Supplement

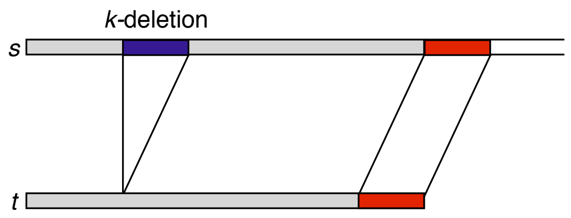

Fig. 1. When we find approximate matches of reads of length L in genomes, our approach compares 2-gram frequency vectors of the read with 2-gram frequency vectors of length L windows in the genome. If the read, t in this figure, derived from position 1 of the genome s by a deletion of size k (blue), the last k positions of t match $s[L+1, L+k]$. Affine edit distance is not tight if $s[L+1, L+k]$ at least partially matches the deletion. We argue that the probability of this happening is low.

Probability of catastrophic failure: For edit operations which affect positions q letters apart, it is easy to show that the inequality $L_1 \leq$ AED is sharp, that is L_1 gives the edit distance with affine gap costs. Consequently finding matches of minimal L_1 prefers matches with fewer indels over matches with frequent substitutions. However, the lower bound can be still arbitrarily bad (when one string is a rotation or transposition of another), but the probability is small. The 2-gram frequency vectors can be naturally viewed as the sufficient statistics of a first-order Markov chain on nucleotides, and we will view all our points, both reads and genomic positions, as first-order Markov chains.

Consider the fixed L-length window s (Fig. 1) in the reference genome and its q-spectrum $c_q(s)$. How is $c_q(s)$ altered by the deletion? A total of $k+1$ subtractions by one—one for each pair of nucleotides from one position before the deletion to one position after the deletion—is applied to not more than $k+1$ coordinates of $c_q(s)$. As t is also of length L, the deletion has to be balanced by k additional characters from s. This introduces $k-1$ pairs (red in Fig. 1) at the end, one containing the character before and the character following the deletion and one pair (s_L, s_{L+1}), which were not present in $s[1:L]$. These $k+1$ additional pairs give a total of $k+1$ additions of 1 to not more than $k+1$ coordinates of $c_q(s)$. It can happen that these $k+1$ subtractions and $k+1$ additions exactly cancel, yielding $L_1(s,t)=0$ even though the Levenshtein distance $\mathrm{ED}(s,t)=k$. Under the Markov assumption we can now compute an upper bound for the probability $P(L_1(s,t)=0|\,\mathrm{ED}(s,t)=k)$ if we consider exactly one deletion of length k as in Fig. 1. If we look at a single 2-gram within the deletion then the probability that it appears in the red part in Fig. 1—hence the subtraction of the count due to the deletion and is offset by the addition of the count—is largest, when it is the most frequent 2-gram. Assuming the two subtractions involving pairs across the border of the deletion in s and the pairs covering the deletion in t and across the boundary to the red part in t cancels out, the probability of finding exactly the $k-1$ deleted counts (contained inside the deletion of s) compensated by the added counts at the end of t is bounded from above by

$$P(L_1(s,t)=0|\,\mathrm{ED}(s,t)=k, \text{one deletion}) \leq \binom{k-1}{c_A, c_C, c_G, c_T} P_1^* \leq \frac{(k-1)!}{\left(\frac{k-1}{4}!\right)^4} (P_2^*)^{k-1},$$

where c_A, c_C, c_G, c_T are the nucleotide counts in the deletion, P_1^* is the maximal probability of a realization of length $k-1$ produced by the underlying Markov chain and P_2^* the maximal transition probability in the Markov chain.

Memory reduction: We create d-dimensional q-gram frequency vectors by shifting a window of size l (read length) over a genome of size G. This naive approach uses $O(Gd)$ amount of memory. To reduce this large memory requirement for storing count vectors, we apply two techniques. First, we only consider every g-th genomic window for computing frequency vector, which brings down the memory requirement to $O(\frac{Gd}{g})$. Second, we observe that consecutive g-th windows can have at most $2g$ differences in their frequency vectors. We exploit this observation by not storing the left α vectors and right α vectors of a particular vector V. Instead we define these 2α vectors by their differences from V. When we need these vectors in later stages we compute them on-the-fly. After applying these two ideas, the total memory requirement for storing count vectors is $\frac{Gd}{g(2\alpha+1)} + \frac{2G\alpha(\alpha+1)}{2\alpha+1}$.

Table 1. Running times for read mappers used for evaluating simulated data.

Read mapper		S1			S2			S3			S4		
	Version	125	200	500	125	200	500	125	200	500	125	200	500
Bowtie (-v 3)	0.12.7	0:05	0:03		0:03	0:03		0:04	0:04		0:03	0:06	
BWA	0.5.9	0:07	0:04		0:11	0:07		0:08	0:10		0:13	0:29	
BWA (-n 50 -o 10 -e 50 -M 1 -O 3 -E 1)		1:30	2:34		4:30	5:55		3:14	4:42		3:44	4:57	
SOAP2 (-r 1 -g 10 -v 50)	2.21	0:06	0:03		0:05	0:05		0:06	0:05		0:03	0:08	
mrFAST (–best)	2.1.0.0	0:52	0:46		0:56	0:48		1:19	1:20		0:48	1:08	
mrFAST (-e 6)		1:54	1:28		1:41	1:37		2:47	2:28		1:49	2:10	
Novoalign (-l 0 -e 1 -r Random)	2.07.13	0:08	0:12		0:14	0:23		0:10	0:17		0:16	0:23	
SSAHA2 (–best -1)	2.5.5	4:30	8:42		5:16	9:23		8:21	16:43		6:50	14:38	
TreQ ($\tau = 1$, $\beta = 10000$, $\alpha = 0$)	dev	2:29	2:55	3:37	2:37	3:01	3:43	2:33	3:03	4:09	2:40	2:31	3:24
LAST w/ LAMA		0:43	1:51	10:22	0:33	1:18	7:48	0:57	2:32	14:48	0:40	1:46	10:50
LAST w/ LAMA (-d108 -e120)		1:07	2:29	12:33	0:52	1:52	9:15	1:27	3:25	18:35	0:59	2:25	13:15
Stampy		0:18	0:31	1:35	0:37	1:13	5:00	0:40	1:16	4:41	0:36	1:05	3:27
Stampy w/ BWA		0:10	0:19	1:05	0:41	1:18	4:35	0:30	1:00	3:46	0:40	1:12	3:40

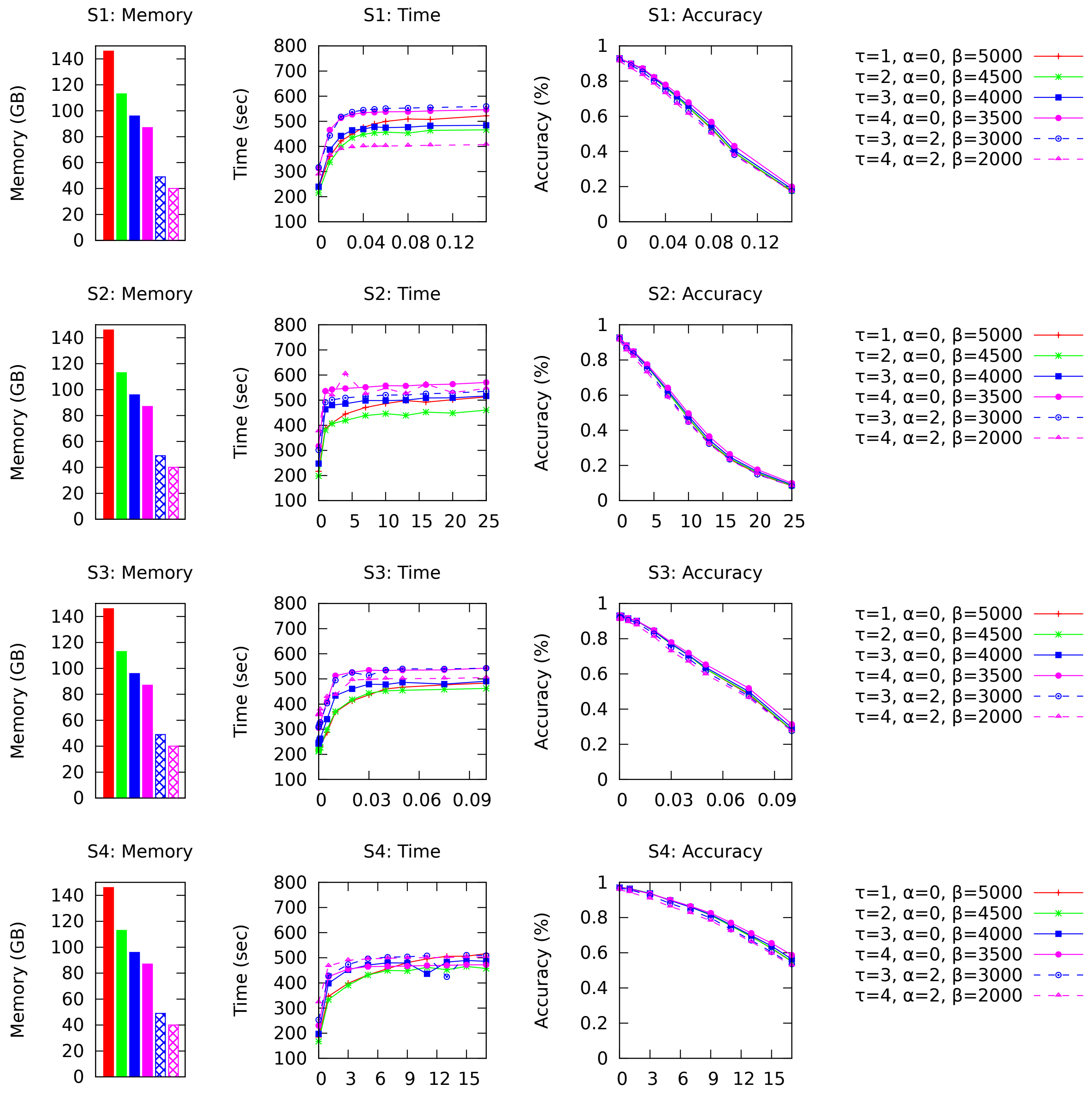

Fig. 2. Effect of different parameters on TreQ. Simulated datasets S1, S2, S3, and S4 (for details see Section: Discussion) of read length 100bp are tested with different parameter choices for TreQ. The memory requirement for TreQ comes down from 150GB ($\tau = 1$, $\alpha = 0$) to 40GB ($\tau = 4$, $\alpha = 2$) while a careful selection of β achieves equivalent mapping accuracy in similar amount of time.