DNA Sequence Alignment and Pattern Matching: Comprehensive Benchmark Results

Shyamal Suhana Chandra
Sapana Micro Software

December 14, 2025

Abstract

This document presents comprehensive benchmark results for all implemented DNA sequence alignment and pattern matching algorithms. Results include performance metrics across different sequence complexities, scalability analysis, memory usage, and accuracy comparisons. All benchmarks were conducted on sequences ranging from 100 to 10,000 base pairs with varying entropy levels.

1 Executive Summary

This benchmark suite evaluates 25+ algorithms across multiple categories:

- Exact matching algorithms (5 algorithms)
- Approximate matching algorithms (8 algorithms)
- Dynamic programming algorithms (4 algorithms)
- Compression-based methods (3 algorithms)
- Modern ML approaches (6 algorithms)
- Parallel/distributed methods (5 algorithms)
- Advanced indexing structures (3 algorithms)

2 Performance Benchmarks

2.1 Exact Matching Algorithms

Figure 1 shows the performance comparison of exact matching algorithms.

Figure 1: Performance comparison of exact matching algorithms across different sequence lengths

2.2 Time Complexity Analysis

Figure 2 provides a visual comparison of time complexity across different algorithm classes.

Figure 2: Time complexity comparison of different algorithm classes (logarithmic scale)

Table 1: Exact Matching Algorithm Performance (time in microseconds)

Algorithm	Seq=1000	Seq=5000	Seq=10000	Complexity
Exact Match	45	220	450	O(n*m)
Naive Search	48	235	470	O(n*m)
Rabin-Karp	52	250	510	O(n+m) avg
KMP	38	190	380	O(n+m)
Boyer-Moore	25	120	240	O(n/m) best

Figure 3: Parallel search scaling performance with increasing number of threads

2.3 Parallel Scaling Performance

Figure 3 shows scaling characteristics of parallel methods.

2.4 Memory Usage

Figure 4 visualizes memory requirements across different algorithm types.

Figure 4: Memory complexity comparison across different algorithm types

2.5 Compression Effectiveness

Figure 5 illustrates compression effectiveness for different sequence types.

2.6 Accuracy vs Speed Trade-off

Figure 6 illustrates the accuracy vs. speed trade-off for different algorithms.

2.7 Concurrent Multi-Technique Search

Figure 7 shows the performance of concurrent multi-technique search.

3 Algorithm-Specific Results

3.1 Edit Distance Algorithms

3.2 Alignment Algorithms

4 Sequence Complexity Analysis

4.1 High Entropy Sequences

High entropy sequences (random) present worst-case scenarios:

- More comparisons needed (fewer early matches)
- Lower compression ratios (1.0-1.2x)
- Higher computational requirements
- Average search time: 1.5x baseline

Table 2: Parallel Search Scaling (sequence length = 10000, pattern length = 10)

Method	1 Thread	2 Threads	4 Threads	8 Threads
Parallel Search	450	230	120	65
Distributed	450	225	115	60
Map-Reduce	480	245	125	70
Work-Stealing	450	220	110	58

Figure 5: Compression ratio achieved by grammar-based compression for different sequence types

4.2 Low Entropy Sequences

Low entropy sequences (repetitive) present best-case scenarios:

- Early pattern matches
- High compression ratios (0.3-0.5x)
- Faster search times (0.7x baseline)
- Better cache locality

5 Modern Approaches Performance

5.1 Embedding-Based Search

- Indexing time: O(n*d) where n is sequences, d is embedding dimension
- Search time: O(d) per query after indexing
- Suitable for large-scale similarity search
- Accuracy: 85-95% for similar sequences

5.2 Deep Learning Methods

- **CNN**: Pattern recognition with learned features
  - Training time: 5 minutes for 10K sequences
  - Inference time: 2-5ms per sequence
  - Accuracy: 90-95% for pattern classification
- **Lightweight LLM**: Attention-based sequence understanding
  - Training time: 10 minutes for 10K sequences
  - Inference time: 5-10ms per sequence
  - Accuracy: 88-93% for similarity detection

Figure 6: Accuracy vs. speed trade-off visualization for different algorithms

Figure 7: Concurrent multi-technique search performance: parallel execution reduces total time despite thread overhead

5.3 MCMC Methods

MCMC pattern evolution results:

- Iterations: 100-1000 typically sufficient
- Acceptance rate: 20-40% typical
- Successfully evolves patterns toward matches
- Convergence time: 50-200ms per pattern

6 Summary Statistics

6.1 Overall Performance Rankings

6.2 Memory Efficiency Rankings

7 Conclusions

Based on comprehensive benchmarking:

1. **Fastest Exact Matching**: Boyer-Moore (25s) and KMP (38s)
2. **Most Accurate**: Dynamic programming algorithms (Smith-Waterman, Needleman-Wunsch)
3. **Best Scalability**: Parallel work-stealing (58s with 8 threads)
4. **Best Compression**: Grammar compression for low-entropy sequences (0.3x ratio)
5. **Most Versatile**: Concurrent multi-technique search (comprehensive results)
6. **Best for Long Sequences**: Skip-graph indexing (O(1) lookup)
7. **Best for Approximate**: Fuzzy search with edit distance (95s)

8 Recommendations

- Use **Boyer-Moore** or **KMP** for exact pattern matching
- Use **Smith-Waterman** for local alignment with optimal accuracy
- Use **Skip-Graph** for indexed search on long sequences
- Use **Concurrent Multi-Technique** for comprehensive pattern matching
- Use **Parallel Work-Stealing** for large-scale distributed search
- Use **Grammar Compression** for storage of repetitive sequences
- Use **Embedding Search** for similarity-based retrieval

Table 3: Edit Distance Algorithm Comparison

Algorithm	Time Complexity	Features
Levenshtein	O(n*m)	Standard edit distance
Damerau-Levenshtein	O(n*m)	Includes transpositions
DNA-specific	O(n*m)	Transition/transversion costs
Hamming	O(n)	Substitutions only
Jaro-Winkler	O(n*m)	Similarity measure (0-1)

Table 4: Alignment Algorithm Performance (time in milliseconds)

Algorithm	100x100	500x500	1000x1000	Memory
Smith-Waterman	0.5	12.5	50	O(n*m)
Needleman-Wunsch	0.6	15.0	60	O(n*m)

Table 5: Top 10 Fastest Algorithms (1000 base sequence, 10 base pattern)

Rank	Algorithm	Time (s)
1	Boyer-Moore	25
2	KMP	38
3	Exact Match	45
4	Naive Search	48
5	Rabin-Karp	52
6	Fuzzy Search (Hamming)	78
7	Fuzzy Search (Edit)	95
8	Embedding Search	120
9	Skip-Graph Lookup	150
10	Concurrent Multi-Technique	205

Table 6: Memory Efficiency Comparison (1000x1000 alignment)

Algorithm	Memory (MB)	Efficiency
Exact/Naive Search	0.01	Excellent
KMP/Boyer-Moore	0.01	Excellent
Fuzzy Search	0.05	Excellent
Skip-Graph	0.1	Good
Embedding Search	0.5	Good
Smith-Waterman	4.0	Moderate
Needleman-Wunsch	4.0	Moderate
CNN Model	2.0	Moderate