Table 3: IUPAC alphabet (Σ_{IUPAC}).

Symbol	A	C	G	T
Bases	A	C	G	T
Symbol	U	R	Y	K
Bases	U	A,G	C,T	G,T
Symbol	M	S	W	B
Bases	A,C	G,C	A,T	C,G,T
Symbol	D	H	V	N
Bases	A,G,T	A,C,T	A,C,G	A,C,G,T

Determining neighbors

In order to quickly find all the existing neighbors of a motif within the allowed error thresholds, EXMOTIF first computes all the exact structured motifs, and stores them into a hash table to facilitate fast lookup. Then for each extracted structured motif $\mathcal{M}$, EXMOTIF enumerates all its possible neighbors and checks whether they exist in the hash table. One problem is that the number of possible neighbors of $\mathcal{M}$ can be quite large. When we allow ε_i substitutions for simple component M_i in $\mathcal{M}$, for $1 \le i \le k$, the number of $\mathcal{M}$'s neighbors is given as $\prod_{i=1}^{k}[\sum_{j=0}^{e_i}\binom{|M_i|}{j}\cdot 3^j]$. For example, for $\mathcal{M}$ = AACGTT[1,5]AGTTCC, when we allow one substitution for each simple motif, the number of its neighbors is 361; when we allow two substitutions per component, the number of its neighbors is 23,716. Instead of enumerating the potentially large number of neighbors (many of which may not even occur in the sequence set $\mathcal{S}$) for each structured motif $\mathcal{M}$ individually, EXMOTIF utilizes the observation that many motifs have shared neighbors, and thus previously computed support information can be reused. EXMOTIF enumerates neighbors in two steps. In the first step, for each $\mathcal{M}$, it enumerates *aggregate* neighbor motifs, replacing the allowed number of errors e_i with as many 'N' symbols (which stands for A,C,G, or T). The number of possible aggregate neighbors is given as $\prod_{i=1}^{k}\binom{|M_i|}{\varepsilon_i}$. The second step, it computes the support for each aggregate neighbor by expanding each 'N' with each DNA symbol, looking up the hash table for the support of the corresponding motif, and adding the supports for all matching motifs. Since the motifs matching an aggregate are also neighbors of each other, the support of the aggregate can be re-used to compute the support of other matching motifs as well. Once the supports for all aggregate neighbors have been computed, the final support of the structured motif $\mathcal{M}$ can be obtained. Thus for each $\mathcal{M}$, the number of "neighbors" to consider can be as low as $\prod_{i=1}^{k}\binom{|M_i|}{\varepsilon_i}$!

For example, consider the example shown in Figure 4. Consider the structured motif $\mathcal{M}$ = TAA[0,3]GG[1,3]CCTT (taken from our example in Table 1); assume that $\varepsilon_1 = 1$, $\varepsilon_2 = 0$ and $\varepsilon_3 = 1$. There are three possible aggregates for TAA, namely TAN, TNA, and NAA, and four aggregates for CCTT, namely CCTN, CCNT, CNTT, and NCTT, giving a total of 12 aggregate neighbors for $\mathcal{M}$, as illustrated in the figure. EXMOTIF processes each aggregate neighbor in turn. Using a hash-table (or direct lookup table if there are only a few neighbors), it checks if the aggregate neighbor has been processed previously. If yes, it moves on to the next aggregate. If not, it gathers the support information from all of its matching structured motifs, to compute its total support. Next, it also updates the neighbor support value for each of the matching motifs, so that once an aggregate is processed, we no longer require its information. All we need to know is whether it has been processed or not. For example, once the support of the first aggregate TAN[0,3]GG[1,3]CCTN for the example motif $\mathcal{M}$ above is computed, EXMOTIF also updates the neighbor supports for all other matching structured motifs, such as $\mathcal{M}'$ = TAC[0,3]GG[1,3]CCTG. Later when processing $\mathcal{M}'$, EXMOTIF can skip the above aggregate and focus on the not yet processed aggregates, e.g., NAC[0,3]GG [1,3]NCTG, and so on.

The pseudo-code for arbitrary substitutions is given in Figure 5. The procedure takes as input the hash-table $\mathbb{H}$ containing all structured motifs $\mathcal{M}$ and their supports $\pi(\mathcal{M})$, the quorum q, and the per simple motif errors e_i or the glo-