ISSN No. 0976-5697
Volume 4, No. 11, Nov-Dec 2013
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Enhanced Gap Sequencing Shell Sort

AgamVerma[1], Jitender Kumar Sharma
M.Tech(CSE)
United College of Engg.and Research
Allahabad, India
agam_alpine@yahoo.com[1], jitenderkumkum@yahoo.com[2]

Abstract:Sorting algorithms are used for arranging alist of numbers or characters in an ascendingor descending order.Many sorting algorithmshave been proposed for sorting a given sequence,some important algorithms of themare Bubble Sort, Insertion Sort,SelectionSort,Shell sort and many more. Sorting algorithms puts elements in a certain order suchas numerical or lexicographical order.Shellsort is the fastest algorithm in comparison tobubble,insertion and selection.Shell sort is anenhanced version of insertion sort. It reducesthe number of swaps of the elements beingsorted to minimize the complexity and timeas compared to insertion sort.Shell sorting algorithm sorts the elements according thegap sequences.The operations depends on theused gap sequences(many gap sequences havebeen proposed).In this paper we analyze thealgorithm by using the following gap sequence:$N_1;N_2;N_3;\ldots; 1$,where

N_1 = floor (3n/4);
N_2 = floor ($3N_1/4$);
N_3 = floor ($3N_2/4$);
………………….;
………………….;
N_k = floor ($3N_{k-1}/4$) and so on until the value of N_k becomesone. Here ‘n’ is the number of elements to besorted.

Keywords:Algorithm,Shell,Sorting,Comparison.

I. INTRODUCTION

Shell sort is the algorithm that has beenthe focus of research.Shell sort is introducedby Donald L. Shell in 1959[1].Shell sortingalgorithm is the generalization of insertionsort algorithm[2].It is easy to implement,andthus a practical choice for sorting moderate-sized lists.Shell sort has been proposed toimprove the average running time of theinsertion sort, which is Θ (n^2)[2].It has beenobserved that Shell sort is a non-stable in-placesort. Shell sort improves on the efficiency ofinsertion sort by quickly shifting values totheir destination [2]. Many different gap sequences may be used to implement the shellsort[1].Typically, the array is sorted with largegap sequencing,then the gap sequencing isreduced,and the array is sorted again. On the final sort,gap sequencing is one.

In this paper work, I use the tools developed in some previous published papers on shell sort to analyze the gapsequence:$N_1;N_2;N_3;$ …………………. 1,where

N_1 = floor (3n/4);
N_2 = floor ($3N_1/4$);
N_3 = floor ($3N_2/4$);
…………………...;

N_k = floor ($3N_{k-1}/4$) and so on until the value of N_k becomesone. Here ‘n’ is the number of elements to besorted.In analyzing Enhanced Gap SequencingShell Sort,It turns out that the value of the gapsequence depends on the number of elements to be sorted. First,we find N_1 = floor (3n/4),after thisall other gaps $N_2;N_3;$ ……………….. ; can be found byputting N_1in place of ‘n’ to get N_2 and N_2 in place of ‘n’ to get N_3 and so on until we get the value 1.For example, if we want to sort 125 numbers of elements,then N_1 = floor (3(125)/4) becausehere n=125,so N_1=93. Now find N_2 = floor ($3N_1/4$) which is 69,and N_3=51,N_4=38,N_5=28,N_6=21,N_7=15,N_8=11,N_9=8,N_{10}=6,N_{11}=4,N_{12}=3,N_{13}=2,N_{14}=1. So the gap sequence for sorting 125 elements is93,69,51,38,28,21,15,11,8,6,4,3,2,1.

In this paper,I made the list of comparisonsof “Enhanced Gap Sequencing Shell sort” with Insertion sort and Shell sort. In Shellsort the numbers of swaps are reduced ascompared to Insertion sort and in “Enhanced Gap Sequencing Shell Sort” the numbers ofswaps are further reduced as compared to Shellsort.[3]

A. Insertion Sort:

Insertion Sort algorithm sorts the elementsby inserting them into their proper position in the final sorted list.Insertion sort keeps making the left side of the array sorted until the wholearray is sorted.It sorts the values seen faraway and repeatedly inserts unseen values inthe array into the left sorted array.It is thesimplest of all sorting algorithms.Although ithas the same complexity as Bubble Sort (Θ (n^2)),the insertion sort is a little over twice as efficient as the bubble sort.The advantage ofInsertion Sort is that it is relatively simple andeasy to implement.

In the following example,the number of swapsis calculated to sort the elements using Insertion Sort algorithm.In this example,the listcontains 38 elements,which are unsorted,theInsertion sort is applied in order to find thetotal number of swaps required to sort theelements in the increasing order.

a. Unsorted list:

22,12,53,94,27,59,50,39,14,88,35,3,115,3,4,230,29,84,62,102,14,54,5,3,87,67,16,43,73,19,27,64,16,4,2,85,51,34.

After applying the Insertion sort on this array,the number of swaps calculated for sortingit, are **367**.

B. Shell Sort:

Shell sort sorting algorithm is introduced bythe Donald L. shell in 1959[1,2].Shell sort worksby comparing elements that are distant ratherthan adjacent elements in an array or list whereadjacent elements are compared.Shellsort usesa gap sequence g_1; g_2; …….; $g_t$called the increment sequence. Any increment sequence is fine as long as $g_1 = 1$ and some other choices arebetter than others.Shellsort makes multiplephases through a list and sorts a number ofequally sized sub lists using the insertion sort.Shellsort is also known as diminishing increment sort[2].

The distance between comparisonsdecreases as the sorting algorithm runs untilthe last phase in which adjacent elementsare compared.After each phase and someincrement g_h,for every i,we have $a[i] \leq a[i+g_h]$all elements spaced $g_h$apart are sorted.The file is said to be g_h -sorted .The size of thesub-lists,which are to be sorted gets largerwith each phase through the list, until thesub-list consists of the entire list. (Note thatas the size of the sub list increases, the number of sub-lists to be sorted decreases.)Thisarrangement makes the insertion sort to runfor an almost-best case with a complexity thatapproaches O(n).

The elements contained in each sub-list are notcontiguous, rather,if there are i sub-lists then asub-list is composed of every i-th element.Forexample,if there are 4 sub lists then the firstsub-list would contain the elements located atpositions 1,5,9 and so on.The second sub-listwould contain the elements located at positions 2,6,10,and so on;while the fourth sub-listwould contain the items located at positions4,8,12,and so on.[3]

The efficiency of the algorithm is depends onthe size of the sub-lists used for each iteration.Along with the benefit of being robust,Shellsort is a complex algorithm and not nearlyas efficient as the merge, heap, and quicksorts.The shell sort is still significantly slowerthan the merge, heap, and quick sorts, butits relatively simple algorithm makes it agood choice for sorting lists of less than 5000items unless speed important. It is also anexcellent choice for repetitive sorting of smallerlists.It has been observed that Shell sort is anon-stable in-place sort. Shell sort improves the efficiency of insertion sort by quicklyshifting values to their appropriate position.Average sort time is $O(n^{1.25})$, while worst-casetime is$O(n^{1.5})$[3].

Knuth has experimented with several valuesand recommends that gaping 'h' for an array ofsize N be based on the following formula: Let$h_1= 1$; $h_{s+1}= 3h_s+ 1$,and stop with $h_t$when$h_{t+2} \geq N$[3].Thus,values of h are computed as follows:

$h_1= 1$
h2 = (3*1) + 1 = 4
h3 = (3*4) + 1 = 13
$h_4= (3*13) + 1 = 40$
$h_5= (3*40) + 1 = 121$
$h_6= (3*121) + 1 = 364$

To sort 125 items we first find an 'h_s' suchthat $h_s \geq 125$.For 125 items, $h_6$is selected. The Final value $h_t$is two steps lower, or h_4.Therefore sequence for the values of 'h' willbe 40,13,4, 1.Once the initial 'h' value hasbeen determined, subsequent values may be calculated using the formula

$h_{s-1}= floor(h_s/3)$.

Now we apply the shell sort for calculating thenumber of swaps to sort the same problem (asdiscussed in Insertion sort) by using Knuth'sgap sequences.

a. Unsorted list:

22,12,53,94,27,59,50,39,14,88,35,3,115,3,4,230,29,84,62,102,14,54,5,3,87,67,16,43,73,19,27,64,16,4,2,85,51,34.

After applying the shell sort using Knuth's gapsequences on this array of numbers,the numberof swaps calculated for sorting it, are **170**.

C. Enhanced Gap SequencingShell Sort:

Enhanced Gap Sequencing Shell Sort introduced a new way to find the gap sequences tosort the large list of elements rapidly. Enhanced Gap Sequencing Shell Sort algorithm alsoworks in same fashion as the previous existing versions of the Shell sort algorithm.The only difference is the way to choose the more efficient gap sequence,which is a key step forthe algorithm to be more effective and better.Calculating the value of the gap sequence$h_s$in conventional Shell sort is a key step inexecution of the algorithm.

In conventional Shell sort, given by Knuth, the value of $h_s$is found by the following formula:

$h_1= 1$; $h_{s+1}= 3h_s+ 1$,and stop with $h_t$when$h_{t+2} \geq N$.

By using this existing formula Shell sortreduces the number of swaps up-to 50% ascompared to that of Insertion sort.

Enhanced Gap Sequencing Shell sort mainlyfocuses to improve the efficiency of the previous existing shell sort algorithms.It can beachieved by choosing the appropriate values ofgap sequences,which can reduce the numberof swaps.In conventional Shell sort,the gapsequences are small,so they divide the list intolarge number of sub-steps,for example,if n=100 then the gap sequence is (13,4, 1),which meansto sort the elements, it first divide the list into8-sublists of size 13,then 25-sublists of size 4.Itmakes algorithm to swap many elements manytimes to place them into right place.So, if we increase the size of gap sequence,it divides thelist into less number of sub-lists of larger sizeand elements are not needed to swap manynumbers of times.

Enhanced Gap Sequencing Shell Sort introduces a new mechanism to calculate the valueof gap sequence.The formula for calculatinggap sequence is as follows:

The gap sequence is of the form N_1,N_2, N_3,…….,1,whereN_1= floor (3n/4);
N_2= floor ($3N_1/4$);
N_3= floor ($3N_2/4$);
…………………..;
…………………..;

N_k= floor ($3N_{k-1}/4$) andso on until the value of $N_k$becomes one.Here 'n' is the number of elements to be sorted.Thusto sort 125 numbers of elements,the gap sequence is calculated as follows:here n=125

N_1= floor (3*125/4) = 93
N_2= floor (3*93/4) = 69
N_3= floor (3*69/4) = 51
N_4= floor (3*51/4) = 38
N_5= floor (3*38/4) = 28
N_6= floor (3*28/4) = 21
N_7= floor (3*21/4) = 15
N_8= floor (3*15/4) = 11
N_9= floor (3*11/4) = 8
N_{10}= floor (3*8/4) = 6

N_{11} = floor (3*6/4) = 4
N_{12}= floor (3*4/4) = 3
N_{13}= floor (3*3/4) = 2
N_{14}= floor (3*2/4) = 1

So the gap sequence for sorting 125 elementsis (93,69,51,38,28,21,15,11,8,6,4,3,2,1).But inconventional Shell sort the gap sequence for 125 elements is (40, 13, 4, 1).

Now we apply the Enhanced Gap Sequencing shell sort for calculating the number of swapsto sort the same problem as discussed in Insertion sort and Shell Sort).

a. Unsorted list:

22,12,53,94,27,59,50,39,14,88,35,3,115,3,4,230,29,84,62,102,14,54,5,3,87,67,16,43,73,19,27,64,16,4,2,85,51,34.
Here n=38,so
N_1= floor (3*38/4) = 28
N_2= floor (3*28/4) = 21
N_3= floor (3*21/4) = 15
N_4= floor (3*15/4) = 11
N_5= floor (3*11/4) = 8
N_6= floor (3*8/4) = 6
N_7= floor (3*6/4) = 4
N_8= floor (3*4/4) = 3
N_9= floor (3*3/4) = 2
N_{10}= floor (3*2/4) = 1

So the gap sequence for 38 elements is (28,21,15,11,8,6,4,3,2,1). After applyingthe Enhanced Gap Sequencing shell sort onthis array of numbers,the number of swapscalculated for sorting it, are **67**.

II. COMPARISON OF ABOVE DISCUSSED THREE TECHNIQUES

Now the comparison for the three techniques is made here for the same problem.

Table 1: Comparison

Sr. No.	*Insertion Sort*	*Shell Sort*	*E.G.S. Shell Sort*
01	367	170	67

It is apparent that Shell sort reduces thenumber of swaps up to half as compared tothe number of swaps in Insertion sort and Enhanced Gap Sequencing Shell Sort reduces thenumber of swaps further up to less than half as compared to the number of swaps in Shell Sort, thus improving the efficiency of the algorithm.

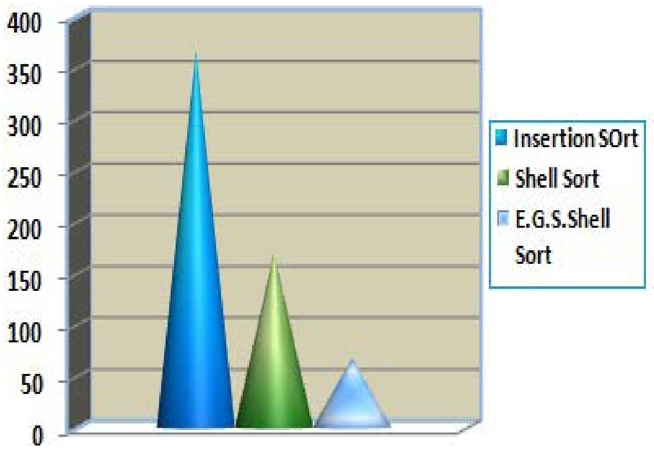

Figure 1: Comparison of Sorting Techniques

III. DETAIL DISCUSSION OF RESULTS FOR MORE PROBLEMS

It is necessary to execute a detailed comparison of all three algorithms by applying the algorithms on a wider variety of data,in order toensure and establish results concretely.

A. For 25 Elements:

0,1,13,4,14,10,19,11,22,27,14,17,13,20,19,15,16,19,18,19,18,17,21,20,14.
Number of swaps in Insertion Sort:83
Number of swaps in Shell Sort:47
Number of swaps in E.G.S.Shell Sort:22

B. For 50 Elements:

0,1,21,4,23,16,32,16,40,50,21,28,16,38,33,16,20,39,35,37,38,40,25,29,37,35,35,33,30,31,31,31,31,31,34,32,35,29,32,32,30,35,23,37,40,34,26,42,30,26.
Number of swaps in Insertion Sort:464
Number of swaps in Shell Sort:127
Number of swaps in E.G.S.Shell Sort:63

C. For 100 Elements:

1,1,38,6,40,27,59,26,75,98,35,50,21,73,61,18,29,80,70,75,79,88,36,51,84,77,81,74,53,61,64,60,58,66,39,52,40,62,55,55,58,51,66,49,48,54,59,49,56,58,57,51,58,54,55,55,56,56,56,58,59,53,59,59,58,61,58,66,57,44,70,59,70,71,65,38,62,60,70,70,70,69,43,35,34,43,59,55,86,81,59,58,50,75,86,41,45,25,25,96.
Number of swaps in Insertion Sort:2154
Number of swaps in Shell Sort:419
Number of swaps in E.G.S.Shell Sort:179

D. For 200 Elements:

2,1,71,9,76,48,113,45,145,193,62,95,32,142,117,22,45,162,138,152,162,184,58,93,180,161,173,155,98,122,130,120,113,138,50,92,53,129,102,104,114,85,153,76,66,96,129,66,110,122,119,51,137,81,101,94,93,113,115,85,75,133,87,91,98,84,99,71,105,141,70,99,76,77,90,135,97,100,89,89,91,93,118,123,123,150,104,107,89,93,104,105,108,99,96,110,108,112,112,99,106,106,106,107,105,105,106,107,106,109,106,111,101,111,102,104,102,101,105,105,116,112,95,95,102,89,87,86,96,106,116,96,118,99,107,125,103,114,89,114,96,133,132,119,75,94,95,89,93,93,87,62,114,101,118,59,155,61,111,134,101,107,92,83,97,72,115,167,65,79,44,112,126,48,87,98,79,146,134,48,170,38,89,171,52,71,79,38,108,142,76,139,131,62,22,32,152,123,30,121.
Number of swaps in Insertion Sort:10180
Number of swaps in Shell Sort:886
Number of swaps in E.G.S.Shell Sort:544

E. For 350 Elements:

3,2,121,14,129,81,193,75,250,335,103,162,48,247,202,28,70,284,241,266,286,328,91,157,323,286,312,276,166,212,229,210,195,246,67,153,71,230,173,178,199,137,282,115,93,160,233,90,191,219,213,51,255,121,171,152,150,201,206,127,101,254,132,141,159,120,161,79,179,289,70,161,87,88,128,281,150,161,118,119,122,130,231,257,259,226,173,186,94,113,173,178,198,137,113,215,206,245,243,106,189,182,200,223,143,137,141,156,190,128,197,115,242,141,222,198,208,210,187,189,140,160,224,218,194,231,235,233,206,182,160,202,160,194,180,151,186,170,204,171,193,151,153,168,212,192,191,195,191,191,194,210,176,184,174,206,156,202,178,169,182,180,185,188,183,190,178,167,189,186, 191,180,178,188,183,181,182,178,179,182,180,181,181,183,179,180,180,176,182,185,178,185,185,175,169,170,189,184,167,184,188,166,196,197,190,198,175,158,194,204,154,197,199,208,180,156,153,198,173,152,169,125,180,154,152,161,171,177,137,135,224,195,158,223,162,199,224,195,210,181,238,122,242,233,213,176,192,218,187,230,222,193,251,187,188,229,239,194,111,250,200,258,154,170,252,251,236,232,234,223,208,267,137,245,112,272,233,113,102,137,126,107,160,223,85,163,150,166,147,222,162,246,73,279,256,173,99,270,209,255,79,184,205,281,145,219,243,68,276,159,243,106,168,189,276,147,84,141,70,53,214,196,256,315,262,126,134,165,50,37,276,325,152,103,168,318,234,61,38,323,77,237,214,152,68,179,250,88,45,57.

Number of swaps in Insertion Sort:29068
Number of swaps in Shell Sort:2232
Number of swaps in E.G.S.Shell Sort:1079

F. For 500 Elements:

5,3,172,19,183,114,274,104,355,478,145,228,65,352,287,35,95,406,344,381,410,471,124,221,465,412,450,398,233,303,329,300,278,354,84,214,89,331,245,252,283,189,413,155,120,223,337,115,217,316,306,52,373,162,240,210,206,288,296,169,126,375,176,191,220,156,223,87,253,436,70,222,97,100,166,427,203,221,148,154,168,345,390,395,336,242,266,99,132,242,250,287,175,131,320,303,378,374,112,272,259,293,338,181,170,176,206,275,148,289,120,385,173,343,292,316,320,270,275,165,210,355,342,286,374,383,380,316,258,204,310,202,290,254,177,270,228,320,229,290,170,175,219,349,290,288,301,289,290,302,360,239,268,230,353,158,343,246,205,264,254,279,294,271,308,243,167,315,293,339,248,231,328,279,266,287,212,226,317,190,324,272,194,306,288,280,314,255,227,279,231,237,288,315,307,225,245,303,246,237,298,217,218,237,220,268,300,233,218,300,233,231,220,258,286,288,237,264,285,267,226,257,277,278,270,262,258,282,282,233,249,267,237,263,249,240,251,247,256,240,270,242,245,249,256,254,250,255,251,252,255,252,255,255,255,256,257,255,259,257,261,254,256,263,264,263,263,264,263,261,271,249,268,243,275,267,241,238,246,243,238,251,267,230,251,248,252,246,269,251,277,221,289,282,254,228,288,267,283,219,257,265,295,242,271,281,210,295,247,282,225,251,260,298,241,213,238,206,198,271,263,291,319,295,230,234,249,192,185,303,328,242,217,249,326,284,194,182,331,202,286,274,241,195,255,294,205,181,187,220,332,238,275,199,190,277,254,174,216,228,355,226,210,283,189,242,353,187,234,155,192,348,211,160,354,337,358,212,280,318,255,351,251,382,312,306,327,188,349,309,174,225,366,126,239,298,281,137,390,255,151,301,148,211,303,399,398,289,161,379,279,123,149,103,247,175,187,101,393,272,325,116,251,93,225,245,382,248,301,184,198,347,232,276,315,413,89,205,99,130,124,257,418,113,371,178,364,312,115,383,235,247,206,261,359,355,285,114,346,138,249,176,279,360,402,184,263,110,359,337,431,378,79,50,117,49,73,312,272,146,286,303,444,62,78,313,406,335,461,205,426,42,154,354,192,412,221,196,65.

Number of swaps in Insertion Sort:62679
Number of swaps in Shell Sort:3456
Number of swaps in E.G.S.Shell Sort:1613

G. For 1000 Elements:

10,5,339,36,361,222,542,202,705,953,282,450,119,701,569,55,177,814,687,763,824,951,234,435,942,832,912,803,459,605,659,600,552,714,141,417,150,666,482,497,565,361,848,287,210,435,685,197,541,638,618,53,766,296,470,403,395,578,598,309,212,777,323,358,425,277,430,114,500,928,71,427,132,137,293,915,381,424,246,247,260,292,724,835,848,704,473,532,117,197,471,492,586,302,188,671,628,821,812,133,549,514,603,724,307,279,294,374,558,214,596,135,859,280,747,606,673,686,546,561,249,376,789,754,593,851,879,871,685,513,352,669,345,610,500,264,549,420,708,424,616,233,247,387,809,617,612,656,617,621,663,859,450,549,419,845,166,814,472,323,538,502,592,648,562,703,460,169,732,651,833,477,407,798,601,548,637,324,381,766,227,803,579,235,729,651,617,777,502,370,620,386,414,665,803,767,347,450,760,456,407,741,290,291,398,296,580,774,368,268,789,355,339,263,522,725,740,370,572,732,594,268,515,688,702,639,569,532,769,773,265,432,631,286,604,418,298,440,373,508,250,768,242,286,373,528,465,361,485,324,354,463,256,487,483,346,314,466,733,288,448,270,588,540,301,306,351,364,361,395,436,283,620,344,679,284,380,669,693,608,633,672,554,416,714,545,571,538,576,423,543,380,715,320,366,521,655,345,456,377,683,501,467,340,565,445,408,686,359,541,412,618,525,494,369,555,643,562,658,679,462,487,408,334,403,575,565,526,665,678,394,338,539,595,521,354,448,636,659,356,614,449,473,535,618,508,439,595,635,622,565,383,534,477,595,608,474,508,625,563,545,369,547,567,470,592,525,385,591,533,627,581,400,557,613,398,418,397,552,481,443,507,413,511,386,453,459,441,567,425,460,575,532,415,611,519,473,486,596,406,506,582,474,581,537,474,410,412,485,566,429,492,586,569,595,511,551,544,590,433,497,470,577,508,585,521,511,447,509,486,537,530,467,516,498,482,445,569,525,563,551,552,505,451,553,468,530,472,489,547,469,511,508,519,504,479,481,499,539,485,532,507,523,501,485,477,519,504,532,488,492,477,532,536,525,534,530,498,504,518,502,501,486,524,522,501,493,499,491,509,495,518,511,501,508,499,507,508,511,504,510,503,506,506,505,506,507,505,507,505,502,509,508,506,499,503,512,506,507,502,519,498,493,503,516,526,515,528,510,530,484,518,516,499,511,517,537,507,541,528,541,493,526,483,507,542,526,469,519,499,535,465,462,482,468,528,537,523,514,466,538,449,463,448,455,558,451,542,453,551,544,464,438,559,498,467,482,540,580,488,445,488,545,484,493,505,468,455,586,438,516,575,469,528,474,507,449,528,504,575,472,517,504,547,440,575,533,525,514,469,417,460,564,579,564,604,556,515,529,583,594,394,516,620,483,570,577,467,584,571,506,380,570,449,538,632,558,434,627,641,440,429,403,402,514,596,574,459,480,377,385,631,450,579,631,362,395,505,543,365,644,408,418,407,405,578,553,664,345,571,513,673,462,344,405,538,511,677,345,661,566,616,572,366,453,393,514,510,401,684,650,494,620,452,409,394,381,576,694,324,617,483,678,698,493,479,448,591,323,547,502,704,459,564,424,457,403,669,585,609,330,486,559,642,660,419,567,603,717,305,300,643,681,409,383,358,673,388,558,669,605,496,453,513,355,502,671,299,512,716,354,690,726,390,670,561,628,589,721,353,527,679,351,390,456,583,742,600,543,429,747,624,372,270,505,490,448,455,307,706,683,246,394,732,586,706,493,653,430,601,277,220,555,277,673,690,602,514,252,444,418,648,406,440,611,431,236,567,239,515,216,668,261,504,652,620,699,658,513,498,714,272,598,378,513,211,256,251,307,498,215,242,617,489,627,346,300,197,678,510,820,194,368,263,509,311,638,779,686,544,740,415,468,492,497,235,470,803,474,283,681,692,747,429,654,674,793,194,310,332,710,606,573,356,662,659,420,635,756,187,540,136,382,372,704,674,149,366,314,199,164,455,340,202,748,692,401,723,767,838,463,788,711,327,223,671,905,510,363,871,337,585,854,850,798,597,531,467,225,604,490,129,202,566,92,859,794,443,313,901,354,280,681,262,768,559,558, 738,811,885,172,699,654,661,402,228,246,806,714,844,247,568,440,338,844,657,341,193,293,597,750,731,202,780,129,359,502,476,485,118,921,726,774,272,65,391,142,799,200,938,571,782,979,576,115,546,430,147,836,394,886,256,905,672,152,730,913.

Number of swaps in Insertion Sort:243764

Number of swaps in Shell Sort:8744

Number of swaps in E.G.S.Shell Sort:3934

Table 2: Comparison of all three sorts

Cases	*Insertion Sort*	*Shell Sort*	*E. G. S. Shell Sort*
3.1	83	47	22
3.2	464	127	63
3.3	2154	419	179
3.4	10180	886	544
3.5	29068	2232	1079
3.6	62679	3456	1613
3.7	243764	8744	3934

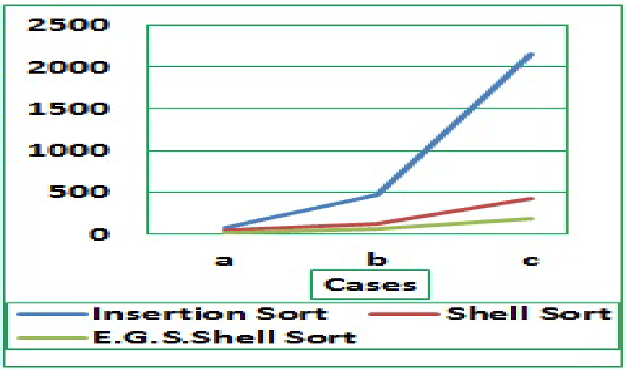

Figure 2: Comparison graph based on cases

The Enhanced Gap Sequencing Shell Sortingalgorithm is a good approach towards achievingthe excellence in the algorithms to provide the more efficient solutions. This has been achievedby decreasing the number of swaps required tosort the list of elements.

The results of above solved problems showthat the new algorithm provides a much efficient way to sort the elements and hence causesto save the computational resources. It hasbeen observed that the Enhanced Gap Sequencing Shell Sorting algorithm can solve the problem in almost 60 times less swaps as comparedto insertion sort and in almost half swaps ascompared to the traditional shell sorting algorithm. Figure 3 shows the detailed overview ofthe number of swaps required to sort differentnumber of elements.

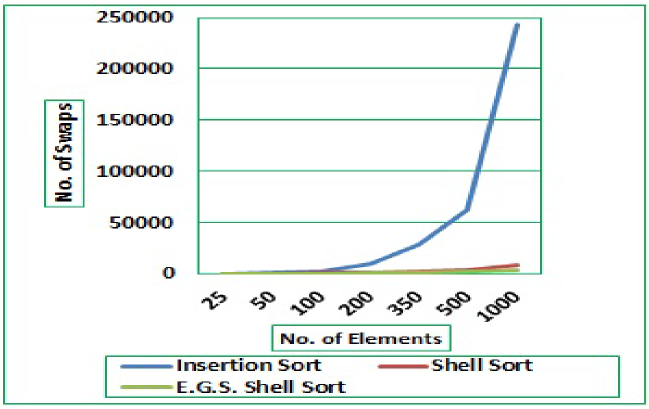

Figure 3: Graph showing elements-swap ratio

IV. ANALYSIS

In the paper "Analysis of Shellsort and Related Algorithms",Robert Sedgewick[4]has described an open problem, "Are there increment sequences that perform better than known ones in practice?", for performance issues and claims thatfinding a sequence that leads to running times 25% lower than the best known certainly would be of practical interest, we can reduce the running time by reducing the number of comparisons for the algorithm. In our proposed algorithm, we have reduced the number of comparison up-to 40% to 50% in some ideal cases bur in many cases up-to 20% to 30%. We compare our algorithm with insertion sort ant shell sort and get some interesting results as can be seen in the above given graphs and tables. MarcinCiura [5] in his paper "Best Increments for the Average Case of Shell sort", shows the result for 128 elements where data get sorted in 535(approx.) swaps but in our case 200 elements takes 544 swaps to get sorted. Thus the proposed algorithm is better for sorting.

V. CONCLUSION

This research paper focuses on an enhancement and improvement in existing sorting algorithms.The traditional Shell sort algorithmresults an average number of comparisons ofelements but it does not give minimum numberof swaps. The older approaches of the ShellSort algorithm have stated that the number ofswaps produced by Shell Sort can be further reduced by choosing an efficient gap sequence.

The main purpose of reducing the number ofswaps is to use the computational resourcesthat are available in terms of processor speed,memory and storage.

Enhanced Gap Sequencing shell sort algorithm provides an efficient and better approach todecrease the number of comparisons as wellas number of swaps. Enhanced Gap Sequencing shell sort algorithm results least number ofswaps on any size of data. This algorithm worksmore efficiently as the size of data grows.

This algorithm has described a simple and easyformula that calculates the values of N_1bythe formula N_1= floor (3n/4), where 'n' is the number of elements to be sorted, and then findvalues of N_2;N_3; ……….. ; $N_k$by placing valuesof N_1;N_2;N_3; ……….; N_{k-1}in place of `n' in theformula. This algorithm improves the performance of the existing algorithms up to 60% insome cases.

VI. ACKNOWLEDGMENT

Anumber of people have made this paper possible. In particular I wish to thankmy friend, Mr. Jitender Kumar, for providingmuch needed help and support and my teachersfor their guidance and displaying inconceivablepatience. My family and friends for constantly asking me when I would finish my thesis.And lastly,my mother for teaching me never to giveup on any endeavor.

VII. REFERENCES

[1]. http://en.wikipedia.org/wiki/Shellsort

[2]. http://faculty.simpson.edu/lydia.sinapova/www/cmsc250/LN250 Weiss/L12-ShellSort.htm

[3]. BasitShahzad, and Muhammad Tanvir Afzal, “Enhanced Shell Sorting Algorithm”, World Academy of Science, Engineering and Technology 3 2007.

[4]. Robert Sedgewick “Analysis of Shell sort and Related Algorithms” Proceedings of the Fourth Annual European Symposiumon Algorithm.

[5]. MarcinCiura “Best Increments for theAverage Case of Shell sort”,Proceedingsof the 13th International SymposiumonFundamentals of Computation Theory,2001, pp: 106117