Project

Introduction:

Write-Once-Memory (WOM) codes were first introduced in 1982 by Rivest and Shamir. These codes make it possible to record information more than once on media which only allows the irreversible change of 0 bits to 1.

A simple example presented by Rivest and Shamir is in the following table:

Data bits	First Write	Second Write
00	000	111
10	100	011
01	010	101
11	001	110

This code permits writing of two bits of information in three 1-bit cells twice. The code usage is as follows: On the first write, choose for the data the appropriate code word in the first column. On the second write, if the data is different than the data written on the first write, choose the appropriate code word in the second column. Otherwise, do nothing.

First it should be noted, that this scheme doesn't violate the restriction of not changing 1 to 0. Second, it is easy to see that the decoding is straightforward and each code word corresponds to exactly one original data word.

This is only the basic idea which was later elaborated by many researchers to include other numbers of repetitions on different numbers of cells.

Goal:

Show that by introduction of WOM codes to the Flash Transmission Layer (FTL), the number of page erases performed by the physical memory, a key factor in the longevity of flash memories, can be reduced significantly.

Outline:

A software was designed to simulate the structure and operation of a generic memory – the logical layer, the FTL and the flash memory (the physical layer).

Definitions and notations:

LPN – the total logical page number in logical memory.

PPN – the total physical page number in flash memory.

PBN – the total physical block number in flash memory.

PPB – number of pages per physical block.

α – the ratio of logical pages to physical pages, i.e. $\frac{\text{LPN}}{\text{PPN}}$.

β – the number of pages to which a logical page will be mapped on a second write.

Flash memory

1. The Flash memory can be written in units of a page, or half a page.
2. Page rewriting is limited to changing zeroes to ones (The introduction of WOM codes utilizes this feature).
3. Full rewrite is possible only after deletion of a full block.

FTL:

The data structure:

Consists of a mapping table and a block table.

The mapping table: LPN entries, an entry for each logical page. Each entry holds: Status: status of the logical page, which can be free or used.

Physical pages: In case the page is used, the table holds the numbers of the physical pages the logical page was mapped to. Each logical page can be mapped to more than one physical page as described below.

The block table: PBN entries, an entry for each physical block. Each entry holds:

Write Phase: first write or second write as described below.

Erase count: the number of times the block was erased.

Valid1: the number of valid physical pages in the block written as 1:1, i.e. one logical page mapped to one physical page.

ValidBeta: the number of valid physical pages in the block written as 1:β, i.e., pages written to the block on second write with β physical pages corresponding to one logical page.

Page table: PPB entries, one for each physical page in the block. Each entry holds:

> Status: VALID - if physical page is used and a logical page is mapped to it. OBSOLETE – if physical page is used but a logical page is no longer mapped to it. FREE – free physical page.
>
> LogicalPage: in case the physical page is VALID, the corresponding logical page.
>
> nthPhysical: in case the corresponding logical page is VALID and mapped to more than one physical page, a value n denotes the current physical page was the nth physical page the logical page was mapped to.

The control case:

The basic sector (page) mapping algorithm was used. The algorithm writes data to the physical memory as sequentially as possible by keeping track of the next free page in physical memory.

A write command is executed as follows: the page is written to the next free physical page in memory. If no such page exists, (i.e. all pages are either VALID or OBSOLETE), FTL finds a block with the largest number of OBSOLETE pages. Then it erases the block and writes the valid data back to it sequentially from the beginning of the block. Then the obtained free pages are used sequentially.

This case will be denoted by $\beta = 1$.

Flash usage with rewrite:

Rewrite defined: clever usage of the ability to change zeroes to ones in the flash memory without erasing the block, allows limited rewrite. Introduction of WOM codes allows to rewrite a physical page but with the price that on a second write a logical page will be mapped to β pages, with $\beta > 1$.

Two alternatives were considered for a second write in a block:

1. All valid pages in the block have to be rewritten to β pages and the resulting free space can be used for writing of each logical page to β pages. This option is denoted by $\gamma = 0$.
2. Valid pages can be left intact and the obsolete pages in the block can be used for writing of each logical page to β pages. This alternative is denoted by $\gamma = 1$.

The formulas for calculation of usable pages in a block on the next write:

	$\gamma = 0$	$\gamma = 1$
End of 1st write	$\left\lfloor \frac{PPB - Valid1 \cdot \beta}{\beta} \right\rfloor$	$\left\lfloor \frac{PPB - Valid1}{\beta} \right\rfloor$
End of 2nd write	$PPB - \frac{ValidBeta}{\beta}$	$PPB - Valid1 - \frac{ValidBeta}{\beta}$

Whereas in the control case the choice of the block to be cleaned when physical memory is full, is straightforward, for $\beta > 1$ this choice is more complex.

Several algorithms for choosing the block to be cleaned upon filling of physical memory were tested for various values of α, β, γ.

The naïve algorithm: while there is a block available for 2nd write, choose among those the one with the maximum number of usable pages. If no blocks are available for 2nd write, choose from the blocks available for 1st write the one with maximal number of usable pages. This algorithm yielded poor results in the simulation, showing only minor erase savings compared with the control case.

The greedy algorithm: each time choose the block with the highest number of usable pages on the next write. This algorithm tested poorly.

The naïve with threshold algorithm: variation of the naïve algorithm. While there are blocks available for 2nd write with usable pages on the next write above a certain threshold, choose the one with the maximal usable pages. Otherwise, choose among the blocks available for 1st write, the one with maximal number of usable pages. Trial and error with the threshold value, reached the optimal erase values achieved in the testing, but for different values of α,β,γ the optimal threshold value was different without a visible pattern.

The minimum valid logical pages algorithm: choose the block with the minimum number of logical pages mapped to it. Testing revealed this as the best algorithm. For $\gamma = 0$ no algorithm gave better results. For $\gamma = 1$ this algorithm yielded results very close to the optimal testing results. The minimum valid logical pages algorithm with factor further improved the results to the optimal ones.

The minimum valid logical pages algorithm with factor: Let b_1 be the block with the minimum number of valid logical pages mapped to it among the pages at the end of 1st right and let v_1 be its number of valid logical pages. Let b_2 be the block with the minimum number of valid logical pages mapped to it among the pages at the end of 2nd right and let v_2 be its number of valid logical pages. Finally, let f be a number. Now, the algorithm is: if $v_1 \le fv_2$ choose b_1. Otherwise, choose b_2. Clearly, for $f = 1$ this is the same as the previous algorithm. It should be noted, that in the case $\gamma = 0$ different values of f were tested and the global optimal results were achieved for $f = 1$. In the case $\gamma = 1$, global optimal results were achieved for $f > 1$, as we be shown below.

The Simulation:

The following configuration was used for the simulation:

Page size: 2048 B

Pages per block: 64

Physical block number: 1024

Logical block number: α * Physical block number

Number of written pages: n.

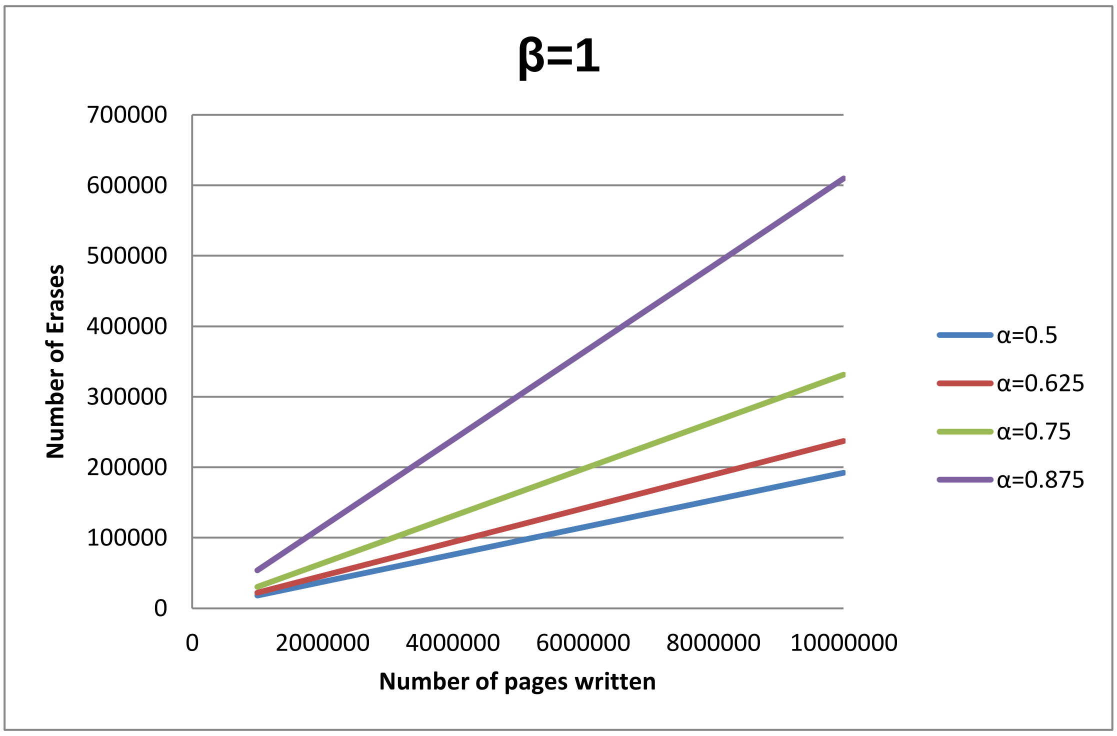
With different values for α,β,γ and n.

The simulation was ran by writing n random content pages to the logical memory and counting the total number of erases performed. Each page was written individually to a randomly chosen logical page. It should be noted, that this isn't a "real world" scenario for a disk workload, were data is mainly written in large chunks, but rather a worst case scenario and therefore the obtained results should be interpreted as upper bounds on the quantities measured.

Results and analysis:

The control case – β=1

As found in other studies, a linear correspondence between the number of erases and the number of pages written, was found. If we define α' to be the average occupancy of the block chosen for cleaning, than the formula presented by Jai Menon $\alpha = \frac{1-\alpha'}{\ln\left(\frac{1}{\alpha'}\right)}$ was confirmed by the simulation results.



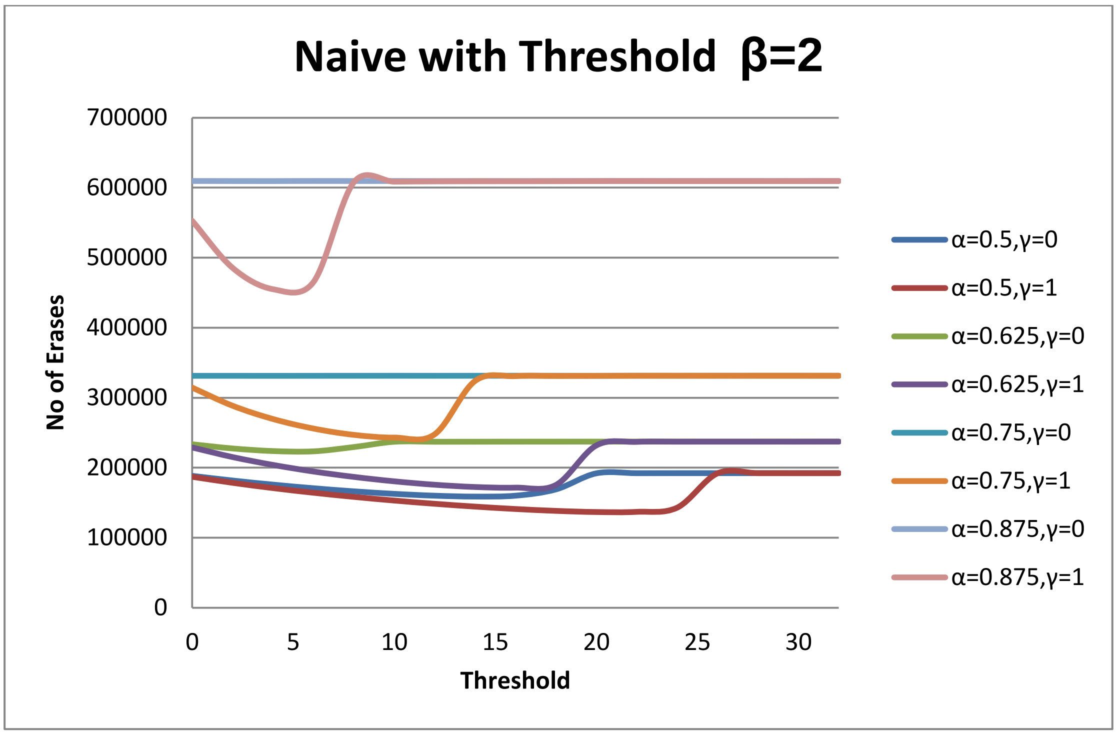
The greedy algorithm

As mentioned above this algorithm achieved only negligible improvements compared with the control case. For example, in the case α=0.5, β=2, γ=0 an average improvement of only 0.05% was achieved.

The naïve algorithm with threshold

As mentioned above, only minor improvements were achieved by the basic algorithm. Addition of a threshold and tweaking it, yielded optimal results.

Here are sample results for β=2:

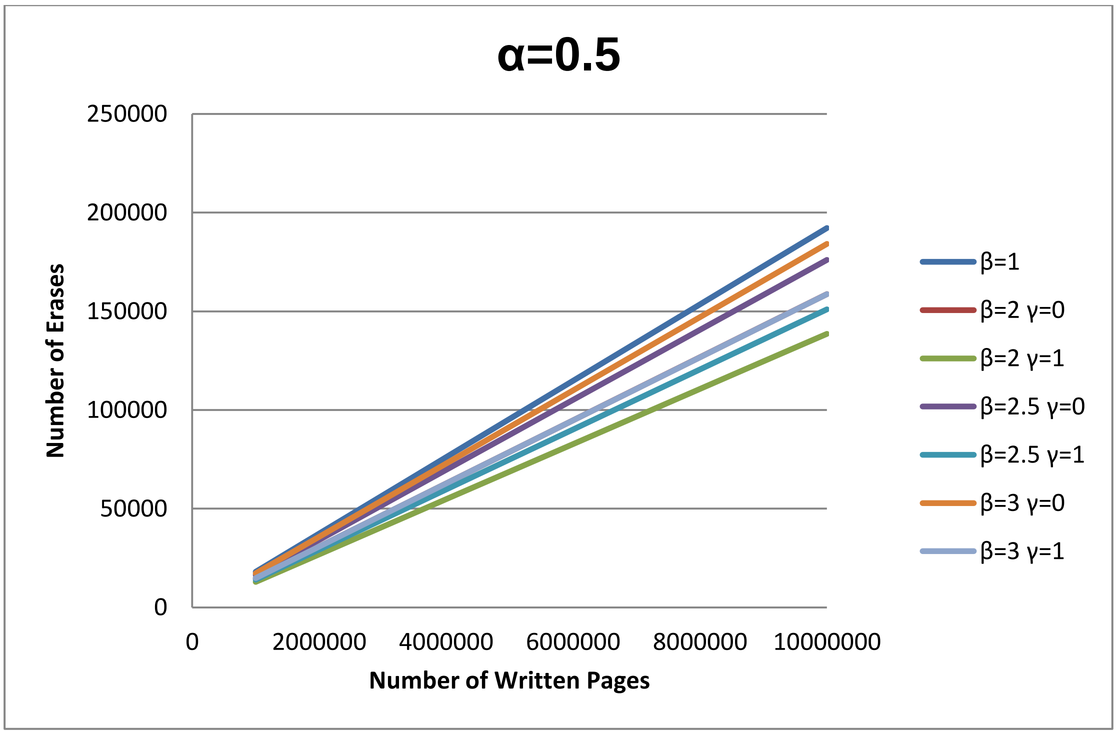


The minimum valid logical pages algorithm (with factor):

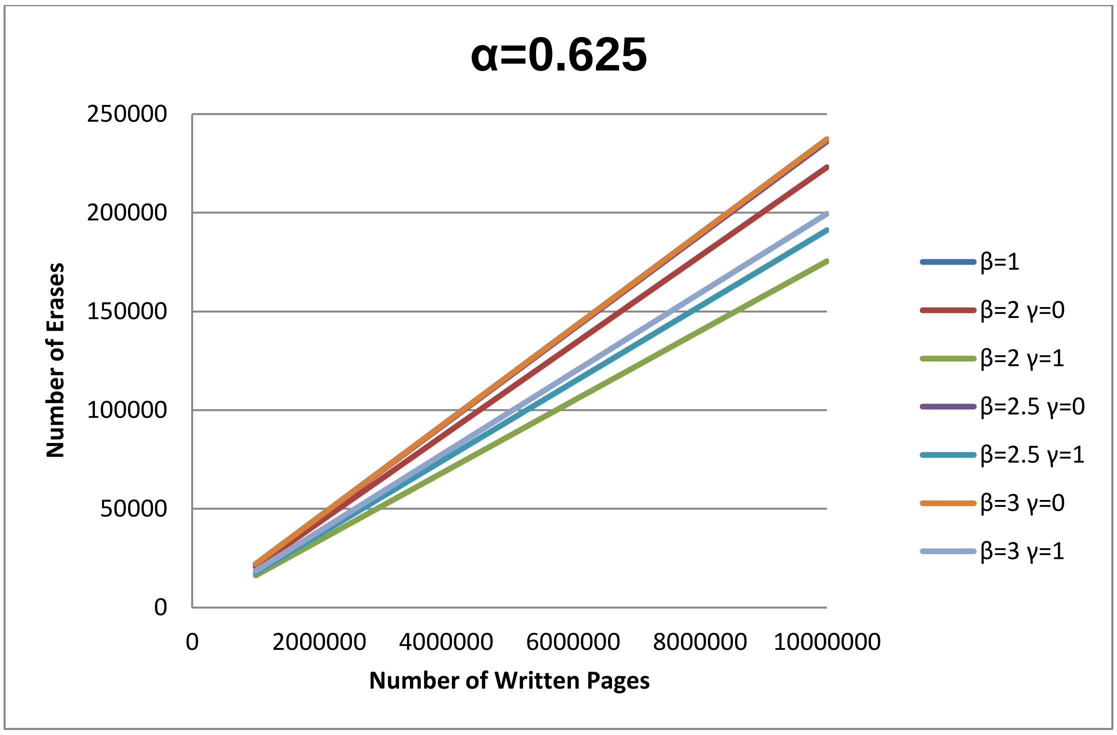
As mentioned above, the basic algorithm achieved global optimal results for γ=0. For γ=1, the results obtained by the basic algorithm were very close to the global optimal. Further tweaking with a factor, achieved the global optimum.

Here are the test results for different α's:

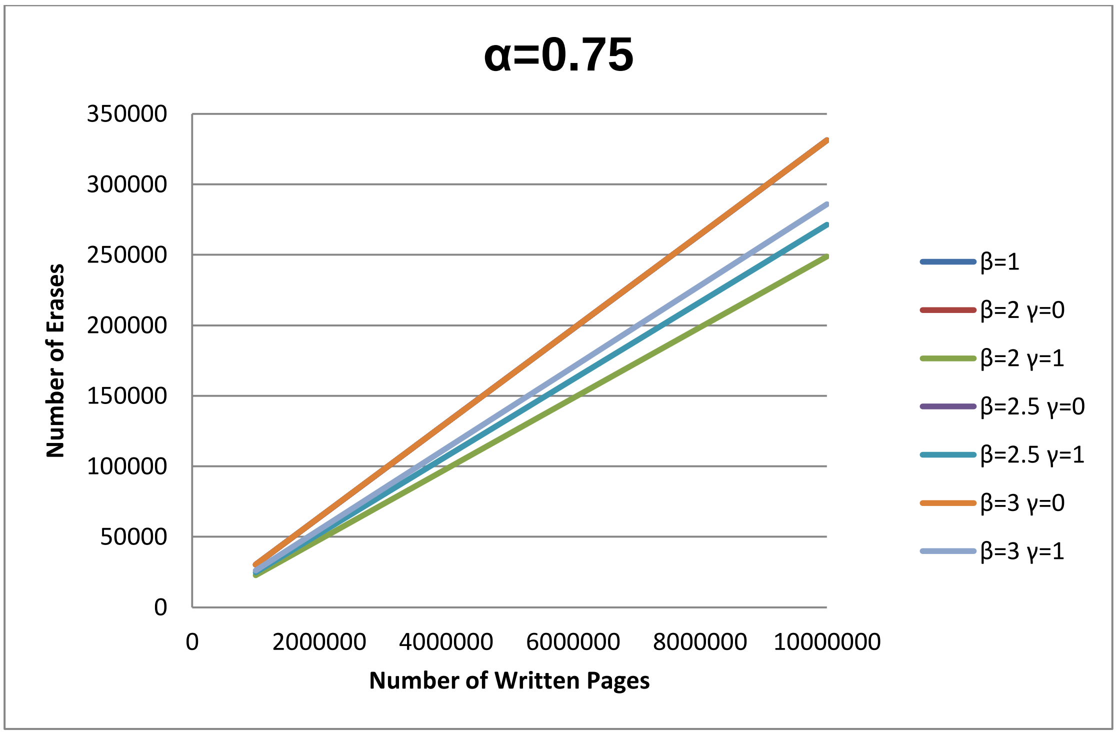
α=0.5	**β=1**	**β=2**		**β=2.5**		**β=3**	
		γ=0	**γ=1**	**γ=0**	**γ=1**	**γ=0**	**γ=1**
1000000	18025	14790	12894	16445	14049	17224	14801
2000000	37376	30777	26838	34180	29261	35752	30783
3000000	56712	46782	40806	51912	44500	54311	46792
4000000	76091	62764	54809	69668	59716	72857	62762
5000000	95457	78758	68750	87395	74923	91400	78749
6000000	114795	94759	82717	105141	90127	109927	94745
7000000	134153	110762	96688	122842	105341	128500	110701
8000000	153513	126747	110671	140621	120560	147045	126696
9000000	172874	142740	124665	158317	135777	165586	142702
10000000	192204	158721	138588	176054	151004	184164	158686
% Save		**17.53**	**28.02**	**8.47**	**21.57**	**4.26**	**17.53**



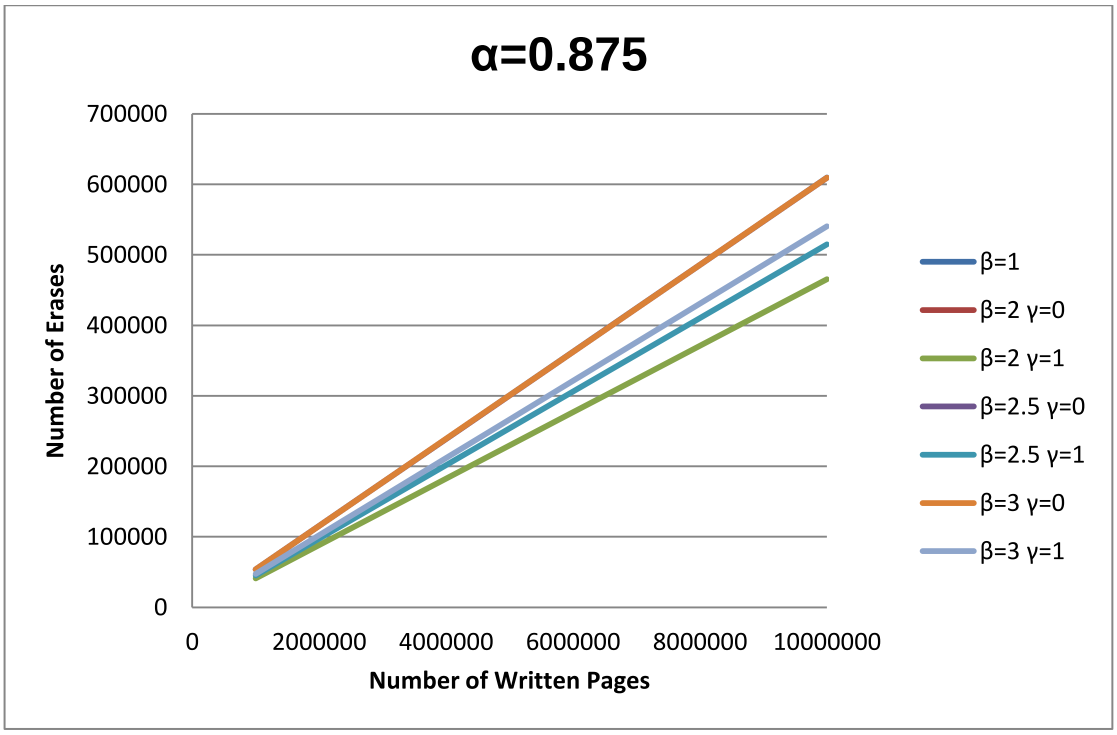
α=0.625	**β=1**	**β=2**		**β=2.5**		**β=3**	
		γ=0	**γ=1**	**γ=0**	**γ=1**	**γ=0**	**γ=1**
1000000	22059	20622	16213	21928	17694	22061	18505
2000000	45996	43146	33911	45704	36970	45983	38622
3000000	69912	65595	51600	69552	56230	69899	58712
4000000	93825	88144	69281	93367	75520	93838	78835
5000000	117731	110649	86958	117125	94819	117778	98949
6000000	141631	133131	104676	140953	114071	141647	119033
7000000	165534	155614	122338	164739	133337	165540	139169
8000000	189450	178068	140094	188553	152632	189465	159288
9000000	213332	200538	157751	212337	171844	213407	179362
10000000	237277	223076	175442	236147	191192	237241	199521
% save		**6.09**	**26.16**	**0.51**	**19.52**	**0.00**	**15.97**



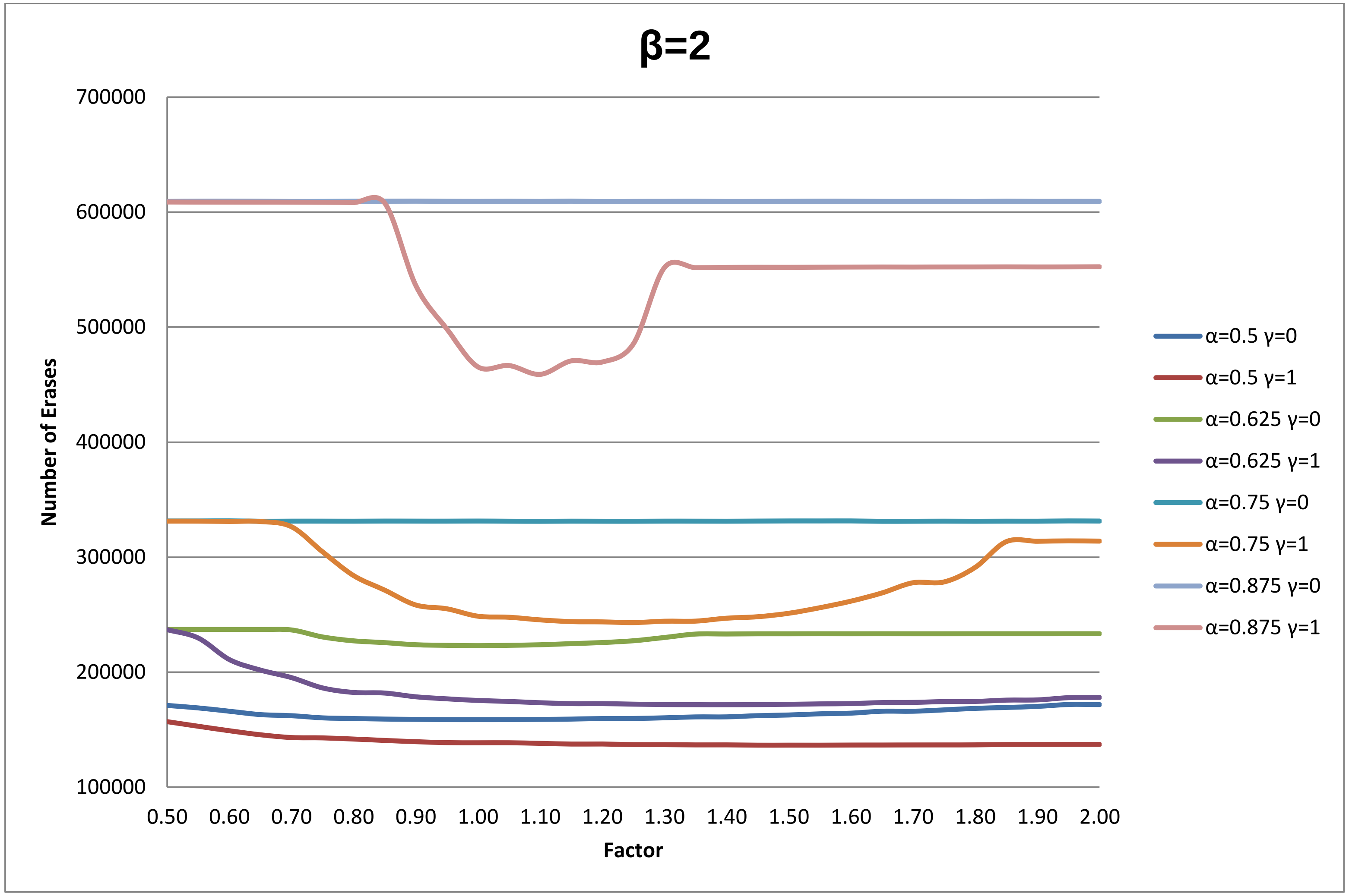
α=0.75	β=1	β=2		β=2.5		β=3	
		γ=0	γ=1	γ=0	γ=1	γ=0	γ=1
1000000	30331	30267	22722	30302	24775	30316	26081
2000000	63782	63705	47825	63742	52136	63790	54960
3000000	97209	97157	72943	97161	79532	97241	83841
4000000	130721	130665	98074	130630	106941	130672	112724
5000000	164098	164069	123223	164145	134253	164079	141568
6000000	197567	197644	148368	197472	161701	197599	170411
7000000	230992	231084	173462	230980	189067	231018	199328
8000000	264368	264433	198592	264359	216477	264460	228191
9000000	297862	297888	223669	297789	243872	297884	257079
10000000	331390	331343	248865	331263	271395	331285	285953
% save		**0.03**	**24.95**	**0.04**	**18.18**	**0.00**	**13.76**



α=0.875	**β=1**	**β=2**		**β=2.5**		**β=3**	
		γ=0	**γ=1**	**γ=0**	**γ=1**	**γ=0**	**γ=1**
1000000	53757	53746	41002	53722	45186	53788	47414
2000000	115539	115523	88273	115565	97374	115520	102310
3000000	177304	177308	135371	177083	149574	177218	156984
4000000	238878	238846	182553	238851	201674	239172	211824
5000000	300838	300551	229632	300435	253888	300744	266435
6000000	362588	362387	276804	362299	306141	362329	321355
7000000	424407	424110	323806	424084	358265	424109	376102
8000000	485934	486016	371070	485690	410406	485844	430882
9000000	547765	547787	418216	547541	462581	547834	485655
10000000	609749	609433	465458	609264	514884	609410	540527
% save		**0.03**	**23.65**	**0.06**	**15.63**	**0.01**	**11.42**



Finally, the results of this algorithm with a factor for β=2 and different values of α,γ:


β=2
Number of Erases
Factor
α=0.5 γ=0
α=0.5 γ=1
α=0.625 γ=0
α=0.625 γ=1
α=0.75 γ=0
α=0.75 γ=1
α=0.875 γ=0
α=0.875 γ=1