ISSN 2278-3091
Volume 9, No.5, September - October 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse45952020.pdf
https://doi.org/10.30534/ijatcse/2020/45952020

The File Carving Method for Category B Images

[1]Dr. K. Srinivas, [2] Dr. T. Adilaxmi
[1]Associate Professor, Department of Computer Science & Engineering, Vasavi College of Engineering, Hyderabad ,India. Email: srinivas.kaprthi@staff.vce.ac.in
[2]Professor and Head, Department of Computer Science & Engineering, Vasavi College of Engineering, Hyderabad, India. Email: hodcse@staff.vce.ac.in

ABSTRACT

File Carving is a method of recovering deleted files without using file-system tables. This method reassembles file fragments to prepare a recovered file. In the literature we find the methods of carving image files from its fragments on a storage media, without using files' metadata in file-system data structures [1, 2]. These methods require that the cluster containing the header of an image be available. In this paper we propose methods that can carve image files from its fragments when a header is corrupted or missing. Two different cases of this problem have been considered; 1) Only header is missing 2) A cluster containing header is missing. We have proposed an algorithm called as extended GSUP algorithm which is an extension of algorithms presented in a paper [1, 2]. This algorithm is implemented in C++ and the experimental results are also presented. We have used the ULFS tool [3, 4] for preparing input for testing our carving tool. This tool is also used to compare the internal structure of a header of a bitmap file with the pseudo header constructed for the purpose of carving.

Key words: File-carving, user-level-file-system, digital forensics, data-recovery, image-header

1. INTRODUCTION

When a user saves a file on a disk, the Operating System uses its File System component to handle it. A File System is a set of software modules at kernel level for file handling operations. Assume that a user has created a file named as "one.txt" containing the text "abc". The size of this file is 3 bytes. The file system allocates one free clusters for this new file, at the time of its creation, from the pool of free clusters that it maintains. A cluster is a set of consecutive sectors on the disk. A cluster is an allocation unit. The kernel file system views the disk as a set of clusters than as a set of bytes. When a user creates a new file, the required number of free clusters is allocated for it. And when a user deletes a file, all the used clusters by the file are freed. So, for the above "one.txt" file, one cluster is allocated. Thus when the properties of the file "one.txt" are viewed, for example, on Windows 7 Operating System, we notice file 'size' as 3 bytes and *'size on disk'* as 4096 bytes. In this paper we assume the cluster size as 4096 because it is the most common size but it can vary [5].

Consider a file named "x.jpg" of size 10KB on the disk saved at cluster numbers 205, 230 and 340. In *conventional method*, to perform read operation on this file, the file system obtains these cluster numbers (that were saved in the file system's data structures when the file was created, as shown in Fig 1), reads data from these clusters and presents the data to user application [6]. It is up to user application how to interpret this data.

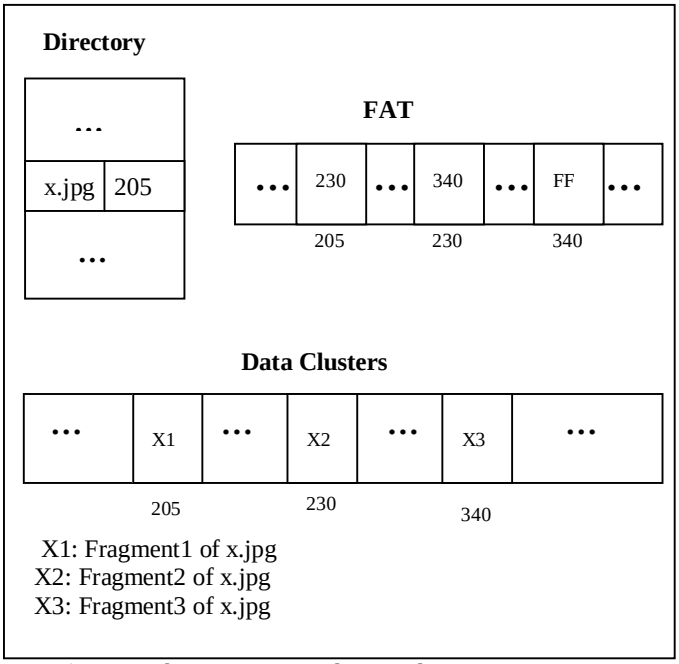

Figure 1: The DIR, FAT and Data Clusters sections before

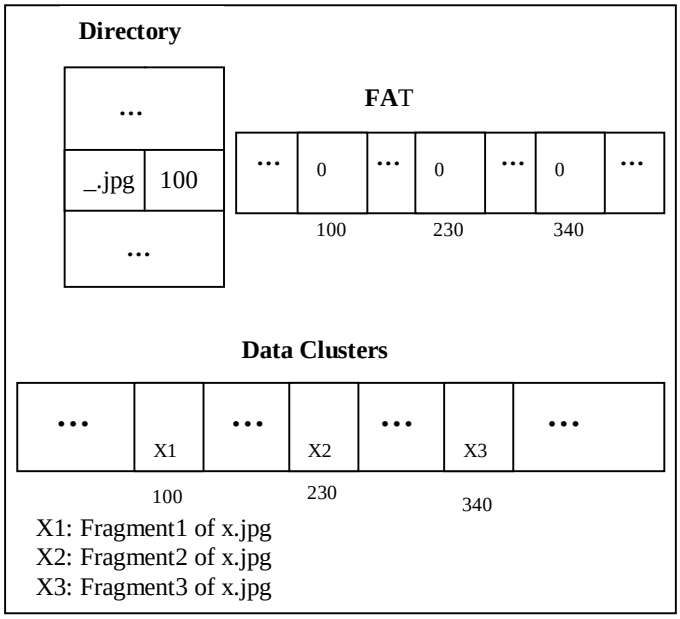

Figure 2: The file system tables after delete operation

This research work is sponsored by Vasavi College of Engineering, Hyderabad.

When the file is deleted, the File System changes the first byte of the file name of x.jpg to '_'. Then it stores a zero in each of the FAT locations at indexes 200, 230 and 340 to indicate that the clusters 205, 230 and 340 are free now. The actual data of a file x.jpg is *not erased* [4].

In an unconventional method of accessing files, file fragments are to be reassembled in the absence of metadata in file system data structures as shown in Figure 2.

Unconventional methods are applied under three different situations. A) When files were deleted accidentally and they need to be recovered. B) When file(s) were deleted by a criminal intentionally to escape from the law for his criminal activities and investigating agencies want to view such files [1]. C) When file system data structures such as DIR and FAT got corrupted and the files present on the disk to be read. Under these three conditions the conventional method cannot be used.

To face the above situations technically in the areas of data recovery and digital forensics, a new technology known as file carving has evolved. In file-carving file-system tables are not used. This method reassembles relevant file-fragments to recover a file [5].

In a conventional method of reading a file, the file system refers to file-system's data structures, reads the data present in clusters and then presents the data to the application at user level. File carving is a method of recovering deleted files without using file system tables. The area of image processing has wide applications like medical field, security etc [11,12]. The digital forensics is one among them.

The presentation of our work is planned, in this paper, as follows. In section II, we present review of literature. In section III, we present the analysis of the missing header problem. In section IV, we present the design and implementation of the image carving system utilizing our proposed algorithm. In section V we present the details of the experiments conducted and the results obtained to prove our research work.

2. REVIEW OF LITERATURE

Pal et al have presented an unconventional method of accessing image files by using greedy algorithms [1]. This method initially identifies a cluster containing header of an image to find image attributes such width, height, file size etc. The process of finding adjacent clusters of an image utilizes width attribute. File size attribute helps in calculating number of clusters that the image spans on the disk. This method addresses the major challenge faced in carving process, that is, file fragmentation. In a research paper [2], sum-of-difference (SoD) measure used in [1] is enhanced by inventing a new measure known as Coherence of Euclidian Distance (CED). In a research paper [7], the problem of missing fragments is addressed for a file type of JPEG.

In paper [5] a method for carving i-node in Linux is introduced. Almost without using the information contained in super block and based on use cases i-nodes are carved. In year 2009, Pal et al presented state of research in the area of file carving [6]. In the literature we also find the progress in research in accessing files of various types such as executable files, zip files, portable document files, html files, document files, JPEG files etc.[5,8, 9] using unconventional methods famously known as file carving. In research paper [3], an implementation of user-level- file-system by following broadly a FAT file system is presented. It is a tool for generating input for file carving algorithms by executing a sequence of commands at its command prompt is presented. In research paper [4] the authors have described how to write a script file for creating a virtual disk suitable for testing a file carving algorithm. We have utilized ULFS tool by writing script files to generate the input for file carving tool as described in [3, 4]. In the paper [10], camera sensor noise is used to address the issue of missing fragments problem in file recovery.

In this paper, we propose methods for reassembling bitmap file fragments with missing both header and files' metadata in file system data structures.

3. THE MISSING HEADER PROBLEM

We analyze the missing header problem in this section. We consider bitmap (24-bit) files for the analysis. We assume that image header is not available. We also do not use file-system tables. As outlined in section II, in the literature we find greedy algorithms for reassembling file fragments of bitmap image type in the absence of file's metadata in file-system data structures but when header of the image file is available. One such algorithm is Greedy Sequential Unique Path Algorithm. We propose a method by which a fragmented image of bitmap-24-bit type can be accessed to its maximum when a header and file metadata in file-system data structures both are not available. We present the analysis of this problem in the following sub sections.

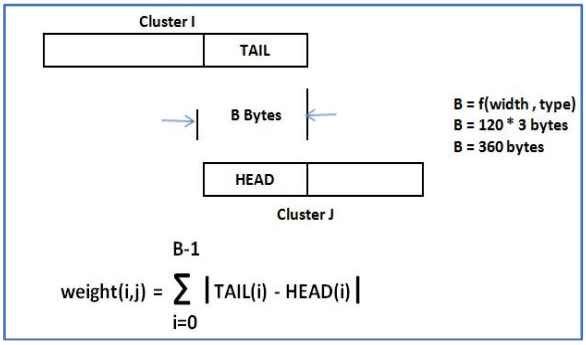

Figure 3: Calculation of weight

3.1 The Overview of the GSUPA and its Limitations

The steps of GSUPA algorithm are as follows. 1) Identify a cluster that contains image header. Let us denote such a cluster as H. If starting bytes in a cluster are "BM" then it is an image header cluster. 2) Decode the image header. We get file-size, width, height etc. attributes. 3) Identify the file fragments and reassemble them to recover a file. The file-size attribute helps to find the count of clusters of the image file. The width attribute helps in reassembling the relevant clusters.

To recover a deleted image the algorithm constructs a graph G. The unallocated clusters are the candidates for the image fragments of a deleted file. So in the graph G, a node represents an unallocated cluster. And an edge <I,J>

represents the likelihood of the cluster J that followed the cluster I in an original image. The group of adjacent pixel pairs is saved across clusters. The smoothness property helps in reassembling all such clusters together.

Based on this, the weights are calculated by using the equation shown in the Figure 3. For a recovered cluster I, to find the adjacent cluster, it finds N number of weights namely weights(I,K) where K = 0, 1, 2, … , N-1. The adjacent cluster is a cluster J such that weights(I,J) is minimum.

GSUPA generates a matrix shown in Figure 4, when the disk contains two images I_1 and I_2, each spanning five clusters on the disk, at cluster numbers (0, 1, 2, 3, 4) and (5, 6, 7, 8, 9) respectively, with headers at cluster number 0 and 5 respectively. This is the case of contiguous images on the disk. But files may not always be saved in contiguous areas on the disk i.e. the files may be fragmented on the disk. The GSUPA addresses this challenging issue of fragmentation. The elements of the matrix are divided into two categories namely near-zero values (NZs) and not-near-zero values (NNZs). In the Figure NNZs are not shown for simplicity. *We note that GSUP algorithm has number of iterations and in every iteration it generates the matrix for reassembling all the images with equal width.* With this matrix as an input, it reassembles the two images as (0, 1, 2, 3, 4) and (5, 6, 7, 8, 9). Similarly it reassembles one image as (0, 1, 13, 3, 14) when matrix in Figure 5 is input in its image reconstruction phase because of NZs at the cells (0,1), (1,13), (13,3) and (3,14). But the existence of more NZs indicates the existence of more images on the disk that the GSUP ignores. This situation arises when header is corrupted / missing on the disk and GSUP cannot consider these additional NZs.

3.2 The Image File Carving with Missing Headers

The idea here is to construct a partial greedy path. This operation uses the residual NZs in the adjacency matrix of the graph G. The partial greedy path is 7->9->10->12. The full greedy path is constructed by using the header cluster of the 2nd image. The full greedy path is 0->7->9->10->12.

Header Cluster=0 & 5 , Image1=(0,1,2,3,4) Image2=(5,6,7,8,9)

	0	1	2	3	4	5	6	7	8	9
0		NZ								
1			NZ							
2				NZ						
3					NZ					
4										
5							NZ			
6								NZ		
7									NZ	
8										NZ
9										

Figure 4: The matrix generated by GSUP Algorithm

It represents the partially recovered image. From forensics point of view this partially recovered image may turn out to be an important clue to crack the crime.

The GSUPA constructs the adjacency matrix of the graph shown in Figure 5. This adjacency matrix represents the two fragmented images on a storage media. They are image1 and image2. The OS had allocated five clusters for each image.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0		NZ													
1														NZ	
2															
3															NZ
4															
5															
6															
7										NZ					
8															
9											NZ				
10													NZ		
11															
12															
13				NZ											
14															

Figure 5: The matrix generated by GSUP algorithm and its residues are marked by encircling them

(Weight(A,B) = NZ) => B is an adjacent of A (1)
Pred(A) = A' | Weight(A',A)=NZ (2)
Succ(B) = B' | Weight(B,B')=NZ (3)

The algorithm begins with one of the remaining NZs. The remaining NZs are at the cells (7, 9), (9,10) and (10, 12). Let the starting cell is (A, B)=(9,10). In an original image, the image fragment in cluster B follows the image fragment in cluster A. Now the partial greedy path can be found. Pred(A)=7 and Succ(B)=12. So the partial greedy path is 7->9->10->12.

If the starting cell is (A, B)=(7, 9) then Pred(A) is NULL Succ(B) is 10. And then the Succ(10) is 12. So the partial greedy path is 7->9->10->12.

If the starting cell is (A, B)=(10, 12) then Pred(A) is 9 Succ(B) is NULL. And then the Pred(9) is 7. So the partial greedy path is 7->9->10->12.

4. DESIGN AND IMPLEMENTATION

The data structure and an algorithm to operate on the data structure to reassemble the images without headers are presented below.

4.1 Using a sparse matrix for leftover NZs

The count of the remaining NZs is much less than the size of the adjacency matrix. So a sparse matrix is the best data structure for these NZs. We can store all the residual NZs very efficiently by using M X 3 sparse matrix where M is number of NZs that have not been used by GSUP Algorithm. For the matrix shown in Figure 3, the sparse matrix is shown in Table 1.

TABLE 1. Sparse Matrix for image2 clusters in Figure 3

ROW	COLUMN	VALUE
7	9	NZ
9	10	NZ
10	12	NZ

The file carving algorithm uses row and column values only. It does not use the third column. So it can be

dropped. Therefore the algorithm uses the below mentioned sparse matrix.

TABLE 2. Reduced size Sparse Matrix for image2 clusters in Figure 3

ROW	COLUMN
7	9
9	10
10	12

The file carving algorithm uses the following set of equations.

The row S[I] => S[I,2] is an adjacent of S[I,1] (4)

$$\text{Pred}(S[I,1])=\begin{cases} S[J,1], \ when\ J \mid S[I,1] = S[J,2] \\ nil, \ when\ there\ is\ no\ such\ J \end{cases} \quad (5)$$

$$\text{Succ}(S[I,2]) = \begin{cases} S[J,2], \ when\ J \mid S[I,2] = S[J,1] \\ nil, \ when\ there\ is\ no\ such\ J \end{cases} \quad (6)$$

$$\text{Pred}(S[I,2]) = S[I,1] \quad (7)$$

$$\text{Succ}(S[I],1) = S[I,2] \quad (8)$$

Table 3: The MH matrix of missing header problem

	54-bytes header missing	**Whole cluster missing**
Some images	Category A	Category B
All images	Category C	Category D

The missing header problem has four categories. They are shown in the Table 3. The name given to this matrix is MH matrix (the long form is Missing Header Matrix). In category A, only the header bytes in the header cluster are corrupted for some images. In category B, header clusters are missing for some images. In category C, only the header bytes are corrupted for all the images. And in category D, header clusters are missing for all the images.

4.2 Extended GSUP Algorithm to Reconstruct Images with Missing Headers of category B

Now we explore the method to carve the images of category B, having no header but having all the remaining data. It is a two step process. The first step is to use one of the existing headers as a header for carving images without header. The second step is to find the sequence of clusters using the data in the sparse matrix as shown in the Figure-6 below.

The values in Table 2 i.e., 7, 9, 10 and 12 are cluster numbers. There are 3 rows in the sparse matrix S of Table 2 representing 3 residual NZs. The row containing the cluster numbers 7 and 9 (i.e. row S[1]) represent a cell (7,9) in the matrix in Figure 3. Similarly S[2] and S[3] represent the cells (9,10) and (10,12).

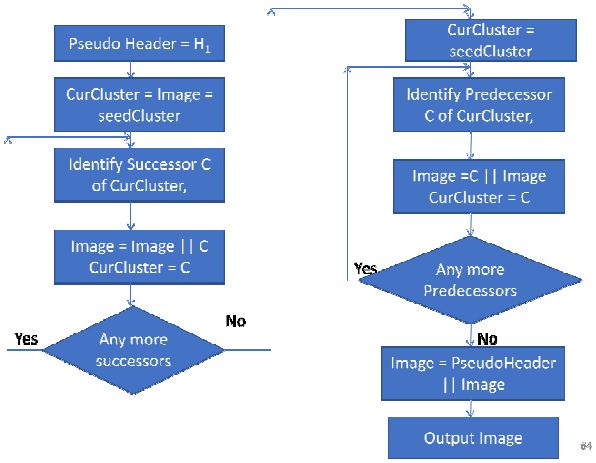

Figure 6: Logic of reassembling an image with missing header

The values in Table 2 i.e., 7, 9, 10 and 12 are cluster numbers. There are 3 rows in the sparse matrix S of Table 2 representing 3 residual NZs. The row containing the cluster numbers 7 and 9 (i.e. row S[1]) represent a cell (7,9) in the matrix in Figure 3. Similarly S[2] and S[3] represent the cells (9,10) and (10,12).

Using one of the clusters from the sparse matrix as a seed cluster, C_{seed} and by applying the above mathematics we find the sequence of clusters. Then we append one by one clusters data of this sequence of clusters, to the cluster containing pseudo header to make up an image file.

The following class named as IRWHM is used to implement the above plan.

```
class IRWHM
{
 public:
  IRWHM(int s[MAXCL][3],int rc,uc psuedohd[]);
  int sparse[MAXCL][3] ;
  int maxnz ;
  int rowcount ;
  int pathlength ;
  int noheadercnt ;
  int paths[MAX_IMGS][MAX_CLS_PER_IMG] ;
  void buildhdr() ;
  int prev(int) ;
  int next(int) ;
  int get_a_path(int seedcl , int p[]) ;
  int getpaths() ;
  void reconstruct(uc *) ;
  void construct_header(ul w, ul h, uc pseudo[]);

} ;
```

The member function prev() is used to implement the equations (5) and (7). The member function next() is used to implement the equations (6) and (8). The member function get_a_path() implements the first step of preparing the sequence of clusters given a seed cluster. It implements the logic of Figure 6. The member function get_paths() is for reassembling all possible sequences. The member function reconstruct() reassembles all the images from the sequences by using the existing header of another image for constructing each image. The row count data member is to store the number of rows in the sparse matrix.

4.3 Extended GSUP Algorithm to Reconstruct Images with Missing Headers of Category A

In this case we copy one of the existing headers is copied to the first cluster of the sequence as a header. In this case we will not see another image's portion at the bottom of the carved images as we will see in the next section.

5. EXPERIMENTS AND RESULTS

We prepare input for our experiments by using ULFS tool. Any file carving software verifies a disk that is suspected to have unlawful data. For our experiments we create a virtual-disk using ULFS tool. So virtual-disk is the input for our software.

The ULFS tool has CUI (Character User Interface). The tool provides a set of commands. The user can create a new Virtual disk. The user types in "newdisk" command at the prompt for this purpose. The "newdisk" command has two arguments. They are name and capacity of the virtual disk to be created. For example, the command line "newdisk vd 2" creates a virtual disk "vd" of the capacity 2MB.

The other commands of this tool are;
create -> To create a new file
del -> To delete a file
directory-> To display the files list
clslist -> To display the clusters' list
isfrag -> To know whether the file is fragmented or not
cb -> To corrupt bytes
format -> To format the virtual disk

5.1 The experiments to recover Category B Images

The script to create a virtual disk for category B is - newdisk one.dsk 1; create s0.txt; create s1.txt; create 1.bmp; create s2.txt; del s0.txt; create 2.bmp; del s1.txt; del s2.txt; create 3.bmp; cb 2 0 4096; cb 3 0 4096; format;.

The files s?.txt are small text files. We could have used any other file type. They are used to fragment the images files. The sequence in the script is so chosen that the virtual disk has the layout of category-B images. The Figure 7 shows the effect of each command of the script. This figure shows the progressive development of category-B images. It also shows the final layout of the data on the virtual disk. The "format" command deletes the system tables namely File Allocation Table and the directory table.

Cluster Number→																							
Command ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
newdisk one.dsk 1																							
create s0.txt	D	F	s0																				
create s1.txt	D	F	s0	s1																			
create 1.bmp	D	F	s0	s1	1	1	1	1	1	1	1												
create s2.txt	D	F	s0	s1	1	1	1	1	1	1	1	s2											
del s0.txt	D	F		s1	1	1	1	1	1	1	1	s2											
create 2.bmp	D	F	2	s1	1	1	1	1	1	1	1	s2	2	2	2	2	2	2					
del s1.txt	D	F	2		1	1	1	1	1	1	1	s2	2	2	2	2	2	2					
del s2.txt	D	F	2		1	1	1	1	1	1	1		2	2	2	2	2	2					
create 3.bmp	D	F	2	3	1	1	1	1	1	1	1	3	2	2	2	2	2	2	3	3	3	3	3
format			2	3	1	1	1	1	1	1	1	3	2	2	2	2	2	2	3	3	3	3	3
D: Directory	F: FAT				1: 1.bmp cluster							2: 2.bmp cluster							3: 3.bmp cluster				

Figure 7: The progressive development of Category B images on the Virtual Disk "ONE.DSK"

The Figure 8 shows the original files. The Figure 9 shows the files that are carved by the proposed file-carving algorithm. We have used the adjacency list for the graph in our experiments. The Figure 10 shows the weights of the graph G.

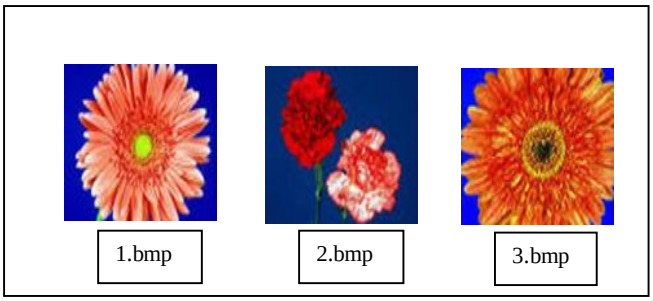

Figure 8: The Category B images (original) on the virtual disk "ONE.DSK"

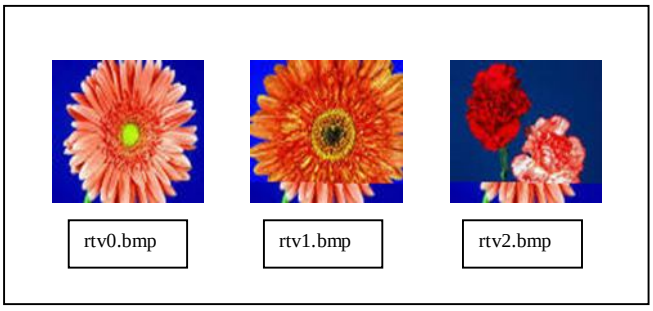

Figure 9: The three recovered images. (The two images are partially recovered. These partially recovered images are also forensically important)

The GSUPA would recover only 1.bmp image file. The proposed algorithm has recovered all the three images. Though two of them are partially recovered they are forensically important. The proposed algorithm used the header cluster of the fully recovered image for the partially recovered images. Therefore in Figure 9, the bottom portions of the second and third images are that of the first image.

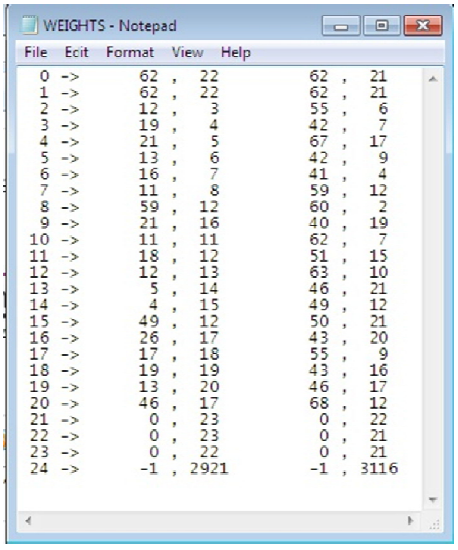

Figure 10: Portion of Adjacency List of the Graph Generated for Images of Category B

6. CONCLUSION

Image files can be carved even if their headers are missing. The ULFS tool is very much useful in constructing test data sets in the form virtual disks. The research area of file carving is important to counter the computer based criminal activities. In the future work we would like to extend our work to carve the image files when *all* headers are missing i.e. images of category C and category D. Also we would like to extend our work to carve image files when some of the data clusters are missing.

REFERENCES

1. Nasir Memon, Anandabrata Pal, "Automated Reassembly of File Fragmented Images Using GreedyAlgorithms",IEEE Transactions onImage Processing, Volume 15,No.2, February,2006
2. Yanbin Tang, Junbin Fang, K.P. Chow, S. M. Yiu, Jun Xu, Bo Feng, Qiong Li, Qi Han: Recovery of heavily fragmented JPEG files, **ELSEVIER,** Digital Investigation, 2016.
3. K. Srinivas, T. Venugopal, "Automated Generation of a Natural Challenge File for File Carving Algorithms",International Conference on Applied Sciences, Engineering, Technology and Management-2017 at DRK Institute of Science and Technology, Hyderabad, Telangana, India.
4. K. Srinivas, T. Venugopal,(In press) "Testing a File Carving Tool Using Realistic Datasets Generated with Openness", International Journal of Data Analysis Techniques and Strategies.
5. Andreas Dewald, Sabine Seufert, "AFEIC: Advanced forensic Ext4 inode carving",DFRWS 2017 Europe – Proceedings of the 4th Annual DFRWS Europe – Elsevier Journal - Digital Investigation 20 (2017) S83-S91
6. Anadabrata Pal, Nasir Memon, "The Evolution of File Carving: The benefits and problems of forensics recovery",IEEE Signal Processing magazine Vol. 26. No 2. March 2009.
7. Husrev T: Sencar, Nasir Memon: Identification and recovery of JPEG files with missing fragments, **ELSEVIER,** Digital Investigation, 2009.
8. Kulesh Shanmugasundaram, Nasir Memon: *Automatic Reassembly of Document Fragments via Context Based Statistical Models*, ACSAC '03 Proceedings of the 19th Annual Computer Security Applications Conference, **IEEE** Computer Society Washington, DC, USA.
9. https://www.dfrws.org
10. Image Carving with Missing Headers and Missing Fragments IEEE 2017 Emre Durmus_, Manoranjan Mohantyy, Samet Taspinary, Erkam Uzunz and Nasir Memon
11. Siti Salasiah Mokri, M Iqbal Saripan, Abdul Jalil Nordin, Mohammad Hamiruce, Noraishikin Zulkarnain "Level Set Based Whole Heart Segmentation in Non-Contrast Enhanced CT Images", International Journal of Advanced Trends in Computer Science and Engineering, Volume 8, No. 1.6, 2019.
12. Rafidah Muhamad1, Azurah A. Samah2, Hairudin Abdul Majid3, Zuraini Ali Shah4, Haslina Hashim5, Nik Azmi Nik Mahmood6, Dewi Nasien7, M. Hasmil Adiya8 "Block-based Approaches for Copy-move Image Forgery Detection :A Review", International Journal of Advanced Trends in Computer Science and Engineering, Volume 8, No. 1.6, 2019.