http://www.ncbi.nlm.nih.gov/blast/discontiguous.shtml

What is discontiguous Mega BLAST?

This version of Mega BLAST is designed specifically for comparison of diverged sequences, especially sequences from different organisms, which have alignments with low degree of identity, where the original Mega BLAST is not very effective. The major difference is in the use of the 'discontiguous word' approach to finding initial offset pairs, from which the gapped extension is then performed.

Both Mega BLAST and all previous versions of nucleotide-nucleotide BLAST look for exact matches of certain length as the starting points for gapped alignments. When comparing less conserved sequences, i.e. when the expected share of identity between them is e.g. 80% and below, this traditional approach becomes much less productive than for the higher degree of conservation.

Depending on the length of the exact match to start the alignments from, it either misses a lot of statistically significant alignments, or on the contrary finds too many short random alignments.

According to [1], as well as our own probability simulations, it turns out that if initial 'words' are based not on the exact match, but on a match of a certainset of nonconsecutive positions within longer segments of the sequences, the productivity of the word finding algorithm is much higher. This way fewer words are found overall, but more of them end up producing statistically significant alignments, than in the case of contiguous words of the same, and even shorter length than the number of matched positions in the discontiguous word.

As an example, we can define a pattern (template) of 0s and 1s of length e.g. 21: 100101100101100101101. For each pair of offsets in the query and subject sequences that are being compared, we compare the 21 nucleotide segments in these sequences ending at these offsets, and require only those positions in those segments to match that correspond to the 1s in the above template.

There are several advantages in using this approach. First, the conditional probabilities of finding word hits satisfying discontiguous templates given the expected identity percentage in the alignments between two sequences, are higherthan for contiguous words with the same number of positions required matched.

If two word hits are required to initiate a gapped extension, the effect of the discontiguous word approach is even larger. In both cases higher sensitivity is achieved because there is less correlation between successive words as the database sequence is scanned across the query sequence.

Second, when comparing coding sequences, the conservation of the third nucleotides in every codon is not essential, so there is no need to require it when matching initial words. This implies the advantage of using templates based on the '110' pattern, which are called 'coding'.

Finally, to achieve even higher sensitivity, one might combine two different discontiguous word templates and require any one of them to match at a given position to qualify it for the initial word hit.

The following options specific to this approach are supported:

Template length: 16, 18, 21.
Word size (i.e. number of 1s in the template): 11, 12
Template type: coding, non-coding.
Require two words for extension: yes/no.

The 'coding' templates are based on the 110 pattern, although more 0s are required for most of them, so some of the patterns become 010 or 100. These are the most effective for comparison of coding regions.

The non-coding templates attempt to minimize the correlation between successive words, when the database sequence is shifted by 4 positions against the query sequence. This means more 1s are concentrated at the ends of the template (at least 3 on each side).

When the option to require two words for extension is chosen, two word hits matching the template must be found within a distance of 50 nucleotides of one another.

Below are the exact discontiguous word template patterns for different combinations of word sizes and lengths:

```
   W = 11, t = 16, coding:     1101101101101101
   W = 11, t = 16, non-coding: 1110010110110111
   W = 12, t = 16, coding:     1111101101101101
   W = 12, t = 16, non-coding: 1110110110110111
   W = 11, t = 18, coding:     101101100101101101
   W = 11, t = 18, non-coding: 111010010110010111
   W = 12, t = 18, coding:     101101101101101101
   W = 12, t = 18, non-coding: 111010110010110111
   W = 11, t = 21, coding:     100101100101100101101
   W = 11, t = 21, non-coding: 111010010100010010111
   W = 12, t = 21, coding:     100101101101100101101
   W = 12, t = 21, non-coding: 111010010110010010111
```

[1] Ma, B., Tromp, J., Li, M., "PatternHunter: faster and more sensitive homology search", Bioinformatics 2002 Mar;18(3):440-5.