Decision Theoretic File Carving Based on Cluster Sampling

Pavel Gladyshev[†], Joshua I. James[‡]
pavel.gladyshev@ucd.ie, joshua@cybercrimetech.com
[†] Digital Forensic Investigation Research Group: Europe
University College Dublin, Belfield, Dublin 4, IE
[‡] DFIRE Labs, Legal Informatics and Forenisc Science Institute
Hallym University, Chuncheon, South Korea

Abstract

Abstract **Keywords:** Decision-theoretic file carving; File carving; Digital Forensic Investigation; Digital Forensic Triage; Preliminary Analysis

1 Introduction

Digital forensic investigations must process large amounts of evidential data with limited resources [1, 2]. Modern solutions to the data processing problem combine advances in automatic detection of relevant data with some form of selective human exploration to identify sample data and to validate output results [3, 4, 5]. Despite significant time and cost savings that automation can bring to digital investigations, this approach still relies on exhaustive processing of the suspect data.

Various methods have been proposed that attempt to prioritise exhibits by first prioritising suspect data sources that the exhibit contains [6, 7, 8]. Several works [9, 1] have discussed the idea of digital forensic triage; an investigation processes with the goal of not always requiring an exhaustive search of all devices, and helps with decisions of relevancy and prioritisation.

There are many situations where a digital investigator is limited either in time or available computational power. For example, a parole officer may use a portable forensic tool to periodically check that an offender's computer does not contain child abuse material. The time and processing power available to the officer on site is limited and – given the growth in data storage capacity – may not be sufficient to exhaustively explore the contents of an offender's computer. Thus a solution is required that would combine automatic processing with probabilistic sampling and prioritisation aimed at discovering relevant information more quickly.

This paper explores the idea of automatic prioritisation and sampling from theoretical and practical standpoints, and demonstrates its viability by example of JPEG image carving.

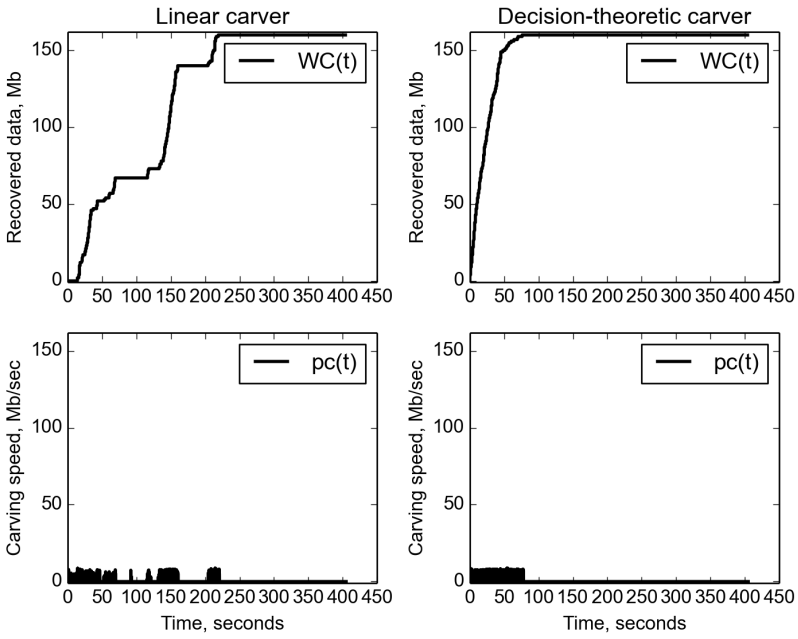

Figure 1: Performance of linear vs decision-theoretic carver.

1.1 Contribution

This work contributes to the field of digital forensic investigation by proposing a limited-resource, file carving method that is suitable for digital forensic triage purposes as defined by [9]. Specifically, this work provides:

- a rigorous statement of the problem of digital forensic investigation with limited resources by viewing digital forensics as a decision problem
- a decision-theoretic approach to file carving that is able to recover up to half of all JPEG files on a disk with 10% of the processing time
- demonstrated viability of the proposed approach by constructing a JPEG image carving application that finds potentially relevant images (photographs) faster than traditional carvers

2 File Carving Approaches

File carving is often used in digital investigations, and a number of file carving techniques have been proposed. Traditional file carvers, such as foremost [10] or photorec [11], examines the disk blocks in a linear fashion – starting at the beginning of the drive and working its way towards the end of the drive. Figures 1a and 1b show typical plots of $WC(t)$ and $pc(t)$ for such a linear file carver.

If the relevant data is spread uniformly over the disk space, the traditional carvers could take long time to find most of it.

Richard, et al [12], and similarly Meijer [13], proposed in-place[1] file carving in an attempt to lower the time and space requirements compared to traditional linear carving. These approaches identify the challenges with traditional file carving, and propose methods to reduce carving resource requirements by minimizing the amount of disk I/O associated with copying all file data. Essentially, once file data is discovered, pointers are created in virtual file systems that point to the data directly on the suspect disk rather than copying data from the suspect disk. However, the approach used to determine if a block contains file data is still linear, as before.

Hash-based: [14] Fragment / Cluster classification: [15] [16]

3 Decision-theoretic File Carving

The main objective of a file carver is to recover as much relevant data as possible, while producing little spurious results. Prior research efforts primarily focused on achieving this objective. Nowadays, the capacity of the data storage encountered in investigations made file carving a time consuming process and it is important to start thinking about making the file carving *quick* as well as precise. To put it into perspective, consider that if a forensic workstation takes 400 seconds to process a 32Gb hard disk drive, it would take it more than 10 hours to process a 3 Tb hard disk drive.

This paper describes an adaptive file carving algorithm that uses initial probing to estimate where the relevant information is likely to be on the drive, and uses this estimate to guide the file carving process.

To explain this idea more precisely, let's introduce two mathematical functions. Let $WC(t)$ denote the amount of relevant data extracted by the carver at the time t since the start of carving, and let $pc(t)$ denote the *carving speed* – the rate at which the file carver produces relevant data. For the purposes of this paper we define $pc(t)$ to be a difference

$$pc(t) = WC(t+h) - WC(h) \tag{1}$$

over some small time interval h.

Alternatively, file carvers could use a heuristic model to *guess* which data blocks are most likely to contain relevant data and prioritize examination of those blocks. If the guessing is sufficiently precise and if the analysis of the guessed blocks can be done efficiently[2], the carver will recover most of the relevant data at the beginning of the carving process as shown in Figures 1c and 1d.

A very basic implementation of this idea is described in the following sections. Despite its simplicity, in our experiments it could recover up to half of all JPEG files from the disk image during the initial 10% of the processing time.

The ability to deliver a lot of relevant data quickly, could make such file carving usable in time-sensitive situations, such as digital forensic triage or on-

[1] Also known as zero-storage

[2] We foresee a reasonable objection from our reader that on hard disk drives the sequential access to the data is orders of magnitude faster than random access, but we defer this discussion until Section X. It suffices to say that the development of solid state drives and the evolution of RAMCloud [17] and similar technologies provide valid business cases for the decision-theoretic file carving.

scene investigations, where file carving is not normally used due to the long wait required.

4 DECA algorithm

There are many games like Battleships, Minesweeper, and the Colossal Cave Adventure in which player explores an unknown environment. It is instructive to consider file carving as an analogy of the battleships game shown in Figure 2.

According to [18] the Battleships game originated in Russia in the pre-World War I days. Each player has a pad of paper with two 100-square grids – one for arranging the player's ships and the other for noting the discovered locations of the opponent's ships. Every square on a grid is identifiable by a letter and a number. Before the start of the game, each player secretly arranges their ships on their grid. A ship occupies a number of consecutive squares and is oriented either vertically or horizontally. The ships cannot overlap. The game proceeds in turns with each player sending a "bomb" to a particular square on the enemy grid. The other player then responds whether the "bombing" succeeded in damaging one of their ships. A ship is destroyed when all of the squares it occupies has been bombed. The game ends when one of the players completely destroys the other player's fleet.

A typical strategy is to send random bombs until one of them hits an opponent's ships. The subsequent bombings are used to determine the orientation of the damaged ship and to completely destroy it by targeting the squares adjacent to the initial hit. What makes this strategy possible is the *a-priori* knowledge of the placement rules and of the dimensions of the opponent's ships.

Imagine now that instead of a square field the Battleship game is played on a long canal with the opponent's ships situated along it as shown in Figure 3. A ship occupies a number of consecutive game squares, and – like before – ships cannot overlap. A turn consists of the player bombing a particular square along the canal aiming to destroy the enemy fleet with the minimal number of bombs dropped. We hope that the analogy with file carving is self evident.

The simplest strategy that is guaranteed to destroy the enemy fleet in Figure 3 is to bomb all squares consecutively from from left to right, which corresponds to the traditional file carving approach. This strategy, however, is not the most efficient if the player has some prior information about the parameters of the fleet and/or its location along the canal. For example, if it is known that all ships occupy hundreds of adjacent squares, then if a bomb hits a ship in some square, then it is very likely that the immediately adjacent squares also belong to the same ship and that another ship is likely to be situated hundreds of squares in either direction of the damaged ship. Like a battle ships game player, DECA algorithm first searches for a file to carve by sampling previously unexplored disk blocks, and then – once a block belonging to the relevant file is found – it examines the nearby blocks to find the beginning and the end of that file. We presently ignore file fragmentation and assume that the beginning and the end of the file can be detected using magic signatures. Similar assumptions are made by popular file carvers, such as Scalpel [10] and Photorec [11].

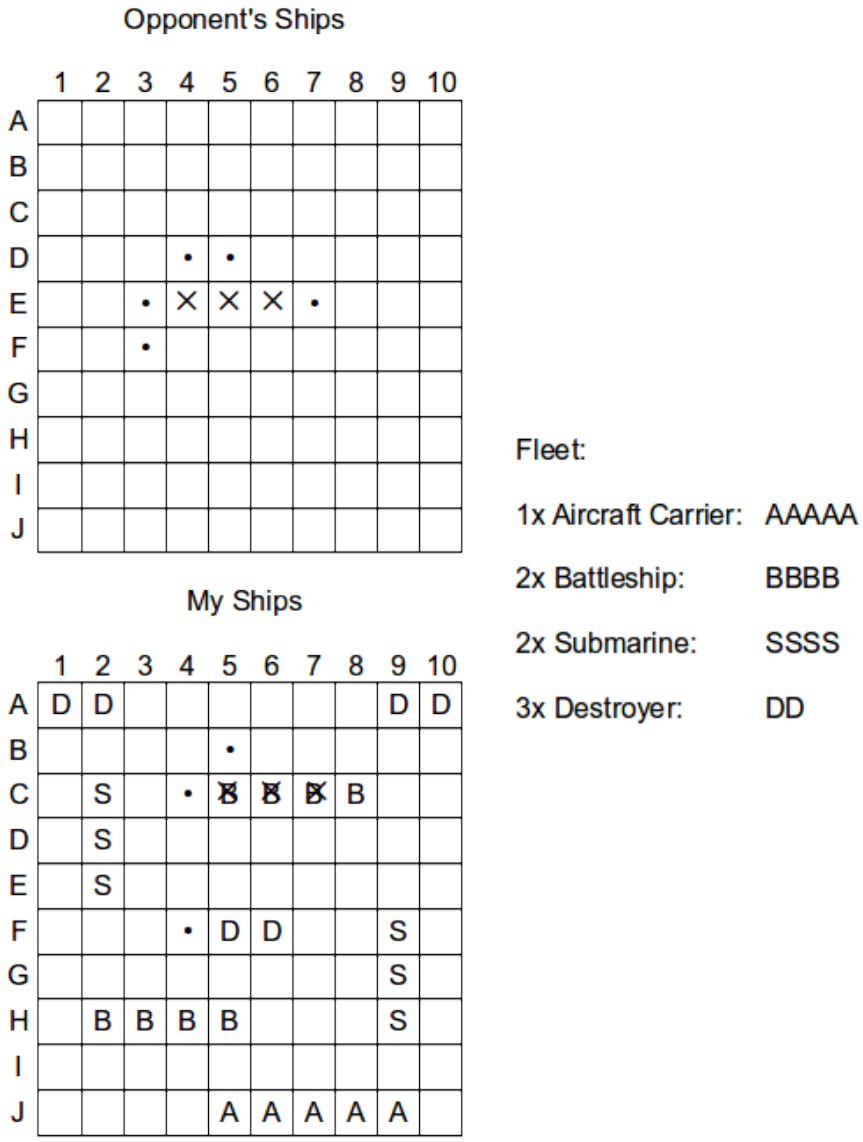

Figure 2: Battleships playing pad.

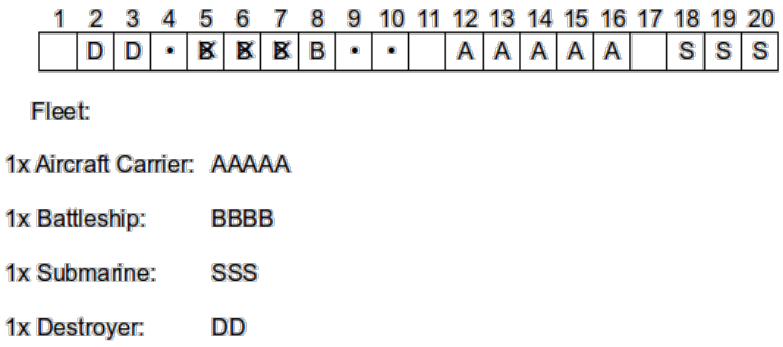

Figure 3: A 1-dimensional Battleships game.

4.1 Decision theoretic analysis of file carving

Decision theory offers a well-developed mathematical apparatus for reasoning about decision making. It models decision making as a choice problem where the decision maker has to choose one of several possible mutually exclusive actions to perform: $A = \{A_1, A_2, \ldots, A_n\}$. There is a finite set of mutually exclusive outcomes $O = \{o_1, o_2, \ldots o_m\}$ that may result from performing an action, and each outcome has a certain probability of happening if a particular action is chosen: $P(o_i \mid A_j)$. The desirability of an outcome is modelled by the utility function $u : O \to \mathbb{R}$ that assigns a numeric value to each outcome. The higher utility values correspond to more desirable outcomes. Decision theory shows that in this setting the right thing to do is to choose an action with the highest mathematical expectation of the utility value across all possible outcomes:

$$\operatorname*{argmax}_i \sum_{j=0}^{m} P(o_j \mid A_i) u(o_j) \tag{2}$$

The first phase of DECA algorithm is concerned with sampling of unexplored disk blocks. Initially all blocks are unexplored, and the algorithm has to choose which of the n available blocks $(b_0 \ldots b_n)$ to examine. There are two possible outcomes of each action: the chosen block belongs to a relevant file (i.e. the file of the kind we are trying to carve) or the block contains irrelevant data. This can be modelled by the predicate $rel(b_i)$, such that $rel(b_i) = T$, if the block b_i belongs to the relevant file and $rel(b_i) = F$ otherwise. Clearly $P(rel(b_i) \mid i) + P(\neg rel(b_i) \mid i) = 1$ and $P(rel(b_i) \mid i) = 1 - P(\neg rel(b_i) \mid i)$. Let's initially ignore the cost of examination of non-adjacent disk blocks imposed by the hard disk drive seek time, and assume a very simple utility function:

$$u(i) = \begin{cases} 1 & rel(b_i) \\ 0 & \neg rel(b_i) \end{cases} \tag{3}$$

The formula 2 then simplifies to $\text{argmax}_i\, P(rel(b_i) \mid i)$, meaning that the algorithm should choose to examine the block with the highest *a-priori* probability of being part of relevant file.

Observe that there are cases when all unexplored blocks have equal probability of containing relevant data. This could happen, for example, while carving images from unallocated space on an old CCTV system, where all of the unexplored blocks are known to contain fragments of relevant files and the carver just needs to identify start and end of each fragment. Linear carver is optimal in these circumstances, because it will always find relevant data by examining the next unexplored block. The same is true if we aim to recover /emphall the data from the unexplored blocks[3].

If the aim is to recover a *specific* type of data, such as digital photographs, digital video, and audio recordings, there is usually prior information that could be used to to speed up the carving process by prioritizing examination of the unexplored blocks. This is demonstrated in the rest of this paper. We use the fact that raw digital photographs tend to be large to compute an estimate of $P(rel(b_i) \mid i)$ and then use this estimate to gain a speed up when carving raw digital photographs.

4.2 Estimating $P(rel(b_i) \mid i)$ for raw digital photographs

A digital camera sensor produces raw photographs at a fixed resolution. The raw sensor data is resized by the camera to fit the selected output resolution, and compressed. The resulting file is, typically, several megabytes in size. Recall that JPEG image is divided into a number of 8×8 pixel fragments before compression, and each fragment is compressed independently. By the central limit theorem, the overall size of JPEG file should obey normal (Gaussian) distribution. Figure 4 shows a histogram of file sizes of a personal collection of raw digital photographs accumulated over a period of 10 years. There are several peaks corresponding to different cameras and image quality settings used over the years. Note, however, that each peak follows the normal distribution.

To get a sense for probability distribution $P(rel(b_i) \mid i)$ a series of stochastic simulations had been performed. Each simulation generated approximately 20 simulated image files. The simulated files were "stored" at random[4] locations throughout the disk block array. $P(rel(b_i) \mid i)$ was then estimated as the frequency of a particular b_i being "allocated" to an image file. The files were always stored in consecutive blocks of disk space (i.e. there was no file fragmentation).

Figure 5 shows the estimate of $P(rel(b_i) \mid i)$ after 10^6 simulations. The average size of the simulated files was kept at 300 blocks and the disk block array contained only 30,000 blocks. Observe that the probability of finding an image file by testing b_0 is much lower than the probability of finding an image file by testing a block at some distance from the beginning of the disk. The probability $P(rel(b_i) \mid i)$ reaches maximum around the block b_{300}, where 300 is the average file size of the generated files. The reason for that is the assumed absence of fragmentation, normal (Gaussian) distribution of simulated file sizes,

[3]Note that while file carvers are routinely used to recover all possible data, typically only a small fraction of that data is useful as evidence or intelligence. Most file carvers can be configured to limit file extraction to only particular kinds of files.

[4]The uniform distribution was assumed.

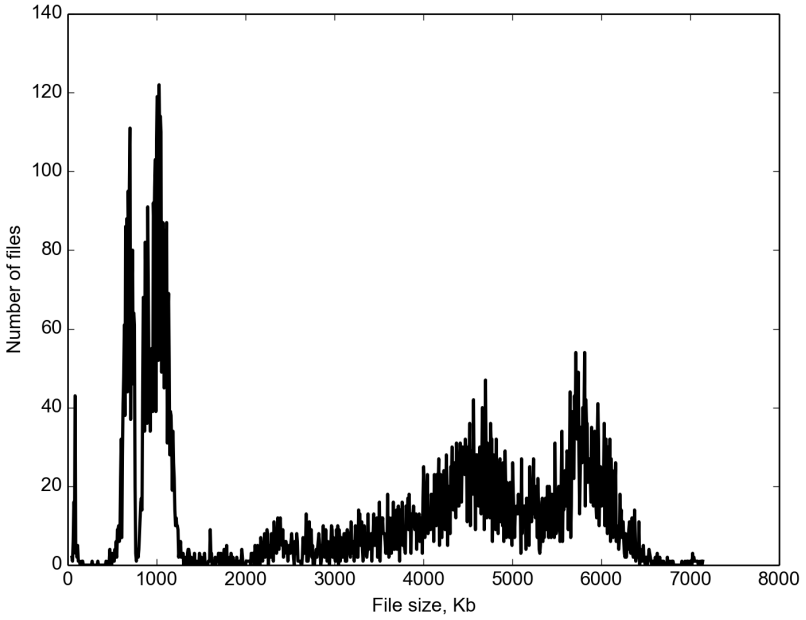

Figure 4: Histogram of file sizes (in 1 kilobyte blocks) of a personal collection of raw digital photographs in JPEG format. Each of the peaks visible in the histogram correspond to different cameras and image quality settings used over the years.

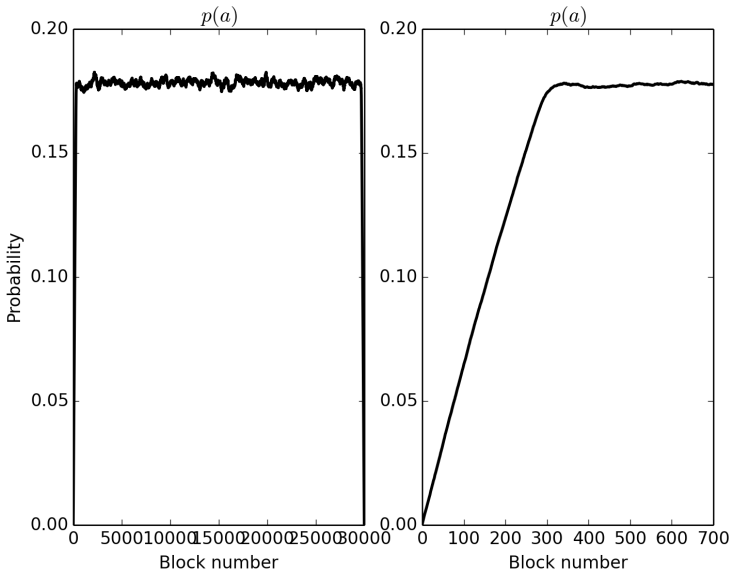

Figure 5: An estimate of $P(rel(b_i) \mid i)$ computed by stochastic simulation. The average size of generated files is 300 blocks, the disk block array consists of 30,000 blocks. The left plot shows the entire distribution. The right plot is a zoom-in of $P(rel(b_i) \mid i)$ to the initial segment of 700 blocks.

and random (uniform) distribution of starting positions of the files in the drive array[5].

Figure 6 shows the estimate of $P(rel(b_i) \mid i)$ using a slightly different experiment: we tested the content of b_{300} after each simulation and counted *only* simulations where b_{300} contains irrelevant data. Thus, Figure 6 models the effect of finding irrelevant data in a certain block on the probability distribution $P(rel(b_i) \mid i)$. Note that the shape of $P(rel(b_i) \mid i)$ to the right of the tested block is virtually identical to the shape of $P(rel(b_i) \mid i)$ in Figure 5 reaching maximal probability at $i = 300 + 300 = 600$. The shape of $P(rel(b_i) \mid i)$ to the left of the tested block is also similar to the overall shape of $P(rel(b_i) \mid i)$ in Figure 5, but the maximal probability is much lower because the segment of blocks between b_0 and b_{300} is quite small and, consecutively, the chance of a relevant file being there is low. Obviously, no file longer than 300 blocks could fit in $b_0 \ldots b_{300}$.

Figure 6 suggests that testing a block and finding irrelevant data in it can be viewed as splitting the disk block array into two parts with the *posteriori* distribution $P(rel(b_i) \mid i)$ of each part similar in shape to the *a-priori* distribution $P(rel(b_i) \mid i)$ of the initial array.

If a limit can be put on the *minimal* size of relevant files, Figure 6 suggests that the relevant files could be detected more efficiently by skipping over mul-

[5]Consider the following informal argument: the block b_0 can only be the first block of a continuous 300-block file, while the block b_{300} could be part of a continuous 300-block that starts in *any* of the positions $i = 0 \ldots 300$. Thus, b_{300} is 300 times more likely to be part of the relevant file than b_0, if all possible starting positions of the file are *equally* likely.

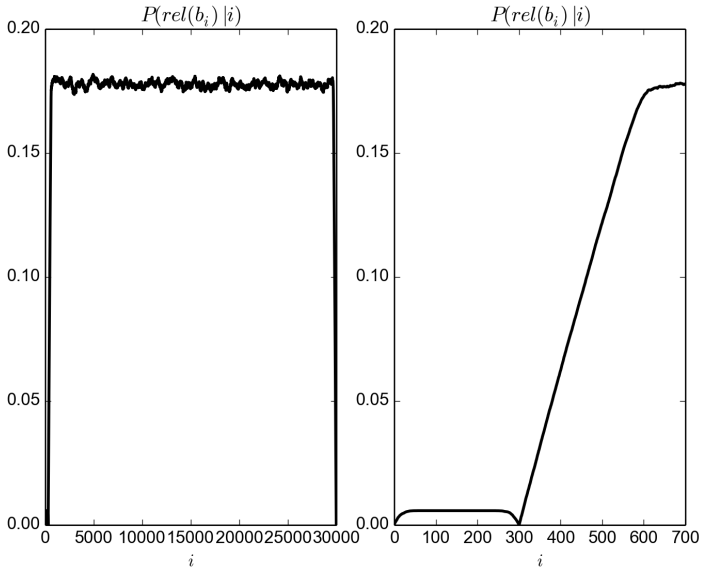

Figure 6: An estimate of $P(rel(b_i) \mid i)$ counting only simulations where block b_{300} was tested and found to contain *irrelevant* data.

tiple blocks while testing for relevant data instead of testing all blocks consecutively. The next secion shows that this is indeed the case, if the non-consecutive examination of blocks does not incur a significant time penalty.

4.3 Taking into account drive seek time

Explanation how to incorporate seek time penalty in our model by redefining utility function on the basis of carving time $WC(t)$...

Given that solid state drives are becoming more commonplace in forensic workstations, and taking into account that the majority of files in file system are not fragmented according to [19], we decided to build and evaluate a prototype carver that skips over unexplored blocks instead of examining them sequentially. It is described in Section 5.

5 Implementation and Evaluation of DECA carver

The previously described DECA algorithm was implemented using C++. The source code can be found at !!!!!!!! CHECK !!!!!!!!!!!!! https://bitbucket.org/pavel_gladyshev/deca. The software implementation defaults to linear carving mode. Two of the main functions of the software. First, a disk can be sampled to determine its seek time. With seek-time information, an investigator can determine jump distance at which DECA carving becomes faster than traditional linear carving. Second, the software allows for carving using the DECA algorithm while setting the minimal sector jump distance (in 512 sector increments).

Experimentation seeks to demonstrate the theorized performance of the

DECA algorithm, and compare the algorithm to other file carvers commonly used in digital investigations.

5.1 Experimentation Design

DECA is based on estimates of the distribution of JPEG data on a disk. Based on this estimation, DECA attempts to search only the most likely locations for related data. The result of the DECA algorithm is that some sectors will be skipped. In some cases, skipping sectors that are unlikely to contain relevant data may result in carving speed up. However, this method is also likely to miss smaller files.

DECA, in practical terms, can be described simply. If the jump size is high, carve times and number of carved files will decrease. If the jump size is low, carve times will increase and the number of carved files will increase. As discussed, this depends on the seek time of the disk holding the data to be analyzed. From the prior description, we propose that DECA is most beneficial when attempting to carve a small number of large files from a disk with a low seek time.

From the description of DECA we propose the following hypotheses for testing:

1. DECA will have a more significant reduction in run-time as the seek time of the disk decreases compared to other carving methods.

2. DECA will have a more significant reduction in run-time compared to other carving methods based on the size of disk/image being analyzed.

3. DECA will normally find a small number of files compared to linear carvers.

4. DECA is best suited for carving a small number of large files on a low-seek-time disk.

Two disk images were created for testing. One is based on a 4GB USB thumb drive, with a collection of JPEG images along with other documents stored in a FAT32 partition. The second is a 32GB image of a Windows XP operating system containing a user-created documents and JPEG images taken with various digital cameras.

Along with DECA, other carvers tested with the same data sets include Photorec, Scalpel 1.60, Scalpel 2.1 and Foremost. Each of these carvers were configured to only search for JPEG images.

5.2 Evaluation

Created disks with data from GovDocs1 [20] To test hypothesis 1, each carver was used on the test data set while stored on an sold state drive (SSD) and hard disk drive (HDD). The 4GB image was also tested inside a RAMDisk. RAM cache was cleared before each run. Measurement of carving performance is based on sampling the number of carved files in the carver's output directory (contained on the same disk as the test image) every 0.25 seconds until the carver terminates. All raw test data and testing scripts can be found in the project repository given above.

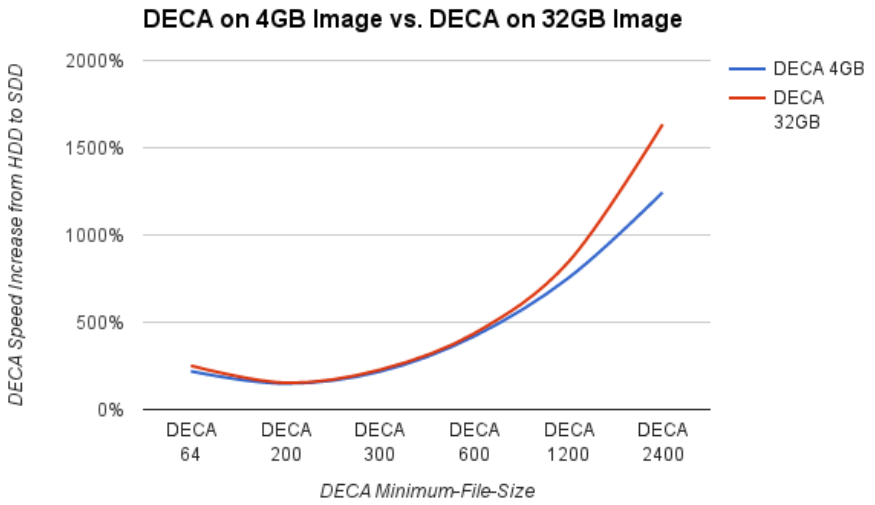

Figure 7: DECA time performance improvements in percentage when comparing carving times on an HDD to carving times on an SSD. This graph shows that DECA performs better with larger disk images when minimum-file-size is greater than 600.

To determine whether DECA had a more significant reduction in run-time compared to linear carving, we compared the average speed increase over all linear carvers (including DECA in linear carving mode) to the average speed increase between DECA with 7 minimum-file-size configurations (default, 64, 200, 300, 600, 1200, 2400).

The average speed increase for linear carving from HDD to SSD was 237%. The average speed increase for DECA carving from HDD to SSD was 553%. The average speed increase for linear carving from SSD to RAMDisk was 327%. The average speed increase for DECA carving from SSD to RAMDisk was 1229%. Overall, the average speed increase for linear carving was 298%, while the average speed increase for DECA carving was 891%. As expected, when DECA had lower minimum file-size values (64 to 300), performance was equal to or below linear carving on disks with high seek times. Hypothesis 1 is found to be true in most cases except when DECA minimum-file-size is less than 600 on high seek-time disks. The disk seek-time will affect the level of minimum-file-size that should be used.

To test hypothesis 2 we calculated the average speed increase for linear carving and DECA carving from HDD to SSD. For the 4GB image, DECA performed 316% over the linear carver average. For the 32GB image DECA performed 434% over the linear carver average. Similar to before, if DECAs minimum-file-size was below 600, DECA performed at or below the linear carver average for the 32GB drive. As can be seen in Figure 7, DECA saw greater performance increases from HDD to SSD with a larger disk image when minimum-file-size is greater than 600. This is explained by the cumulative effect of continually skipping large sections of the disk.

To test hypothesis 3 we counted the average number of files returned from each carving process. On the 4GB disk image, linear carving returned 356 files on average, and DECA carving returned 30 files on average (regardless of underlying disk). This is consistent with what was predicted since DECA is likely to skip small files and fragments that linear carving will find.

To test hypothesis 4, we examine results of the largest tested DECA minimum-file-size. In our experiments, the largest minimum-file-size used was 2400, which tells DECA that a file is likely to be located within 1,200Kb of its current location. With this setting, DECA was 5 times faster than the linear carving average on an SSD, and 19 times faster than the linear carving average on a RAMDisk. However, DECA only recovered 2% of the files recovered by linear carving. Overall, it appears that hypothesis 3 is also true. By expanding the minimum-file-size DECA accepts and using a disk with a lower seek-time, DECAs speed will dramatically increase at the expensive of smaller file recovery.

5.2.1 Discussion

The DECA algorithm worked as theorized, however, a practical implementation is likely more interesting to investigators. DECA produces significant carving speed increases under certain circumstances, but always at the cost to the number of files carved. Specifically, DECA is ideal for carving a small number of large files across a large, low-seek-time disk. For this reason, DECA may be appropriate for triage purposes, where very fast but incomplete image recovery is acceptable. In cases where the maximum number of files should be carved, or when carving high-seek-time devices (HDD), linear carving largely outperforms DECA.

As discussed, DECA jumps to sections of the disk based on a calculated data distribution. DECA then checks for an image header or footer. If not found, then DECA attempts to classify the data as a JPEG or not. The classifier that was developed could likely be improved. This would increase the number of images correctly recovered, potentially at the cost of speed.

The current implementation of DECA focuses on JPEG images, while imagining current digital cameras that produce images greater than 1 megabyte. However, this algorithm may be suitable for other specific file types, such as music, video, zip files and email containers. This method could also be used in conjunction with other carving methods, such as in-place carving [12, 13].

6 Conclusions

Decision theoretic analysis allows a file carver to consider the most likely locations of relevant data based on what is currently known about the distribution of data on the disk. By skipping sections of the disk that are unlikely to contain relevant data, carving times can be greatly decreased, where speed benefits are cumulative as the disk becomes larger. Through a practical implementation we demonstrated that decision theoretic analysis is useful when carving a small number of files from a large, low-seek-time disk. With current storage disk technology, decision theoretic carving (DECA) is best implemented as a triage solution. For traditional carving needs, linear carving gives more comprehensive results, and on standard hard disk drives, also has similar carving speeds.

6.1 Future Work

This work is a first effort to propose decision theoretic analysis applied to file carving. There are currently a number of limitations to the practical implementation (DECA) carver. Specifically, file distribution has been pre-calculated based on surveyed data in other drives. Future work will attempt to find a method for calculating a data distribution on the disk to be analyzed without increasing carving time significantly. Next, as DECA jumps to a location on the disk it must detect whether the data stored in that location is relevant. We have attempted several detection methods, most of which significantly increased the overall run time. We will seek to improve the data classifier that DECA uses.

References

[1] E. Casey, M. Ferraro, and L. Nguyen, "Investigation Delayed Is Justice Denied: Proposals for Expediting Forensic Examinations of Digital Evidence*," *Journal of Forensic Sciences*, vol. 54, pp. 1353–1364, nov 2009.

[2] M. M. Pollitt, "Triage: A practical solution or admission of failure," *Digital Investigation*, vol. 10, no. 2, pp. 87–88, 2013.

[3] F. Marturana and S. Tacconi, "A machine learning-based triage methodology for automated categorization of digital media," *Digital Investigation*, vol. 10, no. 2, pp. 193–204, 2013.

[4] B. H. Schell, M. V. Martin, P. C. Hung, and L. Rueda, "Cyber child pornography: A review paper of the social and legal issues and remediesand a proposed technological solution," *Aggression and violent behavior*, vol. 12, no. 1, pp. 45–63, 2007.

[5] J. I. James, A. Lopez-Fernandez, and P. Gladyhsev, "Measuring accuracy of automated parsing and categorization tools and processes in digital investigations," in *Digital Forensics and Cyber Crime*, pp. 147–169, Springer, 2014.

[6] A. Shaw and A. Browne, "A practical and robust approach to coping with large volumes of data submitted for digital forensic examination," *Digital Investigation*, vol. 10, no. 2, pp. 116–128, 2013.

[7] M. Rogers, "The role of criminal profiling in the computer forensics process," *Computers & Security*, vol. 22, no. 4, pp. 292–298, 2003.

[8] R. E. Overill, J. A. Silomon, and K. A. Roscoe, "Triage template pipelines in digital forensic investigations," *Digital Investigation*, vol. 10, no. 2, pp. 168–174, 2013.

[9] M. B. Koopmans and J. I. James, "Automated network triage," *Digital Investigation*, vol. 10, pp. 129–137, sep 2013.

[10] G. G. Richard III and V. Roussev, "Scalpel: A frugal, high performance file carver.," in *DFRWS*, 2005.

[11] C. Grenier, "Photorec," 2007.

[12] G. Richard, V. Roussev, and L. Marziale, "In-Place File Carving," in *Advances in Digital Forensics III* (P. Craiger and S. Shenoi, eds.), vol. 242, pp. 217–230, New York, NY: Springer New York, 2007.

[13] Meijer, Rob, "The carve path zero-storage library and filesystem," may 2012.

[14] S. L. Garfinkel and M. McCarrin, "Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb," *Digital Investigation*, vol. 14, pp. S95–S105, aug 2015.

[15] C. J. Veenman, "Statistical Disk Cluster Classification for File Carving," in *Third International Symposium on Information Assurance and Security*, pp. 393–398, IEEE, aug 2007.

[16] Q. Li, A. Ong, P. Suganthan, and V. Thing, "A novel support vector machine approach to high entropy data fragment classification," *SAISMC*, pp. 236–247, 2011.

[17] J. Ousterhout, P. Agrawal, D. Erickson, C. Kozyrakis, J. Leverich, D. Mazières, S. Mitra, A. Narayanan, G. Parulkar, M. Rosenblum, *et al.*, "The case for ramclouds: scalable high-performance storage entirely in dram," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 4, pp. 92–105, 2010.

[18] "Salvo Is New Game With a Nautical Air," *The Milwaukee Journal*, July 1931.

[19] S. L. Garfinkel, "Carving contiguous and fragmented files with fast object validation," *digital investigation*, vol. 4, pp. 2–12, 2007.

[20] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing science to digital forensics with standardized forensic corpora," *Digital Investigation*, vol. 6, pp. S2–S11, sep 2009.