Optimal Path Construction for Fragmented File Carving

H. Ying and V. Thing

Institute for Infocomm Research, Singapore
e-mail: {hmying,vriz}@i2r.a-star.edu.sg

Abstract

Fragmented File carving is an important technique in Digital Forensics to recover files from their fragments in the absence of the file system allocation information. In this paper, the fragmented file carving problem is formulated as a graph theoretic problem. Using this model, we describe two algorithms, “Optimal Carve” and “Probabilistic-based Carve”, to perform file reconstruction and recovery. Optimal Carve is a deterministic technique to recover the best file construction path. We show that this technique is more efficient and accurate than existing brute force techniques. The Probabilistic-based Carve technique involves a trade-off between the final score of the constructed path of the file and the file recovery time to allow a faster recovery process for highly fragmented files.

Keywords

Fragmented data carving

1. Introduction

The increasing reliance on digital storage devices such as hard disks and solid state disks for storing important private data and highly confidential information has resulted in a greater need for efficient and accurate data recovery of deleted files during digital forensic investigation.

File carving is the technique to recover such deleted files, in the absence of file system allocation information. However, there are often instances where files are fragmented due to low disk space, file deletion and modification. In a recent study (Garfinkel, 2007), FAT was found to be the most popular file system, representing 79.6% of the file systems analyzed. From the files tested on the FAT disks, 96.5% of them had between 2 to 20 fragments. This scenario of fragmented and subsequently deleted files presents a further challenge requiring a more advanced form of file carving techniques to reconstruct the files from the extracted data fragments.

The reconstruction of objects from a collection of randomly mixed fragments is a common problem that arises in several areas, such as archaeology (Kampel et al., 2001), (Sablatnig and Menard, 1997), biology (Stemmer, 1994) and art restoration (de Gama Leito and Soltfi, 2002), (de Gama Leito and Soltfi, 1998). In the area of fragmented file craving, research efforts are currently on-going. A proposed approach is known as the Bifragment gap carving(BGC) (Pal et al., 2008). This technique searches and recovers files, fragmented into two fragments that contain identifiable headers and footers. An idea of using a graph theoretic approach to perform file craving has also been studied in (Memon and Pal, 2006), (Pal et al., 2003), (Shanmugasundaram and Memon, 2003) and (Shanmugasundaram and Memon, 2002). In graph theoretic carving, the fragments are represented by the vertices of a graph and the edges are assigned weights which are values that indicate the likelihood that two fragments are adjacent in the original file. For example in JPEG files, we list two possible techniques to evaluate the candidate weighs between any two fragments (Memon and Pal, 2006). The first is pixel matching whereby the total number of pixels matching along the edges for the two fragments are summed. Each pixel value is then compared with the corresponding pixel value in the other fragment. The closer the values, the better the match. The second is median edge detection. Each pixel is predicted from the value of the pixel above, to the left and left diagonal to it (Martucci, 1990). Using median edge detection, we would sum the absolute value of the difference between the predicted value in the adjoining fragment and the actual value. The carving is then based on obtaining the path of the graph with the best set of weights. In addition, Cohen introduced a technique of carving involving mapping functions and discriminators in (Cohen, 2007), (Cohen, 2008). These mapping functions represent various ways for which a file can be reconstructed and the discriminators will then check on the validity of them until the best one is obtained. We discuss these methods further in Section 3 on related work.

In this paper, we model the problem in a graph theoretic form which is not restricted by the limitation of the number of fragments. We assume that all the fragments belonging to a file are known. This can be achieved through identification of fragments for a file based on groups of fragments belonging to an image of same scenery (i.e. edge pixel difference detection) or context based modeling for document fragments (Shanmugasundaram and Memon, 2003).

We define a file construction path as one passing through all the vertices in the graph. In a graph, there are many different possible file construction paths. An optimal path is one which gives the largest sum of weight (i.e. final score) for all the edges it passes through. The problem of finding the optimum path is intractable (Leiserson, 2001). Furthermore, it is well known that applying the greedy algorithm does not give good results and that computing all the possible paths is resource-intensive and not feasible for highly fragmented files. In this paper, we present two main algorithms namely the "Optimal Carve" and the "Probabilistic-based Carve". Optimal Carve is a "pairing" method of sub-paths which will reduce the required computations. This algorithm is more efficient and faster than brute force which computes all the possible path combinations. It is suitable for relative small values of n. For larger values of n, we introduce the Probabilistic-based Carve, which is a tradeoff algorithm to allow a flexible control over the complexity of the algorithm, while at the same time, obtain sufficiently good results for fragmented file carving. Our work is also applicable in modeling routing problems. For example, in telephone networks, internet routing as well as vehicle routing problems, our method can be applied in computing the optimal route based on edge scores.

The rest of the paper is organized as follows. In Section 2, we define the problem formally. In Section 3, we present the existing related work and in Section 4, we illustrate the main idea of our proposed methods. We describe the design of our Optimal Carve algorithm and present our evaluations of it in Section 5. In Section 6, we describe the Probabilistic-based Carve and its evaluation results, together with the implications for different ranges of values of n. We perform a comparison of the various algorithms in Section 7. We give an example of the implementation design in Section 8 and future work is described in Section 9. Conclusions follow in Section 10.

2. Statement of Problem

In fragmented file carving, the objective is to arrange a file back to its original structure and recover the file in as short a time as possible. The technique should not rely on the file system information, which may not exist (e.g. deleted fragmented file, corrupted file system). We are presented with files that are not arranged in its proper original sequence from its fragments. The goal in this paper is to arrange them back to its original state in a short a time as possible.

The core approach would be to test each fragment against one another to check how likely any two fragments are a joint match. They are then assigned weights and these weights represent the likelihood that two fragments are a joint match. If there are n fragments, there will be a total of $\frac{n(n-1)}{2}$ weights.

The problem can thus be converted into a graph theoretic problem where the fragments are represented by the vertices and the weights are represented by the edges. The goal is to find a file construction path which passes each vertex exactly once and has a maximum sum of edge weights, given the starting vertex. In this case, the starting vertex will correspond to the header.

A simple but tedious approach to solve this problem is to try all path combinations, compute their sums and obtain the largest value which will correspond to the path of maximum weight. Unfortunately, this method will not scale well when n is large since the number of computations of the sums required will be (n-1)!. This complexity increases exponentially as n increases. Another approach is to apply the greedy algorithm but very often the greedy algorithm does not yield the optimum path as we shall see in an example later.

3. Related Work

Bifragment gap carving was introduced as a fragmented file carving technique that assumed most fragmented files comprise of the header and footer fragments only. It exhaustively searched for all the combinations of blocks between an identified header and footer, while incrementally excluded blocks that result in unsuccessful decoding/validation of the file. A limitation of this method was that it could only support carving for files with two fragments. For files with more than two fragments, the complexity could grow extremely large.

Graph theoretic carving was implemented as a technique to reassemble fragmented files by constructing a k-vertex disjoint graph. Utilizing a matching metric, the reassembly was per-formed by finding an optimal ordering of the file blocks/sectors.

The different graph theoretic file carving methods are described in (Memon and Pal, 2006). The main drawback of the greedy heuristic algorithms was that it failed to obtain the optimal path most of the time. This was because they do not operate exhaustively on all the data. They made commitments to certain choices too early which prevented them from finding the best path later.

In (Cohen, 2008), the file fragments were "mapped" into a file by utilizing different mapping functions. A Mapping function generator generated new mapping functions which were tested by a discriminator. The goal of this technique was to derive a mapping function which minimizes the error rate in the discriminator. It is of great importance to construct a good discriminator for it to localize errors within the file, so that discontinuities can be determined more accurately. If the discriminator failed to indicate the precise locations of the errors, then all the permutations need to be generated which could become intractable.

4. Design of the Proposed Method on Paths Pairing

We state the assumptions that we use in our designs.

- All the fragments of the file are known since we are trying to reassemble the fragments of an incorrectly assembled file.
- The edge values give a good indication of how likely two files are adjacent to one another.
- The objective of our work is to devise a method to produce the optimum file construction path and yet achieve a lesser complexity than the brute force approach which requires the computation of all possible paths.

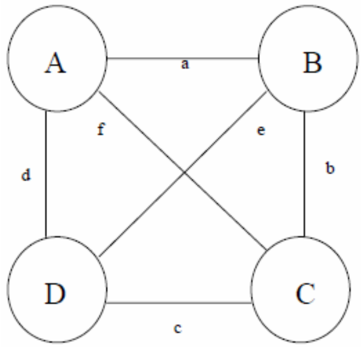

Figure 1: *n*=4 (General Case)

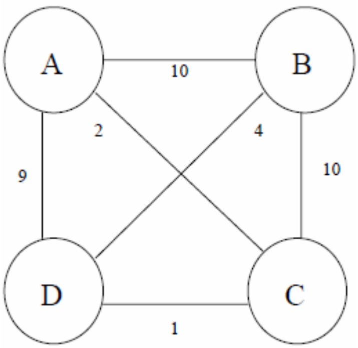

Figure 2: An example of *n*=4

In Figure 1, we show an example of a file with 4 fragments (n=4). A, B, C and D represent the file fragments. The letters, a to f, assigned to the edges represent the numbered values of the likelihood of a match between two adjacent fragments connected by that particular edge. Assume that A is the header fragment which can be easily identified. Let f(x) represent the sum of the edges of a path where x is a path. Computing the values of f(x) for all the possible paths, we obtain:

f(ABCD) = a + b + c, f(ABDC) = a + e + c, f(ACBD) = f + b + e, f(ACDB) = f + c + e, f(ADBC) = d + e + b, f(ADCB) = d + c + b

By observation, we see that to determine the greater total weight between f(ABCD) and f(ABDC), we only just need to compare the values between b and e. In this way, we can eliminate one of the paths immediately. In a similar fashion, we can match paths ACBD with ACDB and paths ADBC with ADCB and eliminate one path for each of them without actually doing any computations. Therefore, we can obtain three candidate paths and only need to perform three computations to find the optimal path.

In Figure 2, specific weights have been assigned to the edges. Applying the greedy algorithm, the output path is ABCD. f(ABCD) = 21 clearly is not optimal as the optimal path is ADBC which gives f(ADBC) = 23. Hence, the greedy algorithm may not achieve the correct file construction in this case.

Instead, if we perform path pairings, we can reduce the total computations and consider only three candidate paths to determine which one will give the maximum value of f(x). The three candidate paths are ABCD, ACBD and ADBC since b > e, b > c and e > c respectively. However, we can in this particular case, eliminate one more computation by performing a comparison between paths ACBD and ADBC, which depend on the values of f and d. In this case, d > f so we can eliminate path ACBD. Hence, we are left with only paths ABCD and ADBC to consider. By computing their f(x) values, we conclude that ADBC does indeed give the optimal path. Thus, by earlier eliminations, we only need to compute the f(x) values of two paths to accurately obtain the right one.

5. The Optimal Carve Algorithm

In this section, we describe our proposed deterministic algorithm to obtain an optimal path to achieve fragmented file carving. We label the vertices 1, 2, 3,......, n and let the weights of the edges be xi,j where i and j are the vertices that the edge connects to. Assuming that we identify the header as vertex 1, we let a, b, c be the three other vertices. If xa,b > xc,a, then f(1Sabc) > f(1Sacb) where S is a permutation of the vertices 2, 3,......., n excluding the selected a, b and c. In all, there are (n-4)! such permutations of S. Thus, we can eliminate all paths of the form 1Sacb and only compute those of the form 1Sabc. Therefore, for every path, there exists another path which can be eliminated by comparison. Hence, the maximum number of computations needed for this algorithm is $\frac{(n-1)!}{2}$, which is half that of a brute force approach.

For general n, a lower bound for the number of computations required to obtain the optimal path is (n-2)!. To see this, we define the special permutations of a path A in the following way. If A is a path denoted by 1v1v2.................. vn-3vn-2r, where vi are r are the non header vertices, then the special permutations of A are the paths 1v1v2........................ vn-3rvn-2, 1v1v2.................. vn-4rvn-2vn-3,.........., 1v1v2.......... vn-k-1rvn-2vn-3vn-k,.........., 1rvn-2vn-3......................................

v2v1. Therefore, we can observe that there are n-1 special permutations of a path. Let Xj be a permutation of v1v2................................. vn-2. There are (n-2)! possible values of j. We group the sets in such a way that each path of the form 1Xjr is grouped together with its special permutations for all (n-2)! possible values of j. It follows that that there will be a total of n-2 groups with each group containing n-1 paths. From this construction, all the possible paths will be in the (n-2)! sets, each containing n-1 paths.

For example, if we choose r = n, we can group the paths 12 3........................ (n-1)n, 123…. ......(n-2)n(n-1), 123........ (n-3)n(n-1)(n-2),....., 1n(n-1)(n-2)............. 432 into one group since 123.................. (n-2)n(n-1), 123....... (n-3)n(n-1)(n-2),....., 1n(n-1)(n-2)................ 432 are the special permutations of the path 123......................... (n-1)n. The other groups are formed by taking the special permutations of each path of the form 1Xjn. If we have xn-2,n-1 > xn-2,n, xn-3,n-2 > xn-3,n,....., x1,2 > x1,n, we can conclude that 123......(n-1)n will be the optimal path in that particular set. If each of xi,j > xk,n for all i, j, k < n then the optimal path will be of the form 1Xjn. In this case, the number of computation needed will be bounded by (n-2)!.

6. The Probabilistic-based Carve Algorithm

In the previous section, we introduced a deterministic way of obtaining the best path. It is suit-able for relatively small values of n where the computational complexity is minimal. For larger values of n, we propose a probabilistic algorithm which offers a tradeoff between obtaining the best path and the computational complexity.

The algorithm is described as follows. Out of the $\frac{n(n-1)}{2}$ edges in a graph of n fragments, we choose k largest edges, where k is between 1 and n-1 inclusive such that the three following conditions are satisfied.

- There is no cycle among any subset of the edges chosen. A cycle is defined as a closed loop path.
- At most 2 edges are connected to a vertex.
- The header vertex can only have at most 1 edge joining it.

The k edges are chosen in the following manner:

- Select the largest edge.
- Select the next largest edge. If this edge, together with all the previously chosen edges do not satisfy all the 3 conditions given above, then it is discarded and the next largest edge which satisfy all of the conditions is selected.
- Continue choosing the edges in this manner until k edges are selected.

All the paths that need to be computed are the paths that pass through these k edges. The optimal path value will then be taken as the best path. k is the parameter of the tradeoff. k = 0 corresponds to the brute force scenario.

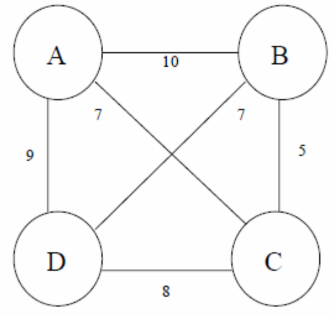

Figure 3: Choosing the edges in the probabilistic carve algorithm

An example in shown in Figure 3 where n=4. When k=1, the largest edge AB is chosen. The paths that need to be computed are all the paths that pass through AB, namely ABCD and ABDC. When k=2, the largest edge AB is chosen. The next largest edge AD has to be discarded since it violates condition 3. Hence, the edge CD is chosen. The set of edges {AB, CD} does not violate any of the conditions and hence the paths that are computed are those that pass through AB and CD. In this case, the only path that is considered is path ABDC.

Low values of k tend to result in a higher complexity but also a higher probability of obtaining a path that is close to optimal while high values of k tend to result in a lower complexity but also a lower probability of obtaining the best path. k can thus be adjusted to a suitable value in different scenarios of file reconstruction.

We conduct an evaluation and present the results to highlight the effectiveness of this algorithm for general n. k is set to 1 in this example.

We consider two cases as follow.

Case 1: The largest edge value joins the header.

In this case, out of the (n-1)! possible paths, (n-2)! of them can be discarded by comparison.

The probability of obtaining the best paths will be at least $\frac{(n\text{-}2)!}{(n\text{-}1)! - (n\text{-}2)!} = \frac{1}{n\text{-}2}$. The number of computations needed for this will be $\frac{(n\text{-}2)!}{2}$ by the path pairing method.

Case 2: The largest edge value does not join the header.

This case is in fact a more likely scenario when n is large. The total number of possible paths that will pass through this edge is $\frac{(n\text{-}1)!}{2}$. Out of the total number of (n-1)! paths, $\frac{(n-1)!}{2} - 2(n-3)!$ can be discarded by comparison. Hence, the probability of obtaining the best path in this case is at least $\frac{(n-1)!/2}{(n-1)!/2 + 2(n-3)!} = \frac{(n-1)(n-2)}{(n-1)(n-2)+4}$. The number of computations required in this case is $\frac{(n-1)!}{4}$.

Figure 4 shows the probability of obtaining the best path against the number of fragments when k=1 while Figure 5 shows the natural logarithm of the computational complexity against the number of fragments for the same k value. The probability and complexity are obtained by considering both cases 1 and 2 and evaluating the average based on the likelihood of occurrence of each case.

Hence, we show that this algorithm is capable of achieving reasonably good results and can be further optimized with a suitable value of k.

Without the assumption that all the fragments belonging to a file are known, there are still two immediate practical uses of the proposed carving methods. The first is that they will be able to determine in an automated manner whether a certain carving procedure recovers the file correctly. The second is they will be able to recover given files that have its fragments arranged in the wrong sequence. In this case, all the fragments of a file are implicitly known.

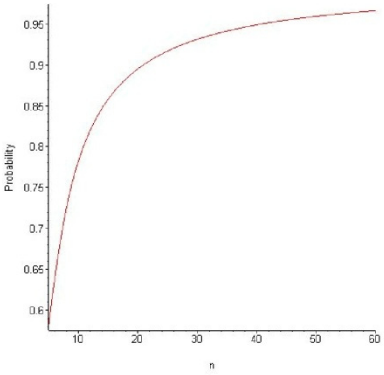

Figure 4: Probability of obtaining the best path against n

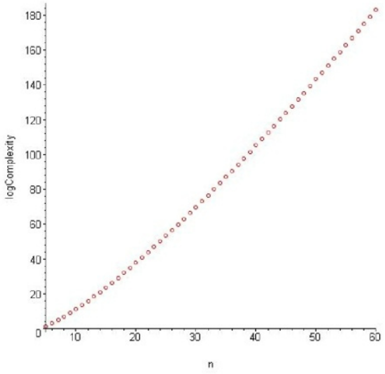

Figure 5: Complexity against n

7. Comparisons of Algorithms

In this section, we conduct an evaluation comparing the optimal-carve algorithm, the probabilistic-based carve algorithm for k = 1 and the greedy algorithm.

For general values of n, the probabilistic-based carve algorithm has a probability of $\frac{2}{n}$ that the largest edge is joined to the header and a probability of $\frac{n-2}{n}$ that it is not.

If we let n approach infinity, the average probability approaches 1. In this algorithm, the average complexity is $(n-2)!(\frac{n}{4} - \frac{3}{4} + \frac{3}{2n})$. Again, if we let n approach infinity, the complexity of the probabilistic-based carve algorithm is ¼ that of brute force and ½ that of the optimal-carve algorithm.

Next, we demonstrate cases where the probabilistic-based carve algorithm fails to obtain the best path. Figure 6 shows a case of n = 4 when both the greedy and probabilistic-based carve algorithm do not yield the optimal path. The optimal path is ADBC which can be obtained by using the optimal-carve algorithm but when we apply the greedy and probabilistic-based carve algorithm, we obtain the incorrect path ABDC.

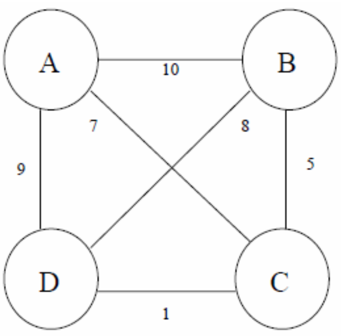

Figure 6: An example of *n*=4

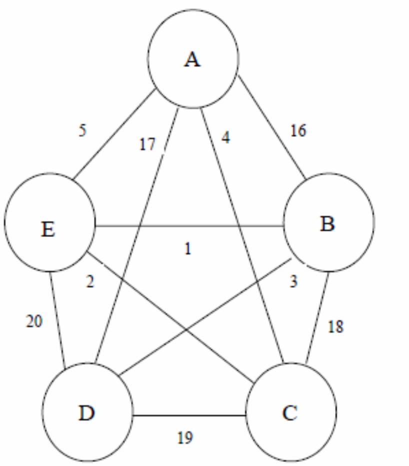

Figure 7: An example of *n*=5

In Figure 7, vertex A is the identified header. The optimal path in this case is ABCDE which gives f(ABCDE) = 73. The greedy algorithm will output path ADECB and its corresponding f(ADECB) = 57. The probabilistic-based carve algorithm will first identify the largest edge DE and compute all paths which pass through DE. It turns out that ABCDE will be one of the paths identified for computations and hence it will be picked out as the optimal path once all the computations have been carried out. Thus, in this case, the probabilistic-based carve algorithm identifies the best path whereas the greedy algorithm does not. The optimal-carve algorithm will also yield the optimal path. The above highlights one of the numerous cases where the probability algorithm is far likely to obtain the correct path compared to the greedy algorithm.

8. Implementation

In this section, we show an implementation of the probabilistic-based carve algorithm when k=1 and n=5.

Relative Weights of Edges	Paths to be Computed
x3,4 < x3,5 < x4,5	12354, 12453, 12543
x3,4 < x4,5 < x3,5	12354, 12453, 12534
x3,5 < x3,4 < x4,5	12345, 12453, 12543
x3,5 < x4,5 < x3,4	12345, 12435, 12543
x4,5 < x3,4 < x3,5	12354, 12435, 12534
x4,5 < x3,5 < x3,4	12345, 12435, 12534

Table 1: The largest edge joins the header

Label the header as vertex 1 and identify the largest edge. If the largest edge is connected to vertex 1, then the paths that need to be computed are given in table 1. If the largest edge is not connected to vertex 1, then the paths that need to be computed are given in table 2.

These tables for the various values of n can be pre-computed and stored. When the need arises to perform the carving, the tables can be used with the appropriate n and k values to do the carving for all possible configurations of edges.

Relative Weights of Edges	Paths Computed	Relative Weights of Edges	Paths Computed
x1,4<x1,5, x3,4<x3,5, x2,4<x2,5 x2,5 < x3,5, x2,4 < x3,4	15234,15324,12354 13254,14532,15432	x1,4>x1,5, x3,4<x3,5, x2,4<x2,5 x2,5 < x3,5, x2,4 < x3,4	14325,14235,12354 13254,14532,15432
x1,4<x1,5, x3,4<x3,5, x2,4<x2,5 x2,5 < x3,5, x2,4 > x3,4	15234,15324,12354 13254,14532,15423	x1,4>x1,5, x3,4<x3,5, x2,4<x2,5 x2,5 < x3,5, x2,4 > x3,4	14325,14235,12354 13254,14532,15423
x1,4<x1,5, x3,4<x3,5, x2,4<x2,5 x2,5 > x3,5, x2,4 < x3,4	15234,15324,12354 13254,14523,15432	x1,4>x1,5, x3,4<x3,5, x2,4<x2,5 x2,5 > x3,5, x2,4 < x3,4	14325,14235,12354 13254,14523,15432
x1,4<x1,5, x3,4<x3,5, x2,4<x2,5 x2,5 > x3,5, x2,4 > x3,4	15234,15324,12354 13254,14523,15423	x1,4>x1,5, x3,4<x3,5, x2,4<x2,5 x2,5 > x3,5, x2,4 > x3,4	14325,14235,12354 13254,14523,15423
x1,4<x1,5, x3,4<x3,5, x2,4>x2,5 x2,5 < x3,5, x2,4 < x3,4	15234,15324,12354 13245,14532,15432	x1,4>x1,5, x3,4<x3,5, x2,4>x2,5 x2,5 < x3,5, x2,4 < x3,4	14325,14235,12354 13245,14532,15432
x1,4<x1,5, x3,4<x3,5, x2,4>x2,5 x2,5 < x3,5, x2,4 > x3,4	15234,15324,12354 13245,14532,15423	x1,4>x1,5, x3,4<x3,5, x2,4>x2,5 x2,5 < x3,5, x2,4 > x3,4	14325,14235,12354 13245,14532,15423
x1,4<x1,5, x3,4<x3,5, x2,4>x2,5 x2,5 > x3,5, x2,4 > x3,4	15234,15324,12354 13245,14523,15423	x1,4>x1,5, x3,4<x3,5, x2,4>x2,5 x2,5 > x3,5, x2,4 > x3,4	14325,14235,12354 13245,14523,15423
x1,4<x1,5, x3,4>x3,5, x2,4<x2,5 x2,5 < x3,5, x2,4 < x3,4	15234,15324,12345 13254,14532,15432	x1,4>x1,5, x3,4>x3,5, x2,4<x2,5 x2,5 < x3,5, x2,4 < x3,4	14325,14235,12345 13254,14532,15432
x1,4<x1,5, x3,4>x3,5, x2,4<x2,5 x2,5 > x3,5, x2,4 < x3,4	15234,15324,12345 13254,14523,15432	x1,4>x1,5, x3,4>x3,5, x2,4<x2,5 x2,5 > x3,5, x2,4 < x3,4	14325,14235,12345 13254,14523,15432
x1,4<x1,5, x3,4>x3,5, x2,4<x2,5 x2,5 > x3,5, x2,4 > x3,4	15234,15324,12345 13254,14523,15423	x1,4>x1,5, x3,4>x3,5, x2,4<x2,5 x2,5 > x3,5, x2,4 > x3,4	14325,14235,12345 13254,14523,15423
x1,4<x1,5, x3,4>x3,5, x2,4> x2,5 x2,5 < x3,5, x2,4 < x3,4	15234,15324,12345 13245,14532,15432	x1,4>x1,5, x3,4>x3,5, x2,4>x2,5 x2,5 < x3,5, x2,4 < x3,4	14325,14235,12345 13245,14532,15432
x1,4<x1,5, x3,4>x3,5, x2,4>x2,5 x2,5 < x3,5, x2,4 > x3,4	15234,15324,12345 13245,14532,15423	x1,4>x1,5, x3,4>x3,5, x2,4>x2,5 x2,5 < x3,5, x2,4 > x3,4	14325,14235,12345 13245,14532,15423
x1,4<x1,5, x3,4>x3,5, x2,4>x2,5 x2,5 > x3,5, x2,4 < x3,4	15234,15324,12345 13245,14523,15432	x1,4>x1,5, x3,4>x3,5, x2,4>x2,5 x2,5 > x3,5, x2,4 < x3,4	14325,14235,12345 13245,14523,15432
x1,4<x1,5, x3,4>x3,5, x2,4>x2,5 x2,5 > x3,5, x2,4 > x3,4	15234,15324,12345 13245,14523,15423	x1,4>x1,5, x3,4>x3,5, x2,4>x2,5 x2,5 > x3,5, x2,4 > x3,4	14325,14235,12345 13245,14523,15423

Table 2: The largest edge does not join the header

9. Future Work

We are currently exploring new techniques to obtain the optimal path when two directed weights assigned to an edge are taken into account. This will model the file carving problem even more accurately. Statistical methods are being used to perform this analysis. Evaluations of the bounds are also performed to give a good estimate on the complexity.

10. Conclusions

In this paper, we modeled the file recovery problem using a graph theoretic approach. We briefly introduced the techniques of brute force and the greedy algorithm and followed up by explanations and illustrations on why in most instances, they are inefficient or inaccurate for practical purposes. We proposed two new algorithms to perform fragmented file recovery. The first algorithm, optimal-carve is suitable for files which have been fragmented into a small number of fragments. This algorithm results in an optimal file path construction in only half the time required by the brute force technique. The second algorithm, probabilistic-based carve is applicable in the cases where a file is fragmented into a large number of fragments. It introduces a trade-off between time and success rate of optimal path construction. This flexibility enables a user to adjust the settings according to his available resources. Analysis of this trade-off technique reveals that for k=1 and large n, the algorithm has a complexity ¼ that of the brute force technique and ½ that of the optimal-carve algorithm while still maintaining a high probability of recovering the optimal path. Therefore, the newly proposed algorithms show very promising results that will aid in the fragmented file carving required in the digital forensics investigations.

11. References

Cohen M.I. (2007), “Advanced carving techniques”, In Digital Investigation, 4(Supplement 1):2-12.

Cohen M.I (2008), “Advanced jpeg carving”, In Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop.

da Gama Leito H.C. and Soltfi J (2002), “A multiscale method for the reassembly of two-dimensional fragmented objects”, In IEEE Transections on Pattern Analysis and Machine Intelligence, vol. 24.

da Gama Leito H.C. and Soltfi J (1998), “Automatic reassembly of irregular fragments”, In Univ. of Campinas, Tech. Rep. IC-98-06, 1998.

Garfinkel S. (2007), “Carving contiguous and fragmented files with fast object validation”, In Proceedings of the 2007 digital forensics research workshop, DFRWS, Pittsburgh, PA, August 2007.

Kampel M., Sablatnig R. and Costa E (2001), “Classification of archaeological fragments using profile primitives”, In Computer Vision, Computer Graphics and Photogrammetry - a Common Viewpoint, Proceedings of the 25th Workshop of the Austrian Association for Pattern Recognition (OAGM), pages 151–158, 2001.

Leiserson C.E. (2001), “Introduction to algorithms”, MIT Press.

Martucci S.A. (1990), "Reversible compression of hdtv images using median adaptive prediction and arithmetic coding", In IEEE International Symposium on Circuits and Systems, pages 1310–1313, 1990.

Memon N. and Pal A. (2006), "Automated reassembly of file fragmented images using greedy algorithms", In IEEE Transactions on Image processing, pages 385–93, February 2006.

Pal A., Sencar H.T. and Memon N. (2008), "Detecting file fragmentation point using sequential hypothesis testing", In Proceedings of the Eighth Annual DFRWS Conference. Digital Investigation. Volume 5, Supplement 1, pages S2–S13, September 2008.

Pal A., Shanmugasundaram K. and Memon N. (2003), "Automated reassembly of fragmented images", Presented at ICASSP, 2003.

Sablatnig R. and Menard C. (1997), "On finding archaeological fragment assemblies using a bottom-up design", Proceedings of the 21st Workshop of the Austrain Association for Pattern Recognition Hallstatt, Austria, Oldenburg, Wien, Muenchen, pages 203-207, 1997.

Shanmugasundaram K. and Memon N. (2003), "Automatic reassembly of document fragments via context based statistical models", In Proceedings of the 19th Annual Computer Security Applications Conference, page 152, 2003.

Shanmugasundaram K. and Memon N. (2002), "Automatic reassembly of document fragments via data compression, presented at the 2nd digital forensics research workshop, syracuse", July 2002.

Stemmer W.P. (1994), "DNA shuffling by random fragmentation and reassembly: in vitro recombination for molec-ular evolution", In Proceedings Natl Acad Sci U S A, October 25, 1994.