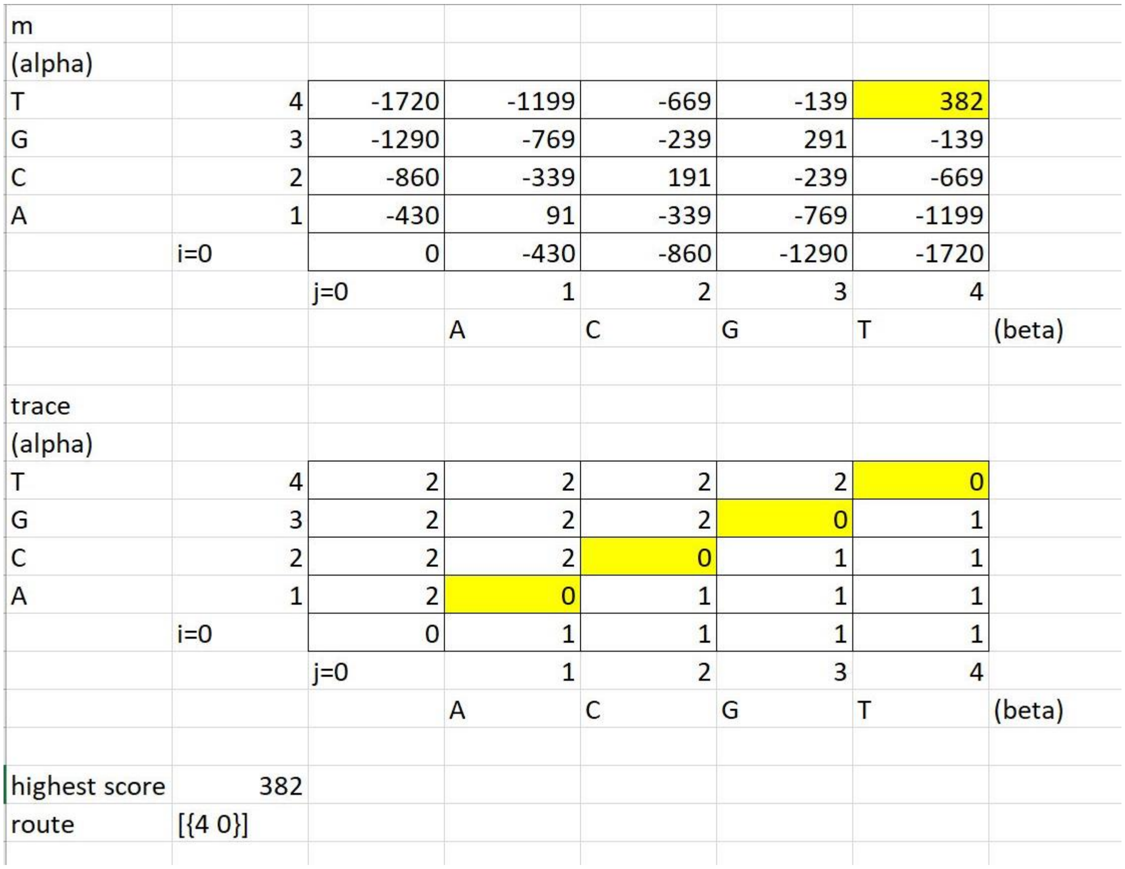
constGap_alignLowMem3_README

Yanting “Raven” Luo

Prior to the alignLowMem modifications, this is how constGap alignment would work. For the example sequences alpha (ACGT) and beta (ACGT), an m matrix of 5x5 would be made to keep track of alignment scores including the highest score, and a trace matrix of 5x5 would be made to allow for traceback and writing cigar routes. The highest score 382 is highlighted in the m matrix, and the cigar route is [{4 0}] as highlighted in the trace matrix. Each 0 means that the in the 2x2 around the square in the m matrix (-139, 291, and -139 around 382), the biggest number (most positive number) is found in the bottom left corner, representing a “match” in the alignment. Note that the top right number is not compared, and only the top left, bottom left, and bottom right are compared using the tripleMaxTrace function to find the maximum.



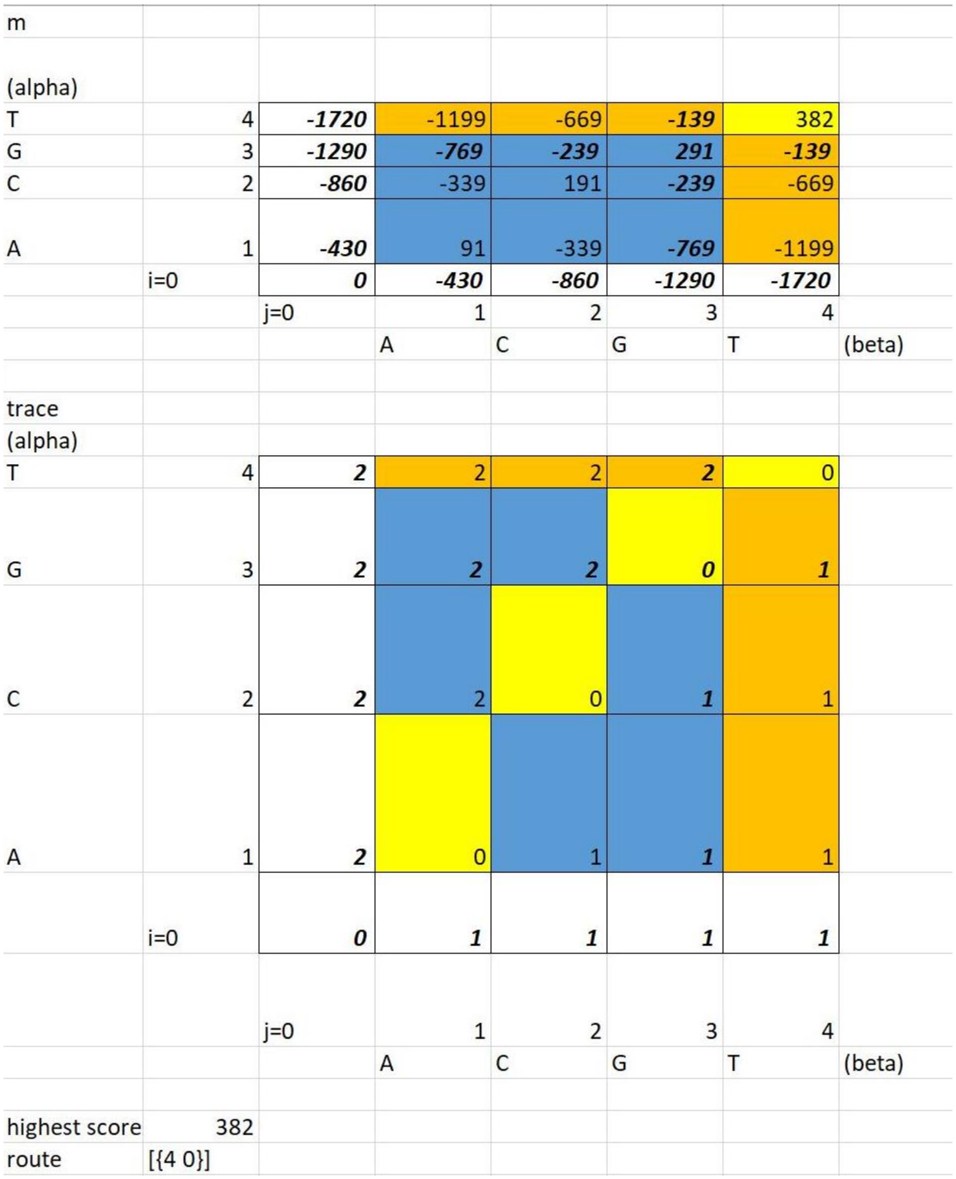
m							
(alpha)							
T	4	-1720	-1199	-669	-139	382	
G	3	-1290	-769	-239	291	-139	
C	2	-860	-339	191	-239	-669	
A	1	-430	91	-339	-769	-1199	
	i=0	0	-430	-860	-1290	-1720	
		j=0	1	2	3	4	
			A	C	G	T	(beta)

trace							
(alpha)							
T	4	2	2	2	2	0	
G	3	2	2	2	0	1	
C	2	2	2	0	1	1	
A	1	2	0	1	1	1	
	i=0	0	1	1	1	1	
		j=0	1	2	3	4	
			A	C	G	T	(beta)

highest score	382
route	[{4 0}]

Earlier alignLowMem implementations replaced the m matrix with “mRowPrevious” and “mRowCurrent”. In the example, mRowPrevious and mRowCurrent correspond to 2 slices, each of size 5, so that only 2 rows need to be saved at a time rather than the entire 5x5 matrix. For instance, when mRowPrevious is the i=0 row, it is used as the basis to fill out mRowCurrent which is the i=1 row. Then, when i=1 Is completely filled, it becomes mRowPrevious, which is the basis for filling the i=2 row, the new mRowCurrent. The memory is recycled, and scores from the i=0 row are overwritten.

In alignLowMem3, I further reduce memory by dividing up both m and trace matrices into checkerboards. In the example, each checkerboard is highlighted blue or orange. I use a checkersize of 3, so a full checkerboard would be 3x3, but at the end of the alpha and beta sequences there are incomplete checkerboards because the sequence lengths are not perfectly divisible by the checkersize.



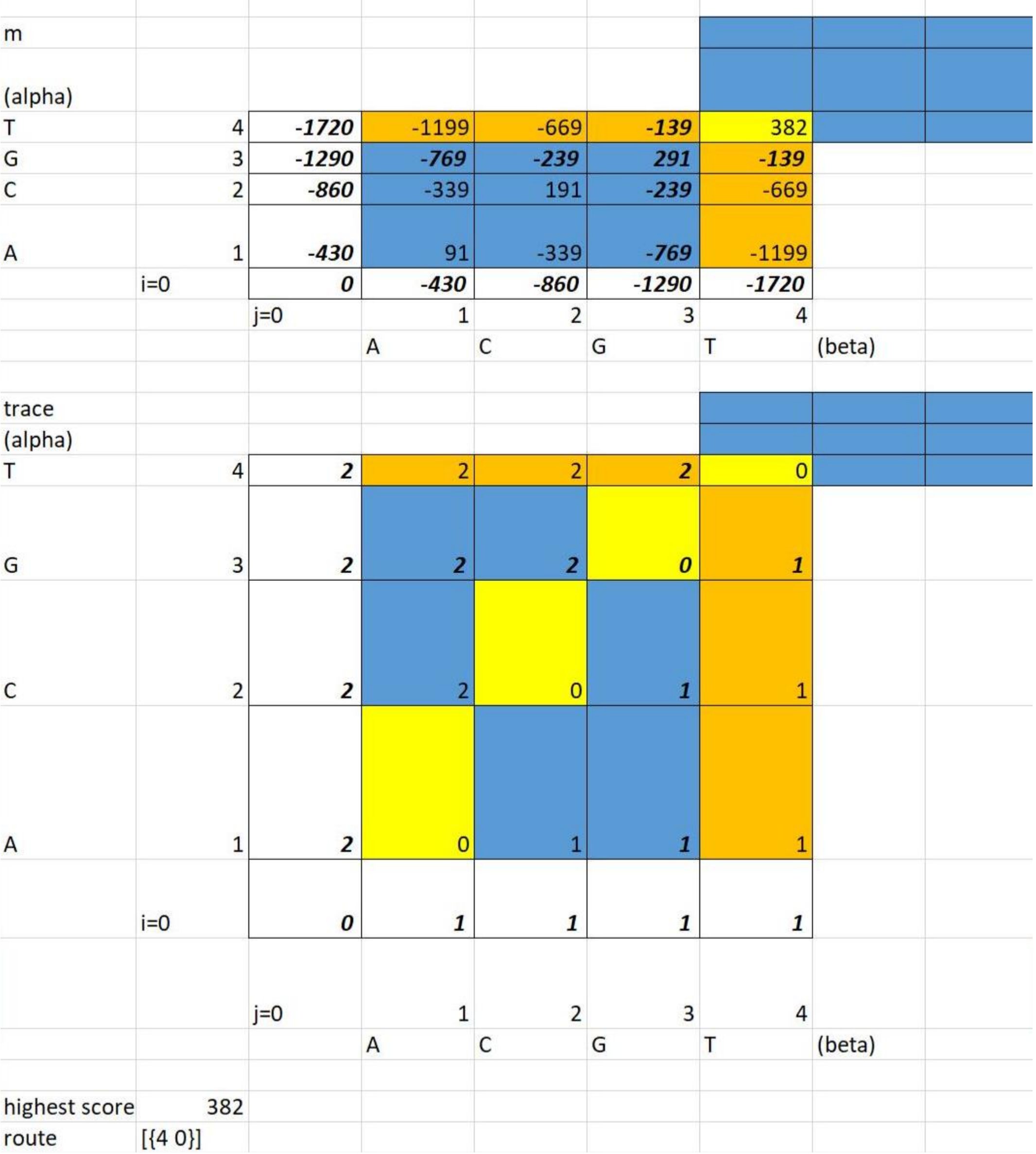
m							
(alpha)							
T	4	***-1720***	-1199	-669	***-139***	382	
G	3	***-1290***	***-769***	***-239***	***291***	***-139***	
C	2	***-860***	-339	191	***-239***	-669	
A	1	***-430***	91	-339	***-769***	-1199	
	i=0	***0***	***-430***	***-860***	***-1290***	***-1720***	
		j=0	1	2	3	4	
			A	C	G	T	(beta)

trace							
(alpha)							
T	4	***2***	2	2	***2***	0	
G	3	***2***	***2***	***2***	***0***	***1***	
C	2	***2***	2	0	***1***	1	
A	1	***2***	0	1	***1***	1	
	i=0	***0***	***1***	***1***	***1***	***1***	
		j=0	1	2	3	4	
			A	C	G	T	(beta)

highest score	382
route	[{4 0}]

In alignLowMem3, Step 1 is to find the highest score. This is implemented using the “mRowPrevious” and “mRowCurrent” system, but only scores are recorded. No trace matrix is made or filled at this step. During Step 1, I also save the rows and columns necessary to fill and traceback the checkerboards. In the example, the bolded and italicized rows i=0 and i=3 are saved in trace_prep_i, and columns j=0 and j=3 are saved in trace_prep_j.

Step 2 and Step 3 are filling checkerboards and writing cigars, respectively. They are implemented in a loop. The loop initializes in the first checkerboard to fill based on the position of the highest score, using the formulae k1=int((score_highest_i-1)/checkersize), and k2=int((score_highest_j-1)/checkersize). In the example, score_highest occurs at i=4, j=4, so k1=int((4-1)/3)=1, k2=int((4-1)/3)=1, meaning the first checkerboard to be filled is the top right blue matrix, with i ranging from k1*checkersize+1=4 to (k1+1)*checkersize=6, and j ranging from k2*checkersize+1=4 to (k2+1)*checkersize=6. Using the corresponding row i=3 and column j=3 saved in the trace_prep_i and trace_prep_j matrices, I now have what I need to fill the m and trace for the checkerboard, i.e. the initial mRowPrevious and left-most position of mRowCurrent with respect to the checkerboard. However, since this checkerboard is at the end of the alpha and beta sequences and exceed score_highest_i and score_highest_j, only the i=4 j=4 square will be filled in Step 2. In Step 3, a cigar route is initiated by identifying the 0 at the i=4 j=4 square in the trace matrix, which is the score_highest_i and score_highest_j. The cigar route is now {1 0}. Then, because the checkerboard happens to be filled up already immediately after score_highest_i and score_highest_j, k1 and k2 will be updated to the next appropriate checkerboard based on the cigar route. The 0 indicates tracing back diagonally left and down, so the program enters the k1=0, k2=0 checkerboard (bottom left blue matrix), and repeats Step 2 and Step 3 to fill the checkerboard and add the cigar route {3 0} to the existing cigar, resulting in the grown cigar {4 0}.



m							
(alpha)							
T	4	-1720	-1199	-669	-139	382	
G	3	-1290	-769	-239	291	-139	
C	2	-860	-339	191	-239	-669	
A	1	-430	91	-339	-769	-1199	
	i=0	0	-430	-860	-1290	-1720	
		j=0	1	2	3	4	
			A	C	G	T	(beta)

trace							
(alpha)							
T	4	2	2	2	2	0	
G	3	2	2	2	0	1	
C	2	2	2	0	1	1	
A	1	2	0	1	1	1	
	i=0	0	1	1	1	1	
		j=0	1	2	3	4	
			A	C	G	T	(beta)

highest score	382
route	[{4 0}]

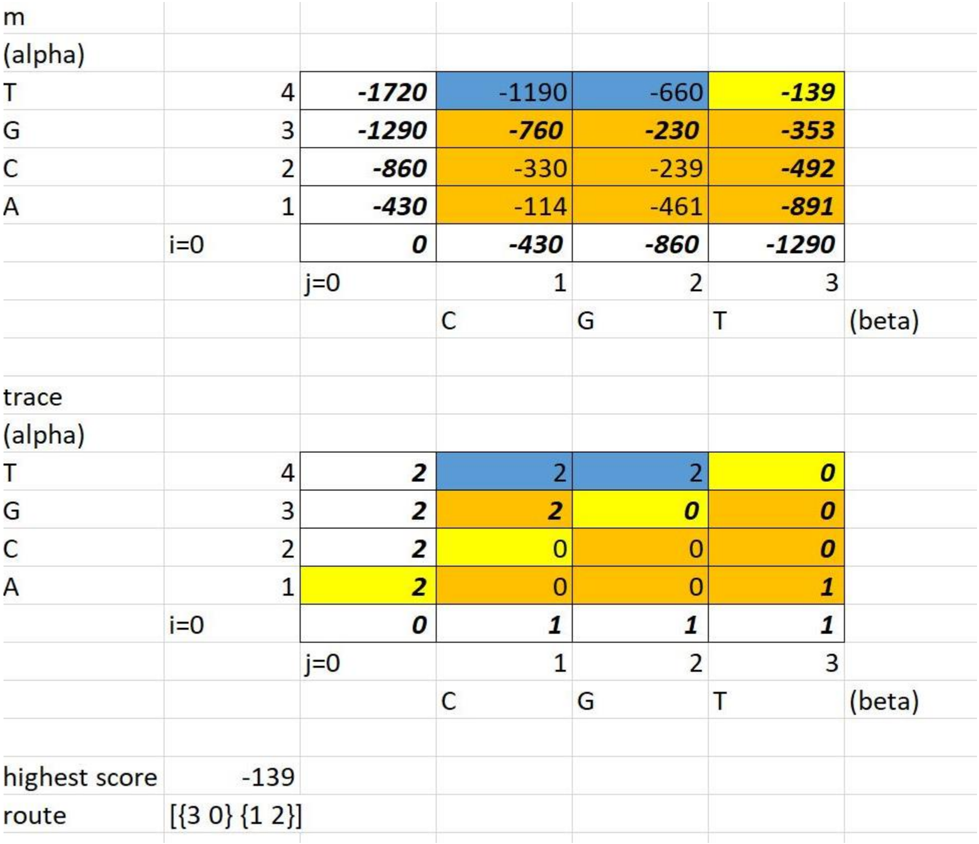
When the loop containing Step 2 and Step 3 ends, Step 4 writes the last cigar. In the above example aligning alpha (ACGT) and beta (ACGT), the cigar from Step 3 ends at the 0 at i=1, j=1. The 0 indicates tracing back diagonally left and down, but there is no sequence at the i=0, j=0 position, so no more cigar is added. However, in another example shown below aligning alpha (ACGT) and beta (CGT), the last iteration of Step 3 ends at the 0 at i=2, j=1, which indicates tracing back diagonally left and down to i=1, j=0, which is a square that represents sequence and has a trace score of 2, so {1 2} is added to the final cigar. When Step 4 ends, the cigar route for the alignment is complete. This final cigar is informative enough for me to recreate the alignment. In the example below, the final cigar [{3 0} {1 2}] indicates 3 matches and then 1 deletion. Because traceback is in reverse, the cigar indicates that for alpha (ACGT) and beta (CGT), the alignment starts with 1 deletion followed by 3 matches, resulting in the following alignment:

“

ACGT

-CGT

”



m						
(alpha)						
T	4	-1720	-1190	-660	-139	
G	3	-1290	-760	-230	-353	
C	2	-860	-330	-239	-492	
A	1	-430	-114	-461	-891	
	i=0	0	-430	-860	-1290	
		j=0	1	2	3	
			C	G	T	(beta)

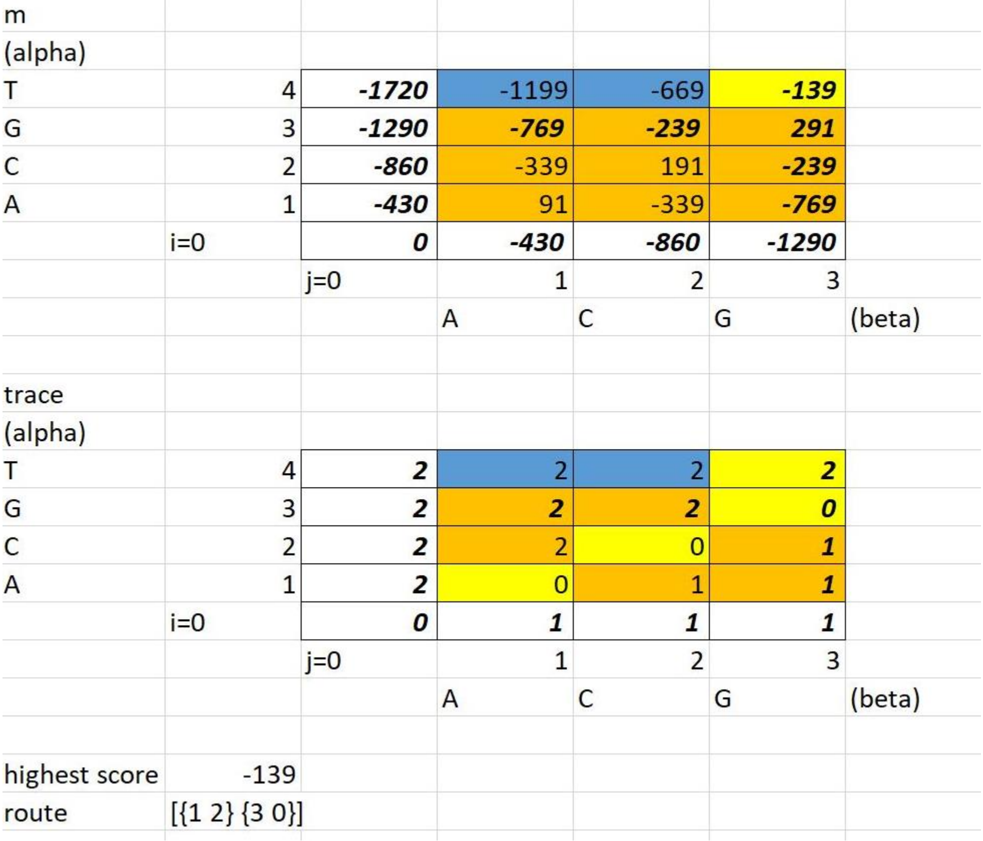
trace						
(alpha)						
T	4	2	2	2	0	
G	3	2	2	0	0	
C	2	2	0	0	0	
A	1	2	0	0	1	
	i=0	0	1	1	1	
		j=0	1	2	3	
			C	G	T	(beta)

highest score	-139
route	[{3 0} {1 2}]

Please note that each checkerboard is only filled up to where it is necessary/allowed. Not all checkerboards are completely filled. In the above example aligning alpha (ACGT) and beta (CGT), the k1=0, k2=0 matrix (orange) is only filled up to the 0 at the i=3, j=2 position in Step 2, and Step 3 starts from that position, rather than at i=3, j=3. This consideration further reduces memory in Step 2 and ensures that Step 3 writes the correct cigar based on the previous checkerboard (based on the 0 at i=4, j=3, the correct square to trace back to is i=3, j=2).

Also note that in the cigar, 0=mismatch (trace back diagonally left and down), 1=insertion (trace back horizontally left), 2=deletion (trace back vertically down).

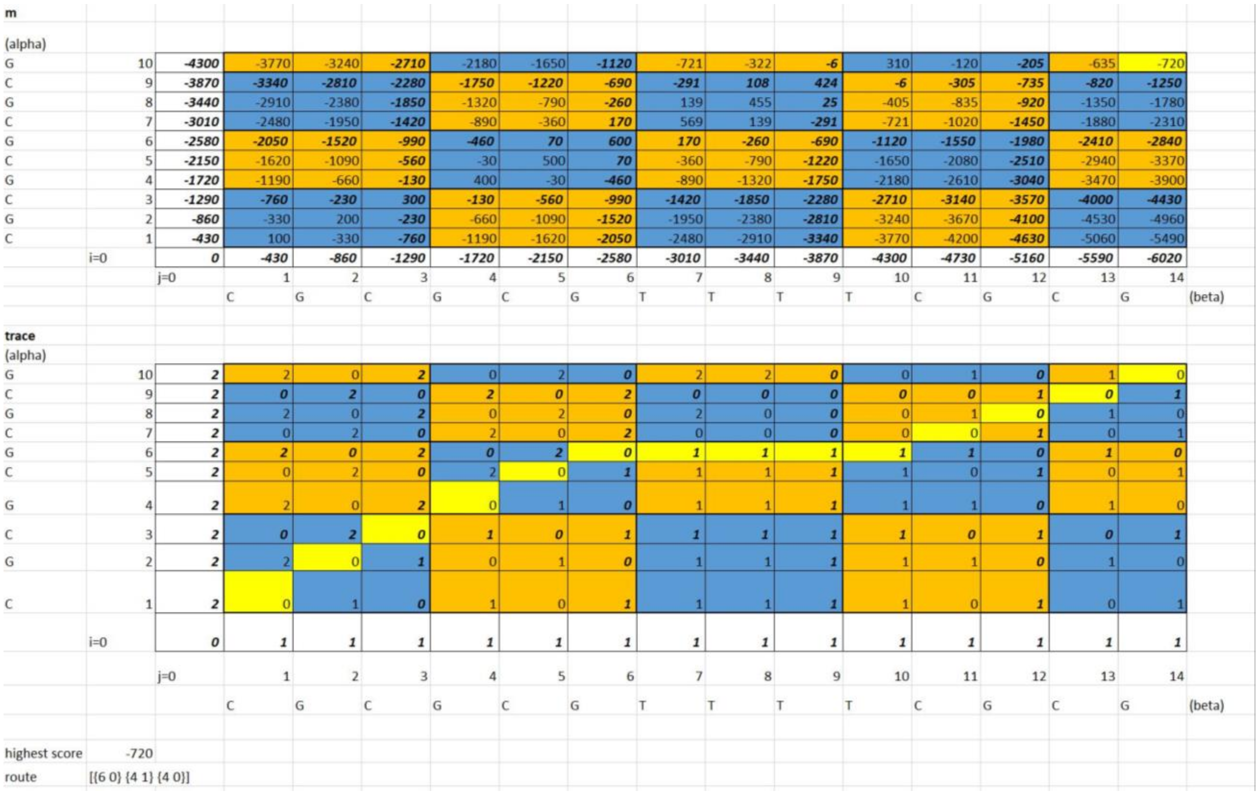
Below are a few more examples for reference.



m						
(alpha)						
T	4	***-1720***	-1199	-669	***-139***	
G	3	***-1290***	***-769***	***-239***	***291***	
C	2	***-860***	-339	191	***-239***	
A	1	***-430***	91	-339	***-769***	
	i=0	***0***	***-430***	***-860***	***-1290***	
		j=0	1	2	3	
			A	C	G	(beta)

trace						
(alpha)						
T	4	***2***	2	2	***2***	
G	3	***2***	***2***	***2***	***0***	
C	2	***2***	2	0	***1***	
A	1	***2***	0	1	***1***	
	i=0	***0***	***1***	***1***	***1***	
		j=0	1	2	3	
			A	C	G	(beta)

highest score	-139
route	[{1 2} {3 0}]



m

(alpha)

G	10	-4300	-3770	-3240	-2710	-2180	-1650	-1120	-721	-322	-6	310	-120	-205	-635	-720	
C	9	-3870	-3340	-2810	-2280	-1750	-1220	-690	-291	108	424	-6	-305	-735	-820	-1250	
G	8	-3440	-2910	-2380	-1850	-1320	-790	-260	139	455	25	-405	-835	-920	-1350	-1780	
C	7	-3010	-2480	-1950	-1420	-890	-360	170	569	139	-291	-721	-1020	-1450	-1880	-2310	
G	6	-2580	-2050	-1520	-990	-460	70	600	170	-260	-690	-1120	-1550	-1980	-2410	-2840	
C	5	-2150	-1620	-1090	-560	-30	500	70	-360	-790	-1220	-1650	-2080	-2510	-2940	-3370	
G	4	-1720	-1190	-660	-130	400	-30	-460	-890	-1320	-1750	-2180	-2610	-3040	-3470	-3900	
C	3	-1290	-760	-230	300	-130	-560	-990	-1420	-1850	-2280	-2710	-3140	-3570	-4000	-4430	
G	2	-860	-330	200	-230	-660	-1090	-1520	-1950	-2380	-2810	-3240	-3670	-4100	-4530	-4960	
C	1	-430	100	-330	-760	-1190	-1620	-2050	-2480	-2910	-3340	-3770	-4200	-4630	-5060	-5490	
	i=0	0	-430	-860	-1290	-1720	-2150	-2580	-3010	-3440	-3870	-4300	-4730	-5160	-5590	-6020	
		j=0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
			C	G	C	G	C	G	T	T	T	T	C	G	C	G	(beta)

trace

(alpha)

G	10	2	2	0	2	0	2	0	2	2	0	0	1	0	1	0	
C	9	2	0	2	0	2	0	2	0	0	0	0	0	1	0	1	
G	8	2	2	0	2	0	2	0	2	0	0	0	1	0	1	0	
C	7	2	0	2	0	2	0	2	0	0	0	0	0	1	0	1	
G	6	2	2	0	2	0	2	0	1	1	1	1	1	0	1	0	
C	5	2	0	2	0	2	0	1	1	1	1	1	0	1	0	1	
G	4	2	2	0	2	0	1	0	1	1	1	1	1	0	1	0	
C	3	2	0	2	0	1	0	1	1	1	1	1	0	1	0	1	
G	2	2	2	0	1	0	1	0	1	1	1	1	1	0	1	0	
C	1	2	0	1	0	1	0	1	1	1	1	1	0	1	0	1	
	i=0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		j=0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
			C	G	C	G	C	G	T	T	T	T	C	G	C	G	(beta)

highest score -720

route [{6 0} {4 1} {4 0}]