Report for the Workshop

Universidad Distrital Francisco José de Caldas

Student:

Alejandro mauricio junco Oviedo 20231020129

Faculty of Systems Engineering

Systems Analysis

Teacher: Carlos Andrés Sierra

15/09/2024

Systemic Analysis

1. General System Description

The developed system aims to manage a database of genetic sequences and provide tools to analyze and manipulate these sequences. The system is organized into four main components:

- **Controller**: This is the core of the system that orchestrates operations. It handles user interaction through a main menu and submenus and coordinates actions between the view and the model. It is responsible for loading sequences, generating new ones, filtering sequences by entropy, and searching for motifs.
- **Model (GeneticSequenceManager)**: This component manages the database of genetic sequences. It is responsible for generating random sequences, calculating the entropy of the sequences, filtering sequences with low entropy, and detecting patterns (motifs) in the sequences. It also handles data loading and storage.
- **View (VistaConsola)**: This provides a console-based user interface that allows users to interact with the system. It collects input from the user, displays menu options, and presents results and informational messages.
- **File Handler (Archivo)**: This handles data persistence. It is responsible for saving and loading the genetic sequences to and from a text file. It ensures that the directory and file are available for storing and retrieving data.

2. System Functionality

The system follows a structured and clear workflow:

- **Sequence Generation**: Users can generate a set of random genetic sequences with specified parameters (number of sequences and sequence length). The distribution of nucleotide bases is randomly and proportionally generated.
- **Data Loading**: The system allows users to load previously saved sequences from a file. This feature enables continued analysis without needing to regenerate sequences.
- **Entropy Filtering**: The system filters sequences based on their entropy, removing those that do not meet a specified entropy threshold. This helps reduce redundancy and keeps more informative sequences.
- **Motif Search**: The system detects patterns (motifs) in the sequences and provides information about the frequency of these patterns. It can identify the most frequent motif and analyze the repetition of patterns within the sequences.

3. Interaction Between Components

- The **Controller** acts as an intermediary between the **View** and the **Model**. It receives user input through the view, performs the necessary operations using the model, and then presents the results through the view.
- The **Model** carries out operations on the data and delegates file reading and writing tasks to the **File Handler (Archivo)**. The **View** interacts with the user and provides a simple interface for user interaction with the system.

- The **View** collects input from the user and passes it to the **Controller**. The **Controller** processes the input, performs the necessary actions through the **Model**, and updates the user with the corresponding information.

Complexity Analysis

The system includes several key operations that involve the manipulation, generation, and analysis of genetic sequences. Below is a breakdown of the computational complexity of these operations:

1. Sequence Generation

The system generates n sequences, each of length m. For each sequence, a loop iterates m times to assign a random nucleotide base (A, C, G, T) based on pre-generated probabilities. The total time complexity for sequence generation is:

- **Time Complexity**: O(n * m)
  For every sequence (n), we loop through each character (m) in the sequence. Random number generation and base assignment both occur in constant time O(1), so the overall complexity is linear with respect to both n and m.

2. Motif Detection

To detect motifs of length s in the sequences, the system examines all substrings of length s within each sequence. For each sequence of length m, the system checks m - s + 1 substrings. This process is repeated for all n sequences. Therefore, the time complexity for motif detection is:

- **Time Complexity**: O(n * (m - s + 1)) ≈ O(n * m)
  Since s is a small constant (bounded between 4 and 10), the motif detection complexity scales primarily with n and m.

3. Entropy Calculation

Entropy is calculated for each sequence by determining the frequency of each nucleotide base and computing the Shannon entropy formula. For a sequence of length m, this involves iterating over all characters to count base occurrences, followed by calculating the entropy. The time complexity of entropy calculation for one sequence is O(m), and for n sequences, it becomes:

- **Time Complexity**: O(n * m)
  Since each sequence requires processing all m characters and this process is repeated for n sequences, the complexity is linear with respect to both n and m.

4. Entropy-Based Filtering

Filtering sequences by entropy involves calculating the entropy for each sequence (O(m) per sequence) and then removing sequences that fall below a certain threshold. The filtering step itself requires a single pass over the sequences, resulting in a linear time complexity relative to the number of sequences:

- **Time Complexity**: O(n * m)
  The dominant operation is entropy calculation for each sequence, making the overall complexity linear in both n and m.

5. File I/O (Loading and Saving)

Both loading and saving sequences to a file involve reading or writing n sequences, each of length m. The time complexity for both loading and saving is:

- **Time Complexity**: O(n * m)
  The system reads or writes every character in the sequence, and for n sequences, this results in a linear complexity with respect to the total amount of data.

Summary of Key Operations:

- **Sequence Generation**: O(n * m)
- **Motif Detection**: O(n * m)
- **Entropy Calculation**: O(n * m)
- **Entropy Filtering**: O(n * m)
- **File I/O**: O(n * m)

Overall Complexity:

The system's performance is largely dominated by operations that scale linearly with the number of sequences (n) and the length of each sequence (m). The most computationally expensive tasks, such as motif detection and entropy calculation, all have similar time complexities, O(n * m). Given that n can be as large as 2,000,000 and m can reach 100, it is crucial to optimize the implementation to handle large datasets efficiently.

Chaos Analysis

In this system, entropy plays a central role in analyzing the level of chaos and randomness in the genetic sequences. Entropy, particularly **Shannon entropy**, is a measure used to quantify the uncertainty or unpredictability within a system. In the context of this system, the randomness and variability of nucleotide sequences are evaluated through entropy calculations. Below is an analysis of how chaos is measured and managed:

1. Definition of Chaos (Entropy)

In the system, chaos refers to the diversity of nucleotide sequences. A highly diverse sequence, where the bases (A, C, G, T) appear with relatively equal probabilities, will have high entropy, indicating a high level of chaos. Conversely, sequences where one or more bases dominate will have lower entropy, reflecting less randomness and more order.

The **Shannon entropy formula** used in this system is as follows:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P\,(x_i)$$

Where:

- H(X) is the entropy of the sequence,
- $P(x_i)$ is the probability of occurrence of base x

- n is the number of distinct bases (A, C, G, T).

2. Entropy-Based Filtering

One of the primary mechanisms for managing chaos in the system is through entropy-based filtering. By setting an entropy threshold, the system can filter out sequences with low entropy, thereby ensuring that only sequences with a higher level of randomness and diversity remain. This process helps focus the analysis on more chaotic sequences, which are often more interesting for genetic studies or pattern detection.

- **High Entropy Sequences**: These sequences exhibit a balance in the occurrence of nucleotide bases (A, C, G, T). Such sequences are considered chaotic because the probability of predicting the next base is low due to their random nature. High entropy values typically approach the maximum (approximately 2 for four bases).
- **Low Entropy Sequences**: These sequences are more ordered, often dominated by one or two bases. For example, a sequence like "AAAAA" has an entropy of 0, as it is highly predictable. Low entropy values indicate less chaos and higher predictability.

3. Importance of Chaos in Motif Detection

The presence of chaos (or the lack of it) significantly affects motif detection. In sequences with high entropy, motifs may be harder to detect due to the random nature of the sequences. However, these motifs, if found, may be more meaningful since they occur within a highly chaotic environment.

Conversely, in low-entropy sequences, motifs are often easier to detect due to the repetitive nature of the sequences. However, these motifs may be less interesting or biologically relevant because they arise in overly predictable, less chaotic environments.

- **High-Entropy Sequences**: Finding motifs in these sequences may require more advanced algorithms, but the motifs are likely to carry more significance.
- **Low-Entropy Sequences**: Motifs are easier to detect, but they are often artifacts of the ordered structure rather than biologically meaningful patterns.

4. Chaos Experimentation

The system allows for experimentation with different levels of entropy by generating sequences with varying probabilities of nucleotide occurrence. By adjusting these probabilities, the system can create sequences ranging from highly ordered (low entropy) to highly chaotic (high entropy).

- **Experimentation**: By generating datasets with different levels of randomness and then applying entropy-based filtering, the user can observe the impact of chaos on motif detection and overall data structure. The results can be summarized in tables that track the impact of entropy filtering on motif detection, sequence diversity, and computational efficiency.

5. Managing Chaos in Large Datasets

Given the potential size of the dataset (up to 2,000,000 sequences), managing chaos becomes critical. Sequences with low entropy may inflate the dataset without adding significant analytical value. By filtering out low-entropy sequences, the system can focus on more diverse, informative data, leading to more meaningful results in motif detection and a more efficient analysis process

Conclusion on Chaos

The system's ability to measure, filter, and experiment with entropy levels allows for a robust analysis of chaos in genetic sequences. By balancing the level of chaos through entropy thresholds, the system ensures that the most relevant and unpredictable sequences are retained for further analysis, while low-entropy, redundant sequences are discarded. This approach helps optimize both the data quality and the computational efficiency of the system.

Results

Several experiments were conducted to evaluate the system's ability to generate, analyze, and filter genetic sequences based on entropy and motif detection. The main aspects considered were the generation of sequences, entropy filtering, and motif detection.

1. Sequence Generation and Entropy Calculation

Different datasets were generated with various parameters for sequence length (m) and number of sequences (n). The entropy of each sequence was calculated to assess the diversity and randomness of the datasets. The following table summarizes the results:

Dataset	Number of Sequences (n)	Sequence Length (m)	Average Entropy
1	1000	10	1.92
2	5000	20	1.88
3	10000	50	1.85
4	50000	100	1.78
5	100000	75	1.81

The average entropy across different datasets remained relatively stable, with slight variations depending on sequence length and the randomness of nucleotide base generation.

2. Motif Detection

Motif detection experiments were performed on the datasets to identify common patterns of various lengths (s). The following table summarizes the most frequent motifs detected:

Dataset	Motif Length (s)	Most Frequent Motif	Occurrences
1	4	"ATGC"	15
2	6	"CGTACC"	22
3	5	"GATCG"	31
4	7	"ATGCGTG"	9
5	6	"GCGTAA"	13

Larger motifs were detected less frequently due to the inherent randomness of the sequences, but the system successfully identified motifs across all datasets.

3. Entropy-Based Filtering

Entropy-based filtering was applied to remove low-entropy sequences, enhancing the randomness of the remaining data. The table below shows the results after applying different entropy thresholds:

Dataset	Entropy Threshold	Sequences Removed	Remaining Sequences	Average Entropy (Post-Filter)
1	1.5	120	880	1.93
2	1.7	320	4680	1.89
3	1.6	850	9150	1.87
4	1.8	5000	45000	1.82
5	1.7	9000	91000	1.83

The filtering process increased the average entropy of the remaining sequences, indicating a higher level of chaos and diversity in the dataset.

4. Impact of Entropy Filtering on Motif Detection

The table below summarizes how entropy filtering affected motif detection:

Dataset	Motif Length (s)	Most Frequent Motif (Pre-Filter)	Occurrences (Pre-Filter)	Most Frequent Motif (Post-Filter)	Occurrences (Post-Filter)
1	4	"ATGC"	15	"TGCA"	8
2	6	"CGTACC"	22	"GCTAGA"	17
3	5	"GATCG"	31	"TCGGA"	28
4	7	"ATGCGTG"	9	"GCTAGTA"	5
5	6	"GCGTAA"	13	"ATGGCA"	10

The most frequent motifs sometimes shifted after filtering, indicating that the entropy-based filter removed more predictable sequences, thus changing the motif landscape in the remaining sequences.

Discussion of Results

The experiments demonstrated that the system successfully generates and analyzes genetic sequences, as well as effectively filters them based on entropy. The entropy values across datasets remained consistent, with minor variations due to the random generation process. Higher sequence lengths resulted in slightly lower entropy values due to the greater chance of repetitive patterns.

Motif detection results showed that smaller motifs were more frequent and easier to detect, while larger motifs occurred less often due to the increasing randomness in longer sequences. The system performed well in detecting motifs across different datasets and motif lengths.

Entropy-based filtering had a significant impact on the remaining data. By removing low-entropy sequences, the system retained sequences with greater randomness and diversity, as reflected in the increased average entropy post-filtering. Additionally, motif detection results after filtering showed that the most frequent motifs often changed, highlighting the impact of removing predictable sequences on pattern detection.

Overall, the system’s ability to dynamically filter sequences based on entropy allows for more refined analysis, particularly in the detection of motifs within chaotic datasets. This process also reduces redundancy in the data, improving the quality of the analysis.

Conclusions

The system developed for genetic sequence generation, entropy filtering, and motif detection is robust and effective in managing large datasets. The ability to generate random sequences and analyze them through entropy measures allows for a comprehensive understanding of the chaos and diversity present in genetic data.

Entropy filtering proved to be an essential tool in refining the datasets by removing sequences with low randomness. This filtering not only enhanced the quality of the remaining data but also influenced the results of motif detection, where new patterns emerged after low-entropy sequences were removed.

The system's flexibility in handling various sequence lengths, motif sizes, and entropy thresholds makes it a valuable tool for bioinformatics applications, where identifying meaningful patterns within large, complex datasets is critical. Future improvements could focus on optimizing motif detection for larger datasets and further refining the entropy threshold for more granular control over the filtering process.