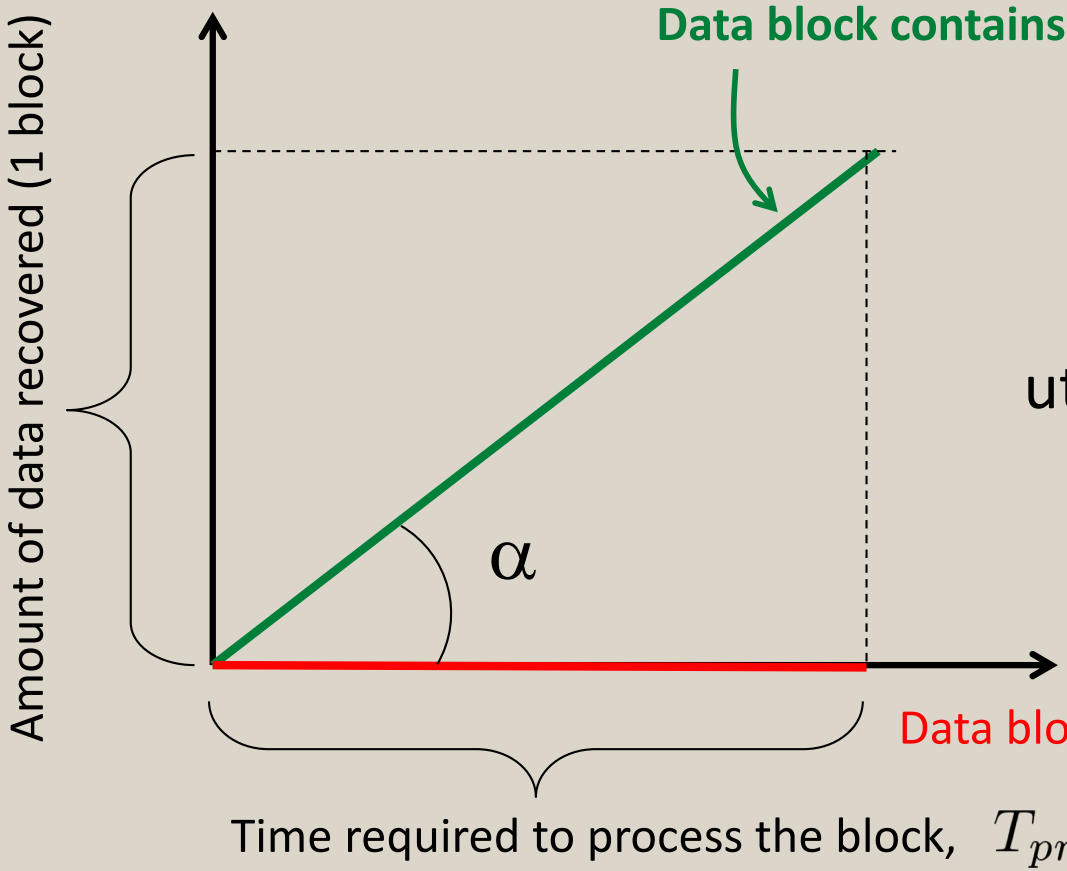
# Decision-theoretic forensic algorithm

- Aims to find *relevant* data as quickly as possible.
- Greedy approach:
  - Input = collection of items to be processed
  - Always pick the item with the maximal expected utility
  - Keep running until all items are processed
- How do we determine expected utility?

**Amount of relevant data recovered by sequential carver over time**

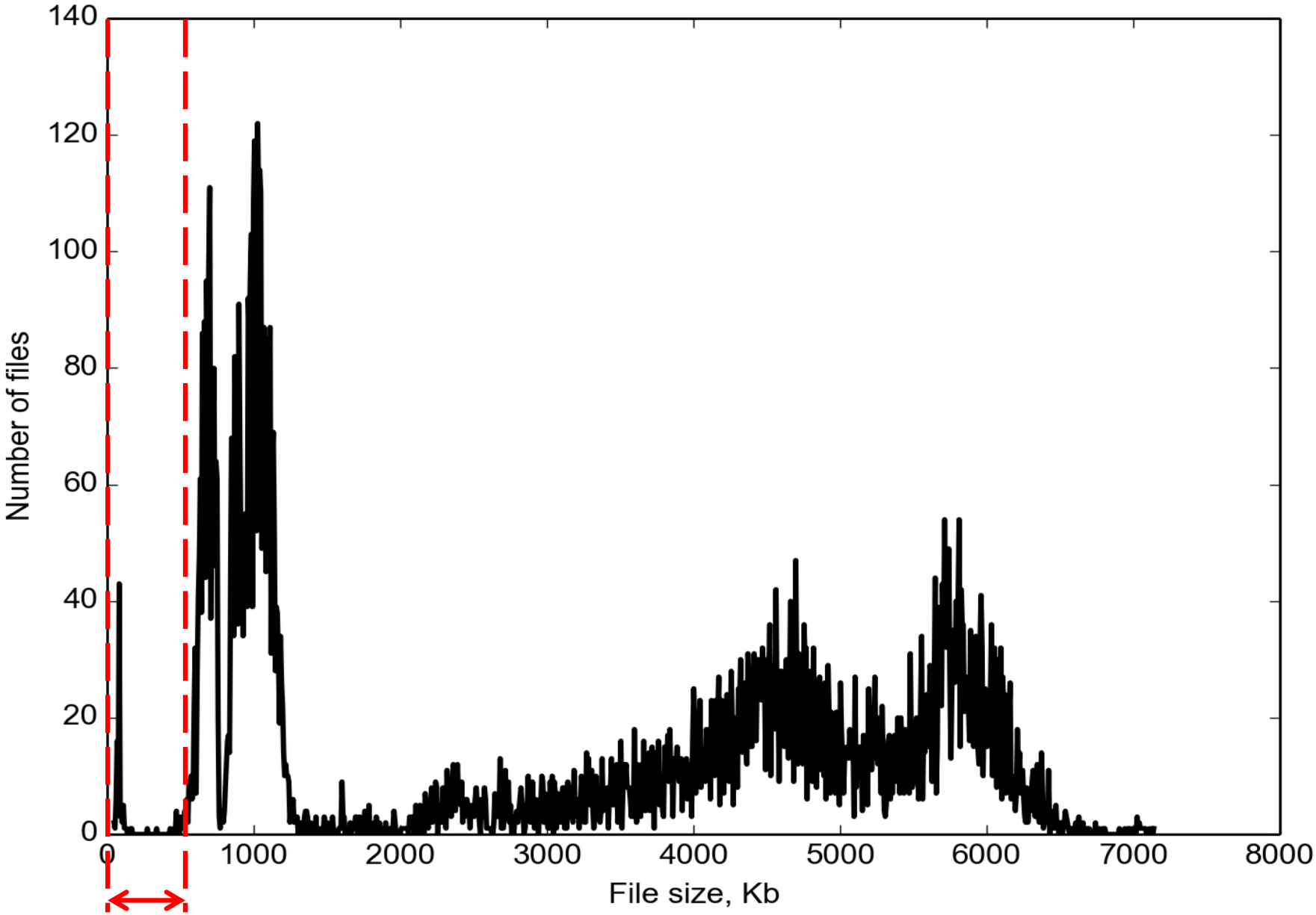


# Utility of processing a block = slope of the graph




$$\text{utility} = \begin{cases} 0 & \text{, if irrelevant} \\ \frac{1}{T_{proc}} & \text{, if relevant} \end{cases}$$

Sizes of 5000 raw JPEG files

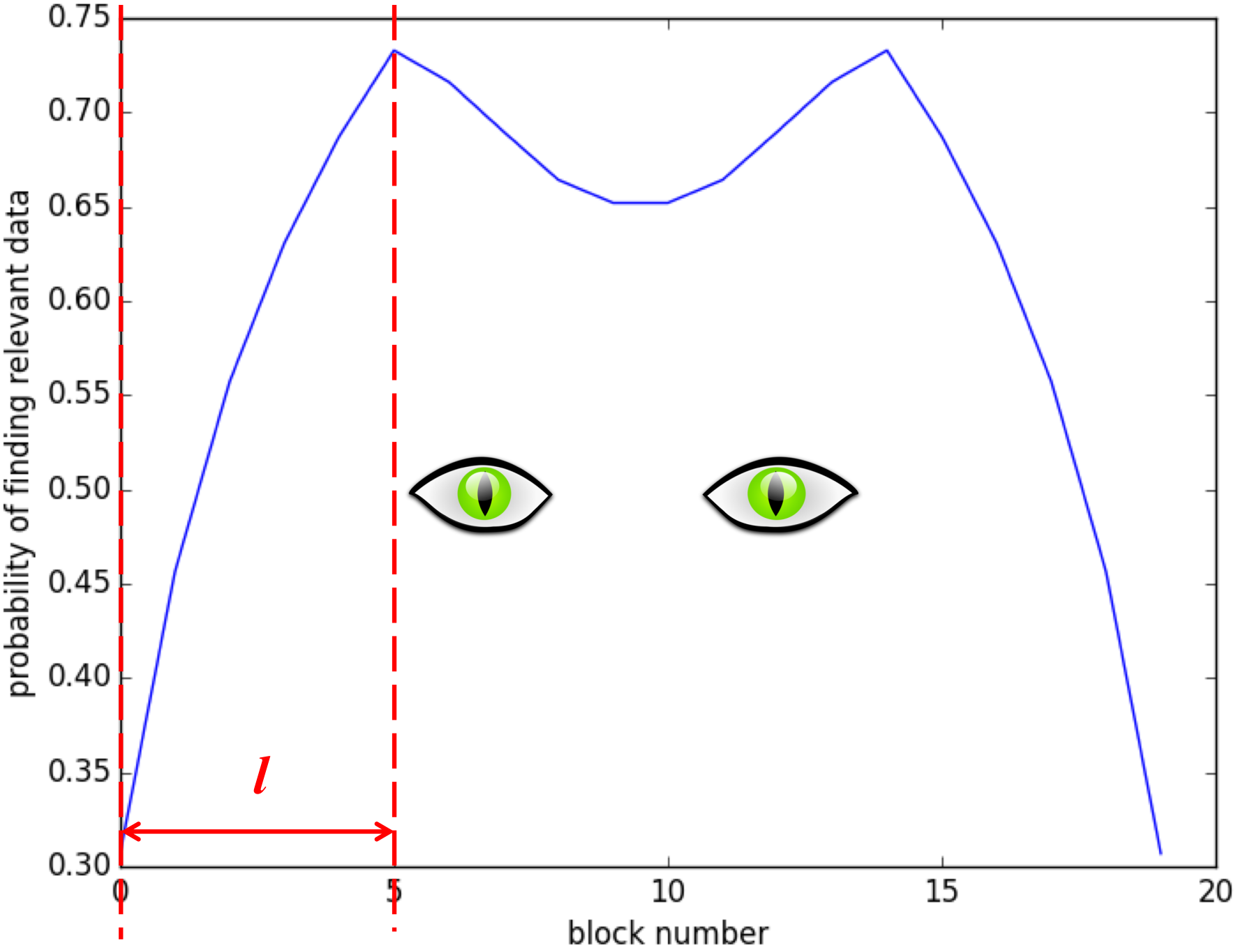



$l$ – minimal JPEG file size

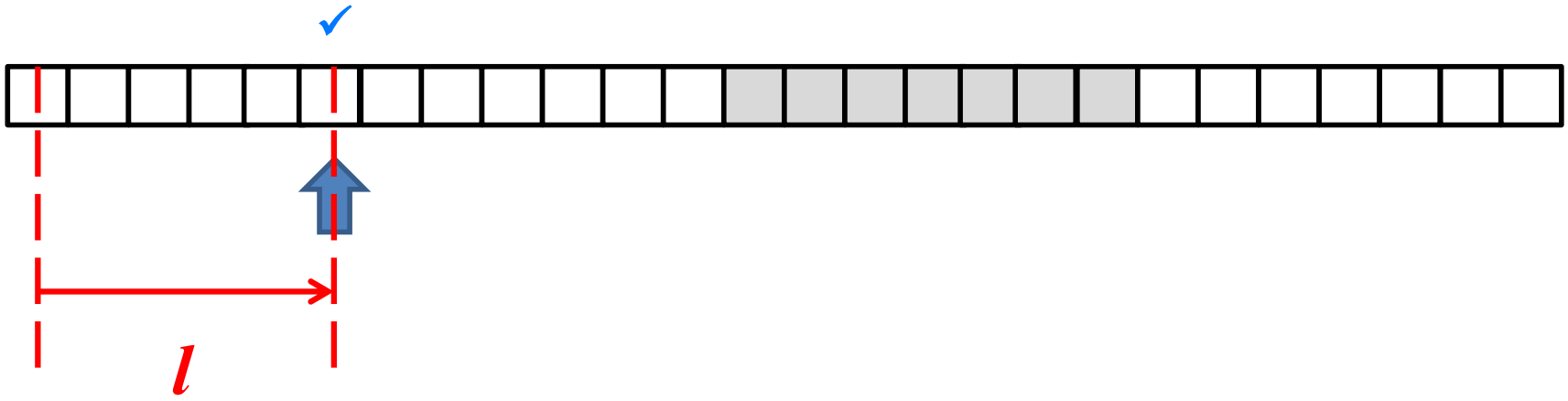
# Assumptions

- No fragmentation
- Minimal file size ($l$) for the target file type (raw JPEG)
- Files are uniformly spread throughout the disk
- Relevant data blocks can be detected accurately and efficiently

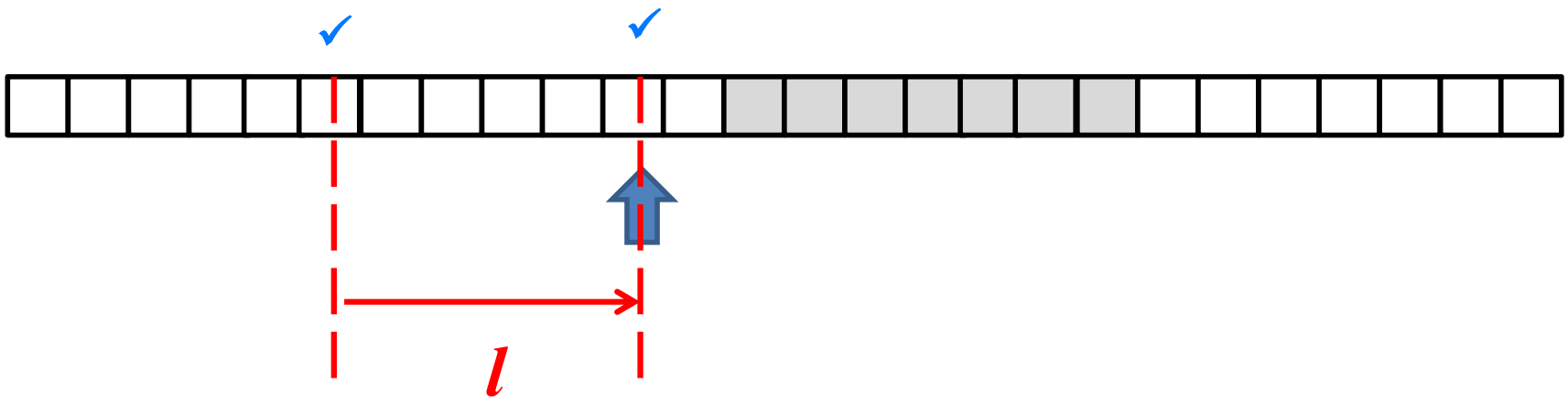
Probability of a data block containing relevant information (simulation result)



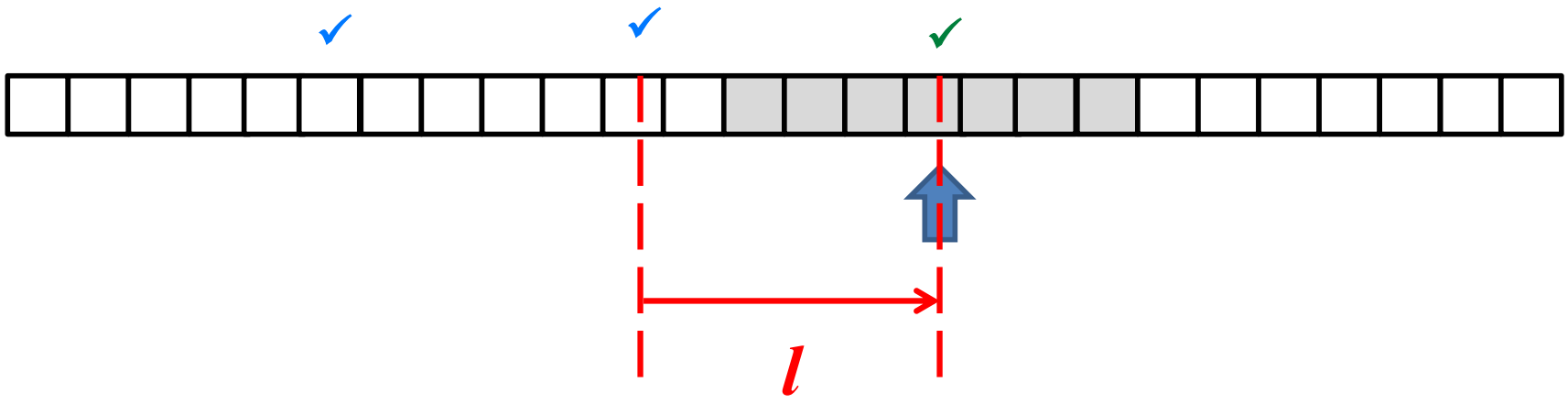
# DeCa Algorithm: Sampling mode



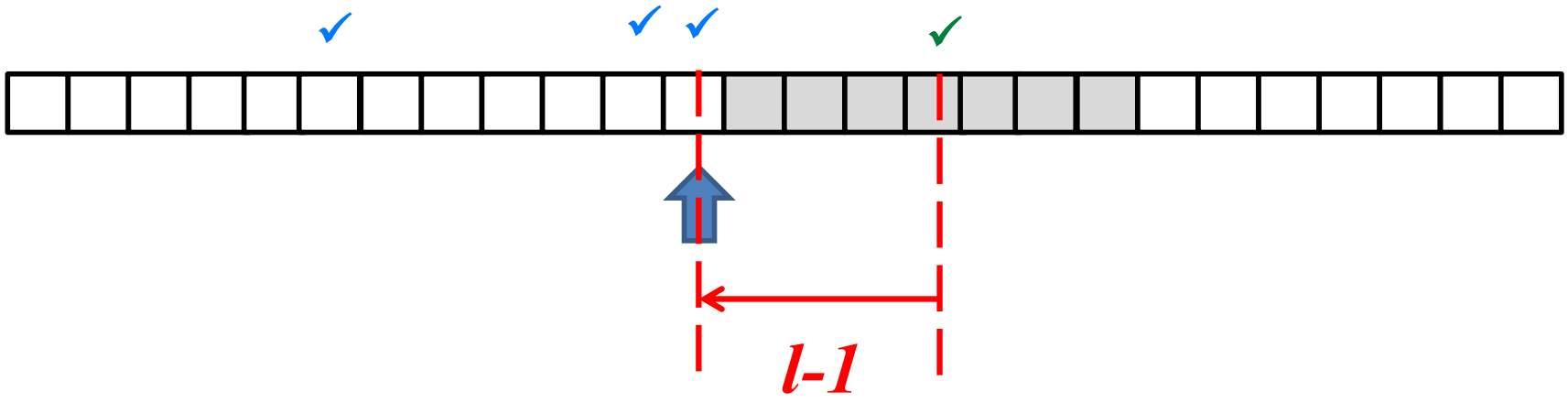
# DeCa Algorithm: Sampling mode



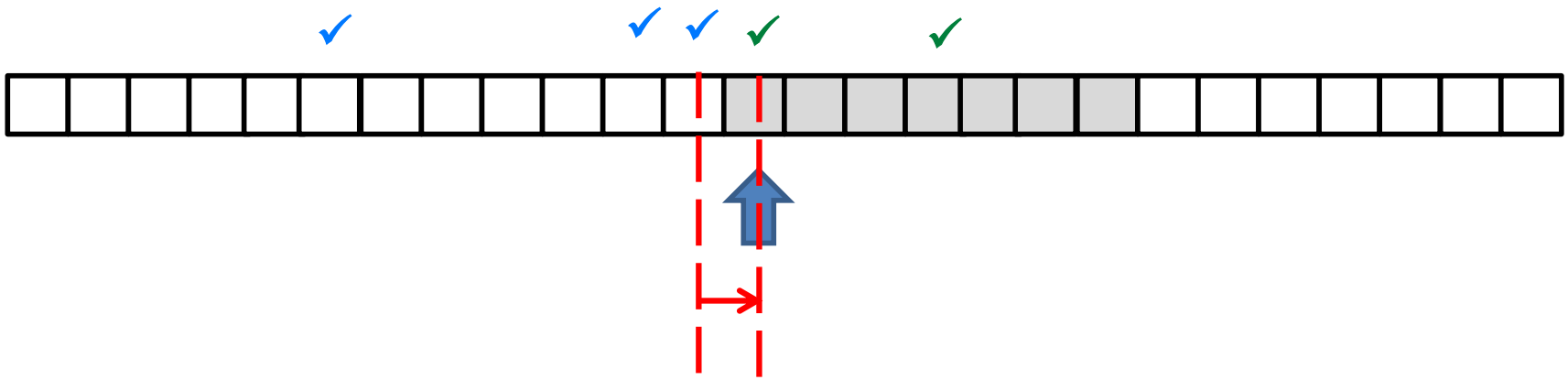
# DeCa Algorithm: Sampling mode



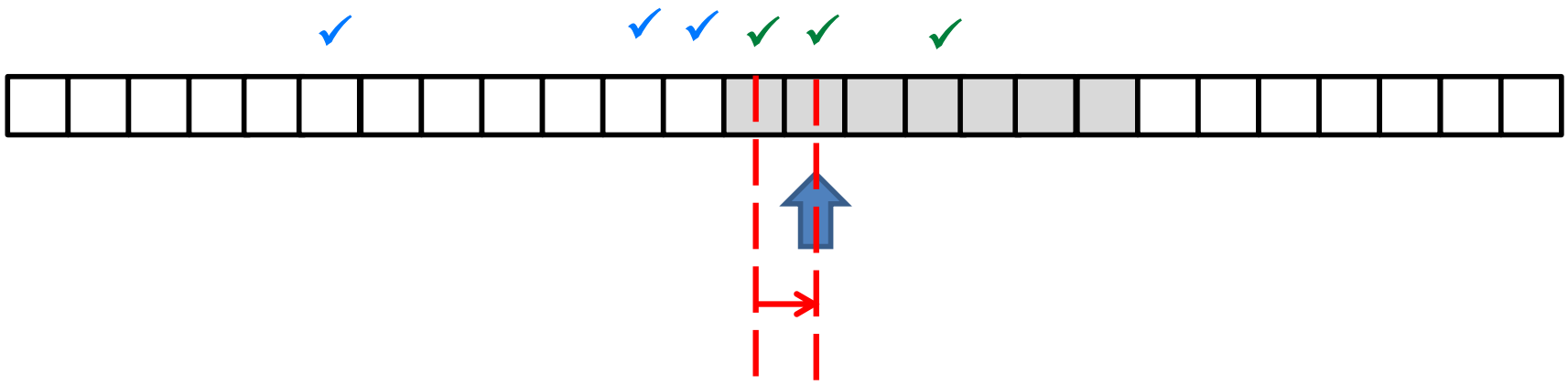
# DeCa Algorithm: relevant data found, jump back



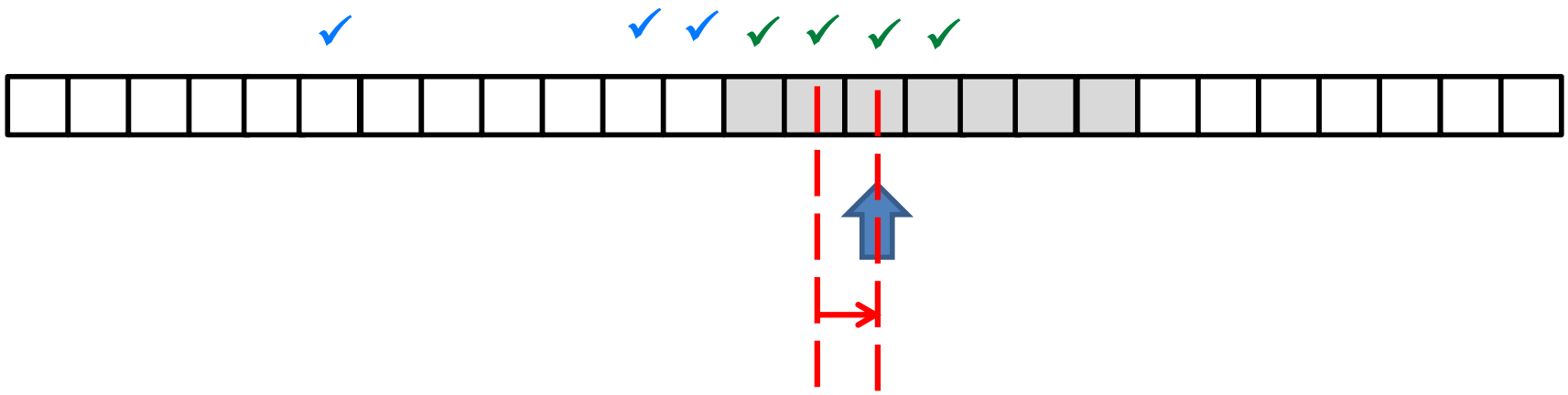
# DeCa Algorithm: Sequential carving (header / footer)



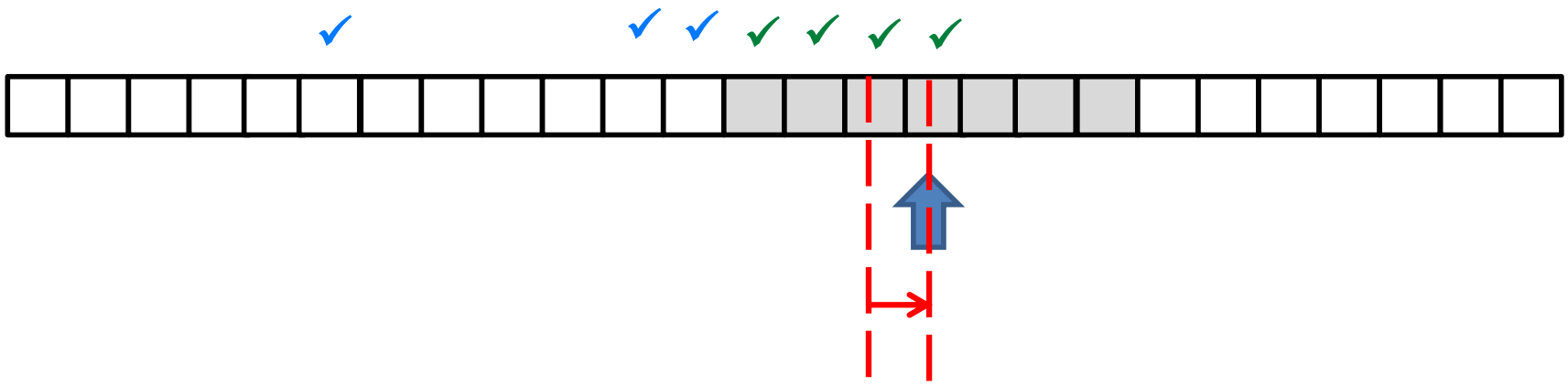
# DeCa Algorithm: Sequential carving (header / footer)



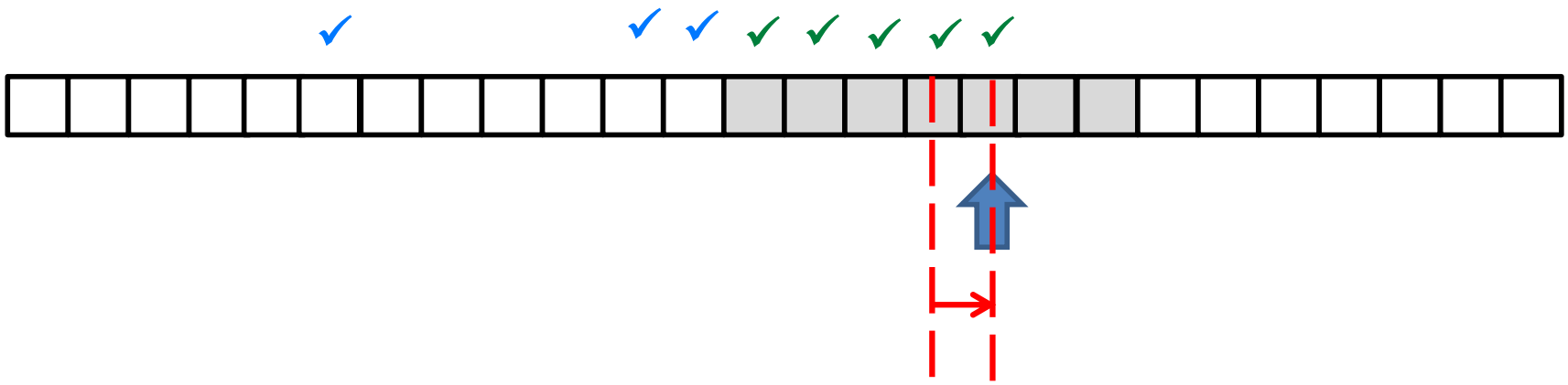
# DeCa Algorithm: Sequential carving (header / footer)



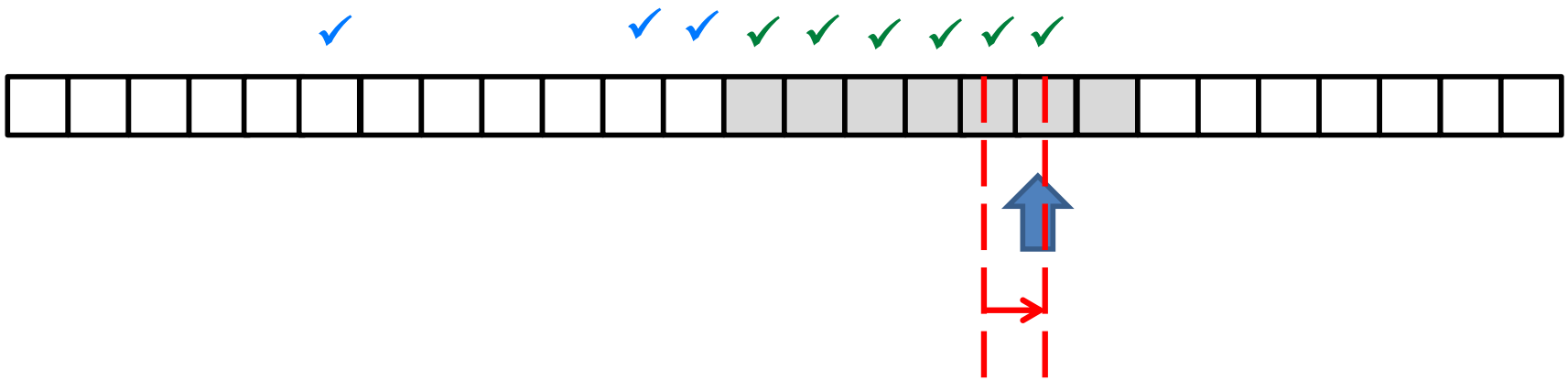
# DeCa Algorithm: Sequential carving (header / footer)



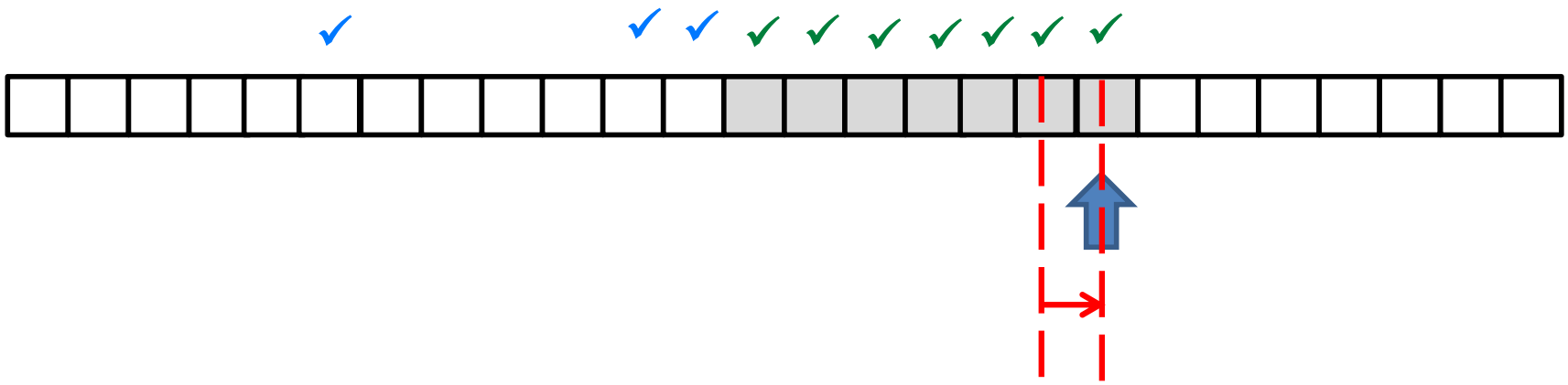
# DeCa Algorithm: Sequential carving (header / footer)



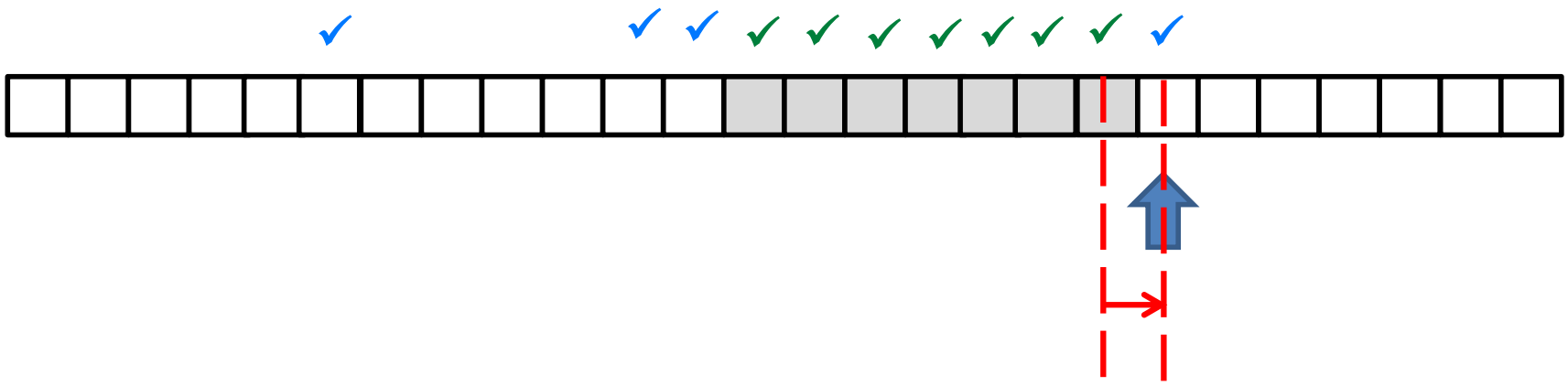
# DeCa Algorithm: Sequential carving (header / footer)



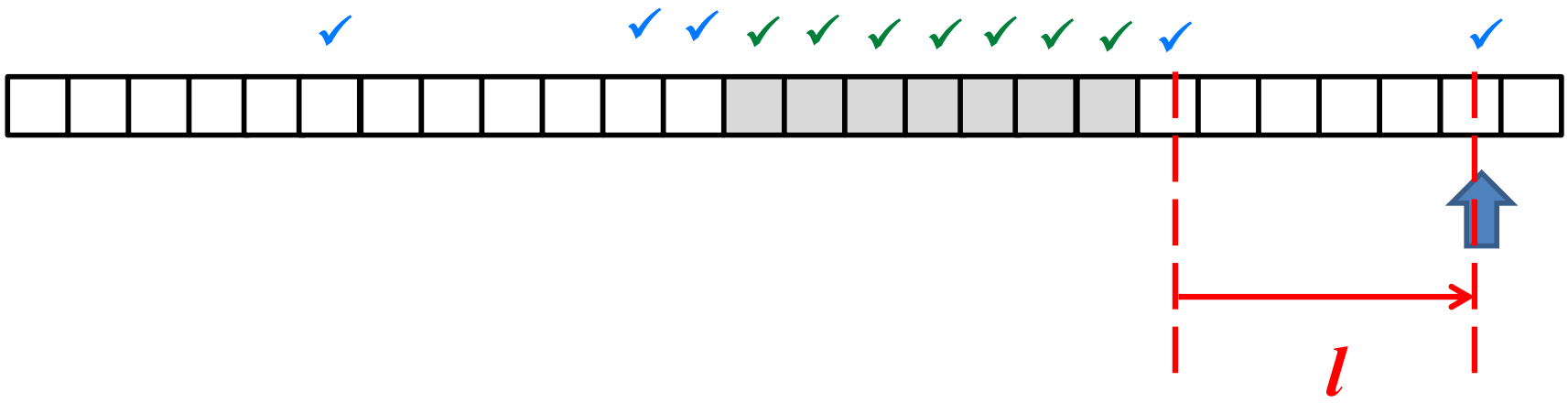
# DeCa Algorithm: Sequential carving (header / footer)

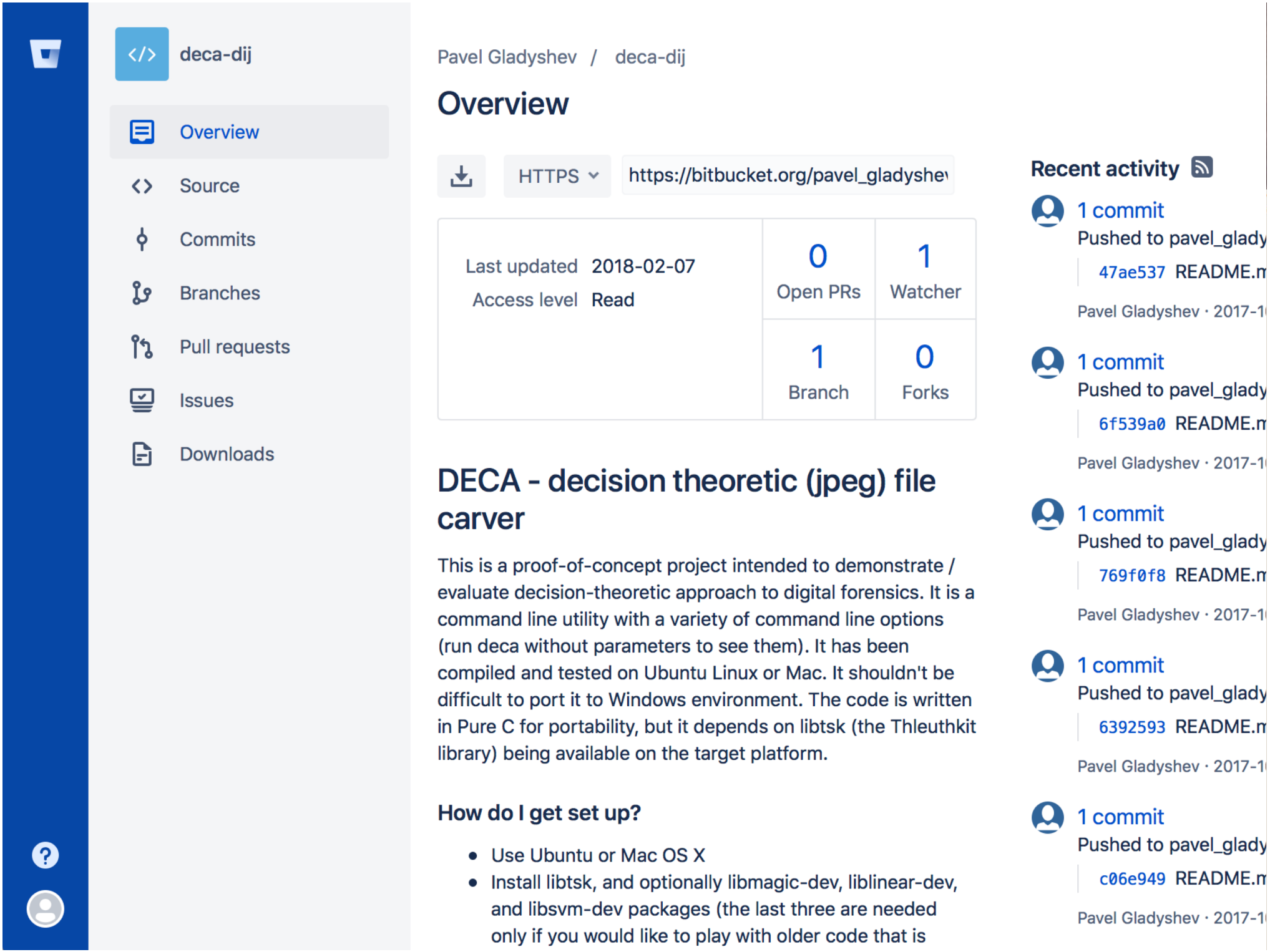


# DeCa Algorithm: next block does not contain JPEG, switching to sampling mode



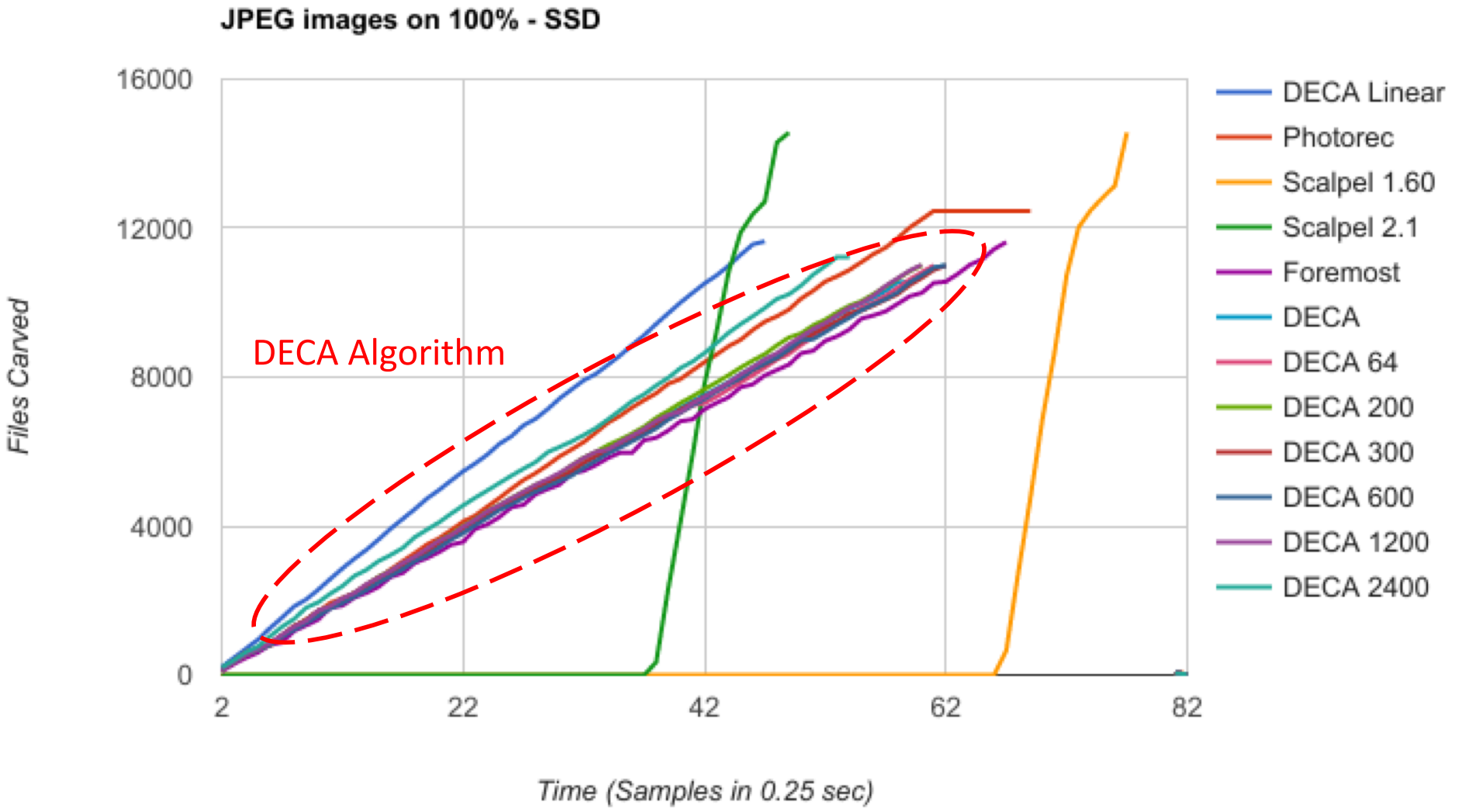
# DeCa Algorithm: Sampling mode

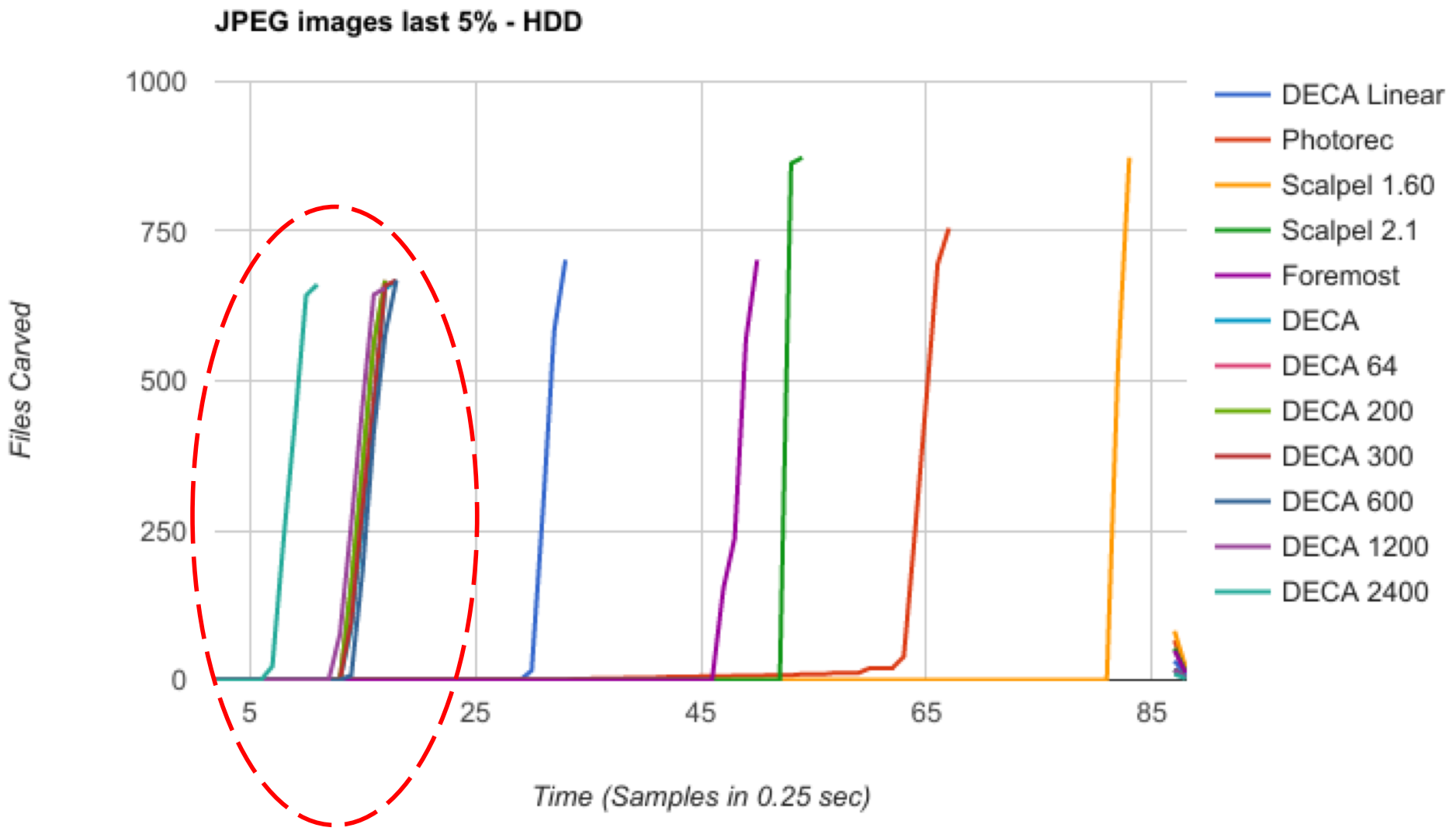



deca-dij
Overview
Source
Commits
Branches
Pull requests
Issues
Downloads
Pavel Gladyshev / deca-dij
Overview
HTTPS
https://bitbucket.org/pavel_gladyshev
Last updated 2018-02-07
Access level Read
0
Open PRs
1
Watcher
1
Branch
0
Forks
DECA - decision theoretic (jpeg) file carver
This is a proof-of-concept project intended to demonstrate / evaluate decision-theoretic approach to digital forensics. It is a command line utility with a variety of command line options (run deca without parameters to see them). It has been compiled and tested on Ubuntu Linux or Mac. It shouldn't be difficult to port it to Windows environment. The code is written in Pure C for portability, but it depends on libtsk (the Thleuthkit library) being available on the target platform.
How do I get set up?
Use Ubuntu or Mac OS X
Install libtsk, and optionally libmagic-dev, liblinear-dev, and libsvm-dev packages (the last three are needed only if you would like to play with older code that is
Recent activity
1 commit
Pushed to pavel_glady
47ae537 README.m
Pavel Gladyshev · 2017-1
1 commit
Pushed to pavel_glady
6f539a0 README.m
Pavel Gladyshev · 2017-1
1 commit
Pushed to pavel_glady
769f0f8 README.m
Pavel Gladyshev · 2017-1
1 commit
Pushed to pavel_glady
6392593 README.m
Pavel Gladyshev · 2017-1
1 commit
Pushed to pavel_glady
c06e949 README.m
Pavel Gladyshev · 2017-1

# Experiments

- Different drive types
  - HDD
  - SSD
  - RAM drive
- Different file allocation patterns
  - Entire drive (100%) filled with JPEGs
  - Last 5% filled with JPEGs
  - Groups of JPEG files throughout the drive (2%-32%)


JPEG images on 100% - SSD
16000
12000
8000
4000
0
2
22
42
62
82
Files Carved
Time (Samples in 0.25 sec)
DECA Algorithm
DECA Linear
Photorec
Scalpel 1.60
Scalpel 2.1
Foremost
DECA
DECA 64
DECA 200
DECA 300
DECA 600
DECA 1200
DECA 2400




DECA Algorithm

# DeCa Code Repository

https://bitbucket.org/pavel_gladyshev/deca-dij.git