Санкт-Петербургский государственный университет

Попов Владимир Витальевич

Выпускная квалификационная работа

Скрытые марковские модели на графах сборки

Уровень образования: бакалавриат

Направление 01.03.02 «Прикладная математика и информатика»

Основная образовательная программа СВ.5004.2017 «Прикладная математика и информатика»

Профиль «Вычислительная стохастика и статистические модели»

Научный руководитель:

Доцент, кафедра статистического моделирования

к. ф.-м. н., доцент А. И. Коробейников

Рецензент:

Старший научный сотрудник

к. ф.-м. н. А. Д. Пржибельский

Санкт-Петербург

2021

Saint Petersburg State University
Applied Mathematics and Computer Science
Computational Stochastics and Statistical Models

Popov Vladimir

Graduation Project

Hidden markov models on assembly graph

Scientific Supervisor:
Associate Professor, Department of
Statistical Modelling
PhD A. I. Korobeynikov

Reviewer:
Senior research fellow
PhD A. D. Prjibelski

Saint Petersburg
2021

Оглавление

Введение

В современной биоинформатике одной из ключевых задач является восстановление первичной последовательности генома и определение его функций, то есть сборка и аннотация генома. Аннотация — это выявление биологической информации из генома, а именно функций генов и свойств их взаимодействий. Наиболее важным этапом аннотации является поиск (предсказание) генов, определение кодирующих последовательностей в геноме. Именно этой задаче и будет посвящено основное внимание в данной работе. На данный момент для решения задачи предсказания генов разработано множество программ, например: Glimmer [1], MetaGene [2], FragGeneScan [3], Prodigal [4], GeneMark [5]. Однако большинство из них принимают на вход полные геномные последовательности, что является значительным препятствием, так как зачастую сам геном неизвестен, дано лишь множество его коротких фрагментов — ридов. Для того, чтобы получить полную геномную последовательность из множества ридов, необходимо решить сложную вычислительную задачу сборки генома. Возникает необходимость в методе решения задачи предсказания генов для случая, когда дано множество ридов, не выполняя предварительной сборки генома. Такую задачу можно эффективно решить, используя представление множества ридов в виде графа.

В данной работе будет изучен метод FragGeneScan, основанный на скрытых марковских моделях, а также реализована его модификация, позволяющая решать задачу предсказания генов на графах.

Глава 1

Основные определения

Определение 1.1. *Геном — строка над алфавитом* $Q = \{ A, C, G, T\}$.

Определение 1.2. *Кодон — тройка символов в геноме (нуклеотидных остатков), единица генетического кода.*

Определение 1.3. *Старт-кодон — кодон, кодирующий начало трансляции белка в рибосоме.*

Определение 1.4. *Стоп-кодон — кодон, который кодирует прекращение трансляции.*

Типичные стоп-кодоны в ДНК — это TAA, TAG, TGA. Наиболее распространённым старт-кодоном является ATG, однако встречаются и альтернативные. Последние чаще встречаются в ДНК прокариотов, в основном это GTG и TTG.

Определение 1.5. *Кодирующая последовательность — подстрока в геноме неизвестной длины, начинающаяся со старт-кодона и закачивающаяся стоп-кодоном.*

Формальное определение кодирующей последовательности — это сложная задача, и на данный момент не существует точного общепринятого определения. Можно рассмотреть геном S как реализацию случайного процесса. Кодирующая последовательность в такой интерпретации — участок генома, конечномерное распределение символов в котором отличается от конечномерных распределениях в соседних участках.
Пусть дана строка S длины $L > 0$, состоящая из букв алфавита $Q = \{ A, C, G, T\}$ (нуклеотидных оснований).

Определение 1.6. *Решением задачи предсказания генов для строки* S *называется множество подпоследовательностей* $\Omega = \{\omega = s_i s_{i+1} \dots s_j \mid i, j \in \mathbb{N},\ 1 \leqslant i < j \leqslant L\}$, *такое, что* $\omega \in \Omega$ *— максимальный по включению кодирующий участок.*

Предсказание генов — это задача идентификации кодирующих участков ДНК, что является одной из наиболее важных и сложных задач во всех проектах секвенирования генома. Введём аппарат скрытых марковских моделей как способ формализации определения кодирующей последовательности и эффективного инструмента поиска в геноме участков с другим распределением.

Глава 2

Скрытые марковские модели

2.1. Определение СММ

Определение 2.1. *Пусть X_t, Y_t — случайные процессы с конечными множествами значений $X = \{x_1, \ldots, x_n\}$ и $Q = \{q_1, \ldots, q_k\}$ соответственно. Скрытая марковская модель (СММ) с дискретным временем — пара случайных процессов (X_t, Y_t), такая, что:*

- X_t — *однородная марковская цепь.*
- $\mathsf{P}\big(Y_j \,\big|\, X_1, \ldots, X_j, Y_1, \ldots, Y_j\big) = \mathsf{P}\big(Y_j \,\big|\, X_j\big)$, $\forall\, j \geqslant 1$.

Тем самым распределение СММ определяется следующими компонентами:

- P — матрица переходных вероятностей марковской цепи X_t.
- p — вектор начального распределения марковской цепи X_t.
- B — матрица вероятностей, где $b_{ij} = \mathsf{P}\big(q_j \,|\, x_i\big)$ $\forall\, i \in 1:n$, $j \in 1:k$, — вероятность получить событие q_j при состоянии x_i.

Пусть задана скрытая марковская модель H с параметрами P, p, B. Для удобства обозначим их $\lambda := (P, p, B)$. Реализацию СММ можно представить как две последовательности: $(a_1, \ldots, a_L)$, $(\pi_1, \ldots, \pi_L)$, где $a_i \in X$, $\pi_j \in Q$. Будем называть последовательность $\mathbf{a} = a_1 a_2 \ldots a_L$ *путём через лежащую в основе марковскую цепь* или просто *путём*, а последовательность $\pi = \pi_1 \pi_2 \ldots \pi_L$ — *последовательностью наблюдений*. Определим функцию правдоподобия $\mathsf{P}(\mathbf{a}, \pi|\lambda)$ реализации $(\mathbf{a}, \pi)$ моделью H:

$$\mathsf{P}(\mathbf{a}, \pi|\lambda) = p_{a_1} b_{a_1}(\pi_1) p_{a_1 a_2} b_{a_2}(\pi_2) \times \cdots \times p_{a_{L-1} a_L} b_{a_L}(\pi_L).$$

Также определим вероятность $\mathsf{P}(\pi|\lambda)$ для последовательности наблюдений π:

$$\mathsf{P}(\pi|\lambda) = \max_{\mathbf{a}:|\mathbf{a}|=L} \mathsf{P}(\mathbf{a}, \pi|\lambda),$$

где максимум берется по всевозможным последовательностям $\mathbf{a}$ длины L.

Ключевую роль при работе с СММ имеет задача нахождения наиболее вероятного пути при известной последовательности наблюдений. Рассмотрим алгоритм, который решает такую задачу.

2.2. Алгоритм Витерби

Пусть дана СММ H с параметрами $\lambda = (P,\, B,\, p)$, множество $X = \{x_1, \ldots,\, x_n\}$ и алфавит $Q = \{q_1,\, \ldots,\, q_m\}$, а также строка наблюдений $\pi = S = s_1, \ldots, s_L,\ s_i \in Q$, $i \in 1 : L,\ L > 0$. Необходимо найти $\mathbf{a}^* = \operatorname{argmax}_{\mathbf{a}:|\mathbf{a}|=L} \mathsf{P}(\mathbf{a}, S|\lambda)$. Задачу решает алгоритм Витерби [6]. Определим

$$v_k(1) = p_k b_k(s_1) \quad \forall\, k \in 1 : n, \tag{2.1}$$

$$v_k(i) = \max_{u_1,\ldots,u_{i-1} \in X} p_{u_1} b_{u_1}(s_1) p_{u_1\, u_2} b_{u_2}(s_2)\ \times \ldots$$
$$\times\, p_{u_{i-2}\, u_{i-1}} b_{u_{i-1}}(s_{i-1}) p_{u_{i-1}\, a_k} b_k(s_i) \quad \forall\, i \in 2 : L,\ k \in 1 : n.$$

Получается, что

$$v_k(i+1) = b_k(s_{i+1}) \max_{l=1,\ldots,n} (v_l(i) p_{l\,k}) \quad \forall\, i \in 1 : L-1,\ k \in 1 : n. \tag{2.2}$$

Таким образом, по начальным условиям (2.1) и рекуррентному соотношению (2.2) могут быть вычислены $v_k(i)$ для $i \in 2 : L$ и $k \in 1 : n$. Будем называть получившуюся матрицу v размера $n \times L$ матрицей Витерби.

На каждом шаге рекурсии формируем множество

$$\gamma_k(i) = \{m \in \mathbb{N} : v_m(i) p_{m\,k} = \max_{l=1,\ldots,n} (v_l(i) p_{lk})\},\ i \in 1 : L,\ k \in 1 : n.$$

Получаем

$$\max_{\mathbf{a}:|\mathbf{a}|=L} \mathsf{P}(\mathbf{a},\, S|\lambda) = \max_{l=1,\ldots,n} v_l(L).$$

Все последовательности $\mathbf{a}^*$ (если их несколько) могут быть восстановлены следующей процедурой: выбираем $m_L \in \gamma(L)$, затем выбираем $m_{L-1} \in \gamma_{m_L}(L-1)$, затем $m_{L-2} \in \gamma_{m_{L-1}}(L-2)$, и продолжаем, пока не выберем $m_1 \in \gamma_{m_2}(1)$. Для найденной последовательности $\mathbf{a}^* = x_{m_1} x_{m_2} \ldots x_{m_L}$ выполнено:

$$\mathsf{P}(\mathbf{a}^*,\, S|\lambda) = \max_{\mathbf{a}:|\mathbf{a}|=L} \mathsf{P}(\mathbf{a}, S|\lambda).$$

Теперь можем перейти к модели интересующего нас метода.

Глава 3

Метод FragGeneScan

3.1. Свойства и особенности алгоритма

FragGeneScan — метод прогнозирования генов, основанный на использовании скрытых марковских моделей. Эффективность метода сравнима с GLIMMER и MetaGene для полных геномов и превосходит их для ридов — небольших участков генома [3]. Одна из важных особенностей метода — это корректная обработка сдвигов рамки считывания, вызванных ошибками в прочтении. FragGeneScan может предсказывать как полные гены, так и частичные гены без старт или стоп кодона.

Введем несколько важных для понимания структуры модели определений. Дана строка $S = s_1, \ldots, s_n$ длины $n > 0$, $s_i \in Q = \{A, C, G, T\}$. Введем отображение $f : Q \to Q$, действующее по формулам : $f(A) = T,\ f(G) = C,\ f(T) = A,\ f(C) = G$.

Определение 3.1. *Комплементарной строкой для строки S называется строка $S' = s'_1, \ldots, s'_n$, такая, что $s'_i = f(s_i) \quad \forall\, i \in 1 : n$.*

Определение 3.2. *Реверс-комплементарной строкой для строки S называется строка $S' = s'_1, \ldots, s'_n$, такая, что $s'_i = f(s_{L-i+1}) \quad \forall\, i \in 1 : n$.*

ДНК состоит из двух цепей, ориентированных азотистыми основаниями нуклеотидов друг к другу. Азотистые основания одной цепи соединены с основаниями другой водородными связями по принципу комплементарности: аденин (A) соединяется только с тимином (T), а гуанин (G) — с цитозином (C). Поскольку принадлежность последовательности конкретной цепи не определена, можно решить задачу предсказания генов для прямой и реверс-комплементарной нуклеотидной последовательности по отдельности. Однако, FragGeneScan решает эту проблему другим способом, осуществляя одновременный поиск генов в обоих цепях.

3.2. Скрытая марковская модель FragGeneScan

Модель FragGeneScan представляет собой СММ, причём $Q = \{ A, C, G, T, \varnothing \}$,

$$X = \big\{E^+, E^-, S^-, S^+, R, I_1, \dots, I_6, M_1, \dots, M_6, D_1, \dots, D_6, I_1^-, \dots, I_6^-, M_1^-, \dots, M_6^-, D_1^-, \dots, D_6^-\big\}.$$

Состояния поделены на две группы, представляющие области генов в прямой и реверс-комплементарной нуклеотидной последовательности. Состояния с отметкой «$-$» — это состояния, относящиеся к реверс-комплементарной последовательности, остальные состояния, кроме R, относятся к прямой последовательности. E — состояние, соответствующее стоп-кодону, S — старт-кодону, R — не кодирующим участкам. Кодирующим участкам отвечают 3 типа состояний: M — состояние совпадения, I — состояние вставки, D — состояние удаления. Распределение марковской цепи X_t можно представить в виде семи блоков (Рис 3.1, затемнённые поля), представляющие участки кодирующие гены (i,vii), старт-кодоны (ii,v) и стоп-кодоны (iii,vi) для прямой (i-iii), реверс-комплементарной (v-vii) последовательности и не кодирующих участков (iv). Выделим из X подмножества Ω и Ω^-, представляющих участки цепи, кодирующие гены.

$$\Omega = \big\{I_1, \dots, I_6, M_1, \dots, M_6, D_1, \dots, D_6\big\},$$
$$\Omega^- = \big\{I_1^-, \dots, I_6^-, M_1^-, \dots, M_6^-, D_1^-, \dots, D_6^-\big\}.$$

Состояния из Ω и Ω^- используют марковскую модель *второго порядка*. Это означает, что

$$\mathsf{P}\big(Y_j \,\big|\, X_i, Y_i, 0 \leqslant i \leqslant j\big) = \mathsf{P}\big(Y_j \,\big|\, X_j, X_{j-1}\big), \quad \forall\, j \geqslant 1.$$

Это обстоятельство делает невозможным использование алгоритма Витерби, поэтому преобразуем множество состояний так, чтобы модель стала привычной МЦ (первого порядка). Для этого достаточно в множестве состояний X заменить Ω и Ω^- на множества Θ и Θ^- соответственно, причём

$$\Theta = \big\{ZZ' \,\big|\, Z, Z' \in \Omega, p_{ZZ'} > 0\big\},\ \Theta^- = \big\{ZZ' \,\big|\, Z, Z' \in \Omega^-, p_{ZZ'} > 0\big\},$$

где p_{ij} — элемент матрицы переходных вероятностей X.

Обозначим $\widetilde{X} = X \cup \Theta \cup \Theta^- \setminus (\Omega \cup \Omega^-)$ — расширенное множество состояний. Введём ещё одно важное понятие.

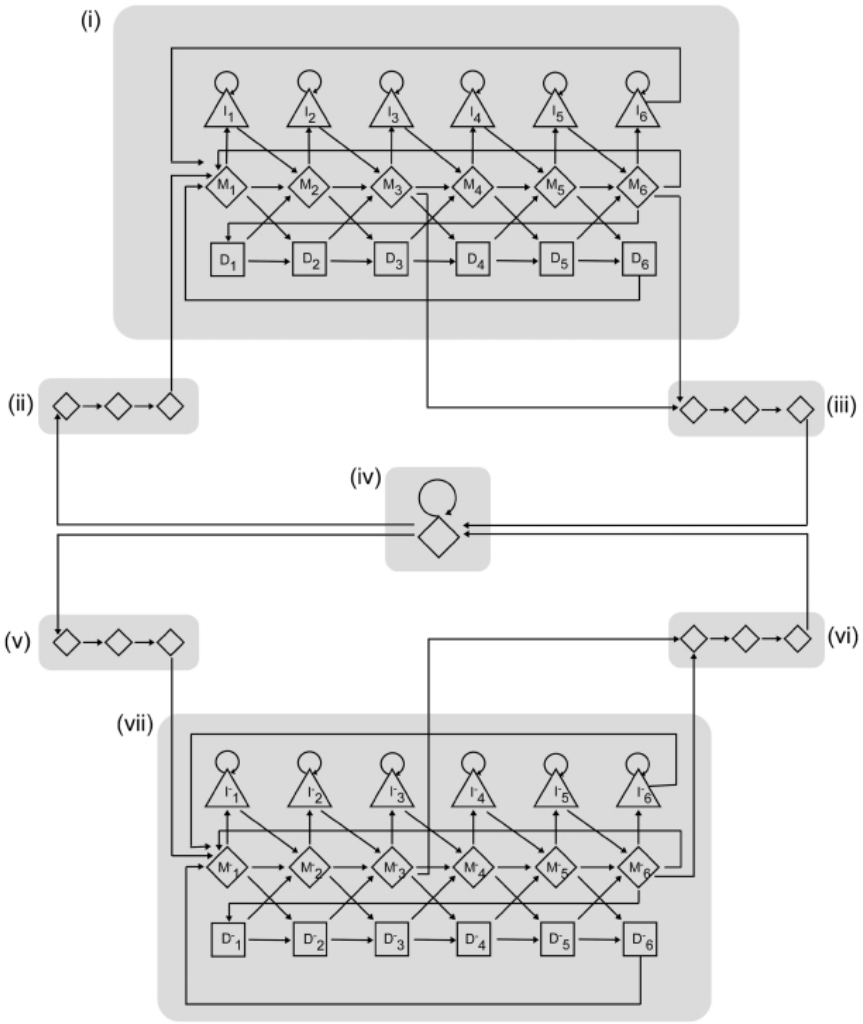


Рис. 3.1. Распределение марковской цепи модели FragGeneScan.

Определение 3.3. *GC-состав для строки S равен доле гуанина(G) и цитозина(C) среди всех символов нуклеотидной последовательности S.*

Возьмём реализацию СММ модели FragGeneScan — строку $S = s_1 s_2 \dots s_N$, $s_i \in Q \ \forall \ i$ и рассмотрим распределение $\mathcal{P}$ отдельного символа s_i. Число состояний конечно, и каждому символу алфавита Q можно сопоставить вероятность — p_A, p_C, p_G, p_T для символов A, C, G, T соответственно. Для этих значений должны быть выполнены соотношения: $p_A + p_C + p_G + p_T = 1$, а также $p_C = p_G$, $p_A = p_T$ (так как GC-состав одинаков для строки S и её реверс-комплементарной строки S'). Таким образом, все p_i могут быть определены, если известен GC-состав $p_{GC} = p_G + p_C$. Рассмотрим элементы матрицы вероятностей B, где $b_{ij} = \mathsf{P}\big(q_j \,|\, x_i\big)$. Поскольку p_{GC} параметризует распределение $\mathcal{P}$, то $\mathsf{P}(s_j \mid x_i) = f_j(x_j, p_{GC})$, где f_j — некоторая функция. Поэтому при известных f_j элементы матрицы B параметризуются GC-составом. В модели FragGeneScan эти f_j

известны, причем есть несколько наборов параметров в завимости от предполагаемой частоты ошибок секвенирования: для 0.5, 0, 1 и 3 процентов ошибок.

3.3. Решение задачи предсказания генов при помощи модели FragGeneScan

Итак, в модели FragGeneScan определено множество состояний $\widetilde{X}$, алфавит Q, а также предварительно оценены элементы матрицы переходных вероятностей МЦ $\widetilde{X}$, а также известны функции $f_j(x_j, p_{GC}), \forall\, j \in 1 : |\widetilde{X}|$. Пусть дана строка $S = s_1, \ldots, s_L$ длины $L > 0$, $s_i \in Q = \{ A, C, G, T \}$. В терминах скрытой марковской модели S — последовательность наблюдений. Посчитаем GC-состав строки S для полного определения параметров модели. Таким образом, для строки S можно применить алгоритм Витерби и найти $\mathbf{x}^* = x_1 \ldots x_L$ — наиболее вероятную последовательность состояний $x_i \in \widetilde{X}$, $i \in 1 : L$. При фиксированном j подпоследовательность $s_j s_{j+1} \ldots s_{j+N}$ будет отмечена методом FragGeneScan как кодирующая, если выполнены четыре следующих условия:

1. $N \geqslant 119$ (длина гена больше 120 нуклеотидов),
2. $x_j = z$, где $z \in \{S, S^-, M_1, \ldots, M_6, M_1^-, \ldots, M_6^-\}$ — ген начинается со старт-кодона или состояния совпадения,
3. $x_k \neq z \quad \forall\ k \in j+1 : j+N-1$, где $z \in \{S, S^-, E, E^-\}$, — старт-кодоны и стоп-кодоны допускаются только на краях строк,
4. $x_{j+N} = z$, где $z \in \{E, E^-, M_1, \ldots, M_6, M_1^-, \ldots, M_6^-\}$ — ген заканчивается стоп-кодоном или состоянием совпадения.

Опишем алгоритм, который позволяет найти соответствующие последовательности.

Алгоритм поиска кодирующих последовательностей в выравнивании. Для простоты будем рассматривать алгоритм только прямой цепи, для реверс-комплиментарной алгоритм аналогичен. Запишем алгоритм в виде псевдокода. В результате его выполнения получим множество подпоследовательностей Ω — решение задачи предсказания генов для строки S.

Data: S, x, N. (строка, оптимальный путь и их длина)

Result: Ω — список кодирующих последовательностей

Initialization: $codonstart \leftarrow 0$; // индикатор кодирующей последовательности

$dlen$; // её длина

$dnastart \leftarrow 0$; // позиция начала

$dnaend \leftarrow 0$; // позиция конца

$start \leftarrow -1$; // вспомогательная переменная

$t \leftarrow 0$;

for $t < N$ **do**

- **if** $codonstart = 0$ *and* $start < 0$ *and* $(x[t] = M_i$ *or* $x[t] = S)$ **then**
  - $dnastart \leftarrow start \leftarrow t + 1$;
- **if** $codonstart = 0$ *and* $start\_t < 0$ *and* $(x[t] = M_1$ *or* $x[t] = M_4)$ **then**
  - $dlen \leftarrow 0$;
  - $prevmatch \leftarrow x[t]$;
  - $codonstart \leftarrow 1$; // начался потенциальный ген
- **else if** $codonstart! = 0$ *and* $(x[t] = E$ *or* $t = N - 1)$ **then**
  - **if** $x[t] = E$ **then**
    - $dnaend \leftarrow t + 3$;
  - **else**
    - $temp \leftarrow t$;
    - **while** $x[temp]! = M1$ *and* $x[temp]! = M4$ **do**
      - $temp \leftarrow temp - 1$;
    - $dnaend \leftarrow temp$;
  - **if** $dlen > 120$ **then**
    - **if** $start = dnastart - 3$ **then**
      - $dnastart \leftarrow start$;
    - $push(\Omega, s_{dnastart} s_{dnastart+1} \dots s_{dnaend})$
- **else if** $codonstart! = 0$ *and* $x[t] = M_i$ **then**
  - **if** $x[t] < prevmatch$ **then**
    - $dlen \leftarrow dlen + x[t] + 6 - prevmatch$
  - **else**
    - $dlen \leftarrow dlen + x[t] - prevmatch$
  - $prevmatch \leftarrow x[t]$
- **else if** $codonstart! = 0$ *and* $x[t] = R$ **then**
  - $dlen \leftarrow codonstart \leftarrow 0$;
  - $end \leftarrow start \leftarrow -1$;

$t \leftarrow t + 1$;

Алгоритм 1: поиск кодирующих последовательностей

Глава 4

Задача предсказания генов на графах

4.1. Задача предсказания генов на ориентированных ациклических графах

Модель FragGeneScan позволяет решать задачу предсказания генов для целой нуклеотидной последовательности (генома). Однако, поскольку не существует методов, позволяющих полностью считать геном, обычно геном наблюдается в виде множества коротких прочтений (ридов). Это множество можно представить в виде ориентированного графа. Для этого существует несколько подходов, например, overlap layout consensus, заключающийся в том, что вершины — это риды, а ребра проведены только между теми ридами, которые перекрываются (имеют overlap). Другой способ представления — это граф де Брёйна, в нем риды лежат на ребрах. Такой граф является ориентированным, и строится так, чтобы в нём содержался эйлеров цикл (замкнутый путь, проходящий через каждое ребро ровно по одному разу). Для нахождения возможного генома в графе де Брёйна нужно найти эйлеров путь в подграфе. Такой метод является удобным способом представления множества ридов, и он часто применяется в современных методах сборки генома. В таких графах присутствуют циклы, но для начала необходимо научиться решать поставленную задачу на ациклических графах.

Пусть геном наблюдается в виде некоторого ориентированного ациклического графа, пример которого представлен на Рис 4.1. Каждому ребру сопоставлена некоторая строка S над алфавитом $Q = \{ A, C, G, T\}$.

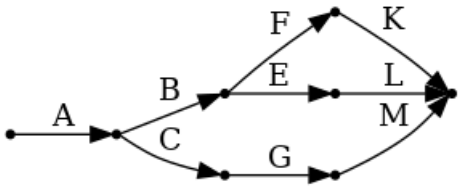


Рис. 4.1. Ориентированный ациклический граф. $A, B, C, D, E, F, G, K, L, M$ — строки над Q.

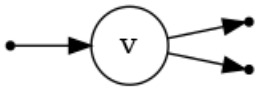
Сам геном может быть получен некоторым неизвестным заранее обходом графа. Необходимо решить задачу предсказания генов на ациклическом графе. Это можно сде-

лать напрямую, перечислив все пути и отыскав фрагменты генома, а затем решить задачу для строк известным алгоритмом. Однако, такая реализация будет вычислительно затратной, сложность растёт экспоненциально с увеличением числа рёбер. Однако, как можно заметить на примере графа на Рис 4.1, возможен иной способ решения такой задачи. Рассмотрим два пути на графе: строки $ABEL$ и $ABFK$. Эти строки имеют общее начало — подстроку AB. Поскольку в алгоритме Витерби для подсчёта каждого следующего столбца необходима информация только о предыдущем столбце, то можно посчитать матрицу Витерби для подстроки AB, а затем использовать информацию о последнем столбце при счёте матрицы для подстрок EL и FK. Именно такой подход и будет применён в следующем разделе.

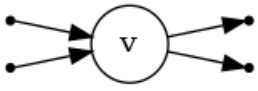
4.2. Алгоритм Витерби на графах (прямой ход)

Ключевую роль в решении задачи предсказания генов методом FragGeneScan играет алгоритм Витерби, поэтому для решения задачи предсказания генов на графах нужно обобщить соответствующий алгоритм. Каждую вершину (кроме начальной и конечной) в ориентированном ацикличном графе можно отнести к одному из двух типов:

1. Развилка — вершина с единичной степенью захода, а степенью исхода равной 2 или больше.



2. Соединение — вершина с произвольной степенью исхода, а степенью захода равной 2 или больше.



Целью алгоритма является получить такие матрицы, которые были бы получены при обходе целой строки, словно разрыва в вершинах не было. В стандатном алгоритме при построении каждого следующего столбца матрицы Витерби необходима информация только о предыдущем столбце, но в случае ребра, выходящего из вершины типа «соединение» возникает неоднозначность, так как входящих ребер больше двух. Простым способом преодолеть эту трудность является взятие поэлементного максимума, и

поскольку в алгоритме необходимо максимизировать вероятности, то этот способ является оправданным и логичным.

Также стоит обратить внимание на порядок обхода графа. Поскольку столбцы матрицы Витерби строятся последовательно друг за другом, то нужно обрабатывать ребро, выходящее из некоторой вершины только после того, как обработаны все входящие в него ребра. Таким образом, получаем алгоритм обхода и обработки, формализуем его.

Дано:

- Связный ориентированный ацикличный граф $D = (V,\ E)$. Он состоит из множества V, элементы которого называются вершинами, и множества упорядоченных пар $(u, v),\ \ u, v \in V$ называемых рёбрами. Каждому ребру графа сопоставлена некоторая строка над алфавитом $Q = \{\, A,\, C,\, G,\, T\}$. Пусть также дана скрытая марковская модель FragGeneScan с множеством состояний $\widetilde{X}$ (структура модели представлена на Рис 3.1). Поскольку СММ параметризуется GC-составом, определим GC-состав для графа как сумму числа символов G и C во всех строках, деленную на сумму длин строк. Обозначим параметры модели $\lambda = (P, B, p)$.

- Для некоторой вершины $v \in V$ обозначим $v^{\rightarrow}$ — список вершин, в которые ведёт ребро из вершины v, т.е $u \in v^{\rightarrow} \Leftrightarrow \exists e \in E : e = (v, u)$. Будем называть $v^{\rightarrow}$ списком выходящих рёбер, а размер этого списка $|v^{\rightarrow}|$ — степенью исхода.

- Аналогично определим список входящих рёбер $v^{\leftarrow}$ для вершины $v \in V$ — список вершин, из которых ведёт ребро в вершину v, т.е $u \in v^{\leftarrow} \Leftrightarrow \exists e \in E : e = (u, v)$. Назовём $|v^{\leftarrow}|$ степенью захода.

- Известна вершина $v_1 \in V : |v_1^{\leftarrow}| = 0$ — начальная вершина с нулевой степенью захода.

Необходимо найти: Матрицу Витерби α для каждого ребра из множества E.

Алгоритм:

1. Упорядочим вершины графа, используя топологическую сортировку, поместим результаты в очередь из вершин U. Получена очередь в которой каждая вершина расположена до всех своих потомков (и после всех своих предков).

2. Пока U не пуста выполняем:

   а. Вынимаем из очереди U вершину v.

   б. Для каждой строки, лежащей на ребрах $e = (v, u), u \in v^{\rightarrow}$ формируем матрицу Витерби по следующему алгоритму. Рассмотрим 2 случая, которые отличаются лишь в построении первого столбца матрицы.

   - Вершина v относится к первому типу. Первый столбец матрицы Витерби v для текущей строки $S = s_1 s_2 \dots s_L$ получаем используя последний столбец σ матрицы Витерби для строки-предка:

$$\alpha_{k1} = b_k(s_1) \max_{l=1\dots|\widetilde{X}|} (\sigma_l p_{lk}) \quad \forall\, k \in 1 : |\widetilde{X}|.$$

   - Вершина v относится ко второму типу. Без потери общности можно предположить, что входящая степень вершины равна двум, а последние столбцы матриц Витерби для строк предков равны σ^1 и σ^2. Получаем первый столбец матрицы Витерби α для текущей строки $S = s_1 s_2 \dots s_L$:

$$\alpha_{k1} = b_k(s_1) \max_{l=1\dots|\widetilde{X}|} (\max(\sigma_l^1, \sigma_l^2) p_{lk}) \quad \forall\, k \in 1 : |\widetilde{X}|.$$

   в. Далее для каждого случая матрица формируется аналогично классическому алгоритму:

$$\alpha_{k,i+1} = b_k(s_{i+1}) \max_{l=1,\dots,n} (\alpha_{li}\, p_{lk}) \quad \forall\, i \in 1 : L-1,\ k \in 1 : |\widetilde{X}|.$$

Так, получаем матрицу Витерби для каждого ребра ациклического ориентированного графа. Для получения матрицы Витерби некоторого пути на графе достаточно объединить соответствующие ребрам пути матрицы в одну матрицу. Опишем подробнее, как именно это осуществить для графа.

4.3. Алгоритм Витерби на графах (обратный ход)

При выполнении обратного хода нужно сделать в точности такой же обход, что и в прямом ходе алгоритма, но в противоположную сторону. При проходе каждой вершины типа «соединение» совокупное число путей увеличивается, а поскольку целью является получить все возможные пути, то необходимо хранить уже полученные части

путей в списке в вершине. Благодаря выбранному порядку обхода у каждой следующей вершины будут обработаны все потомки, и поэтому для получения списка путей у каждого предка достаточно посчитать путь на соединяющем ребре и конкатенировать с соответствующим путём из списка. Запишем это формально.

Дано: Связный ориентированный ацикличный граф $D = (V,\ E)$. Для каждого ребра (u, v) фиксирована некоторая строка $S_{u,v} = s_1 s_2 \dots s_{L_{u,v}}$, где $s_i \in Q\ \forall\ i \in 1 : L_{u,v}$, а также полученная для этой строки матрица Витерби.

Необходимо найти: Множество P_{v_1} наиболее вероятных (скрытых) путей на графе.

Алгоритм:

1. Возьмём очередь U из прямого хода и обратим её, получив U'. Так будет получена очередь из вершин, обратная исходному порядку обработки, и каждая вершина в очереди будет лежать после всех потомков, что и требовалось. Сопоставим каждой вершине список оптимальных путей P_v, ведущих в эту вершину. Изначально все списки пустые.

2. Пока U' не пуста выполняем:

   а. Вынимаем из очереди U' вершину v.

   б. Для всех ребер $e = (u, v), u \in v^{\leftarrow}$, для каждого пути p из списка P_v выполняем: используя матрицу Витерби для строки S_e на ребре e и первое состояние пути p осуществляем поиск оптимального пути p_{S_e} согласно обратному ходу алгоритма Витерби и помещаем конкатенацию путей p_{S_e} и p в список P_u.

3. Возвращаем полученный список путей P_{v_1}, где v_1 — начальная вершина графа.

Итак, алгоритм Витерби обобщён на ацикличный ориентированный граф. Результатом его выполнения является набор всевозможных наиболее вероятных (скрытых) путей на графе.

4.4. Алгоритм решения задачи предсказания генов на графе

Дано:

- Связный ориентированный ацикличный граф $D = (V,\ E)$. Для каждого ребра (u, v) фиксирована некоторая строка $S_{u,v} = s_1 s_2 \dots s_{L_{u,v}}$, где $s_i \in Q\ \forall\ i \in 1 : L_{u,v}$.
- Известна вершина $v_1 \in V : |v_1^{\leftarrow}| = 0$ — начальная вершина с нулевой степенью захода.

Необходимо найти: Множество Ω кодирующих последовательностей для всех возможных путей на графе.

Алгоритм:

1. Создаём пустое множество Ω.
2. Запустим алгоритм Витерби из начальной вершины v_1 (степень захода равна нулю), в результате получим список оптимальных путей P_{v_1}.
3. Для каждого пути $w \in P_{v_1}$ запустим алгоритм поиска кодирующих последовательностей в выравнивании, полученные кодирующие последовательности поместим в множество Ω.

Глава 5

Валидация

Выполнена реализация алгоритма для решения задачи предсказания генов на графе на языке C++ [7]. В качестве параметров скрытой марковской модели были взяты параметры для целых геномов без ошибок секвенирования. Протестируем получившийся алгоритм и посмотрим на результаты, но прежде введем одно важное определение.

Определение 5.1. *Пусть дана СММ H с параметрами $\lambda = (P, B, p)$. Значимость (score) последовательности $S = s_0 s_1 \dots s_N$ со скрытым путём $x = x_0 x_1 \dots x_N$ и матрицей Витерби v равна $\ln \dfrac{\mathsf{P}(S|\lambda)}{\mathsf{P}(S|R)}$, где $\mathsf{P}(S|R)$ — вероятность того, что строка S была получена случайно (с тем же GC-составом, что и у генома, из которого получена кодирующая последовательность).*

Во FragGeneScan, поскольку все вероятности при вычислении матрицы Витерби берутся в логарифмах, значимость считается как

$$v_{x[N-4],\,N-4} - v_{x[3],3} - (N_a \ln p_A + N_c \ln p_C + N_g \ln p_G + N_t \ln p_T),$$

где p_i — вероятность получения символа i (определяется GC-составом генома), а N_i — число символов i в последовательности S, $N_a + N_t + N_g + N_c = N$ (разность вероятностей состояния, предшедствующего стоп-кодону и состояния, следующего за старт-кодоном, за вычетом логарифма вероятности случайного появления такой строки).

5.1. Тривиальный граф

5.1.1. Геном на графе

Для проверки корректности работы алгоритма взят геном *Escherichia coli*, GC-состав генома равен 0.507. Геном был разбит на пять частей, которые были последовательно расположены на ребрах графа (Рис 5.1).

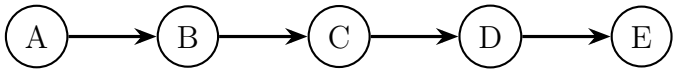


Рис. 5.1

	Число последовательностей в геноме *Escherichia coli*
На самом деле	4305
FragGeneScan на геноме	4565
Алгоритм на графах	4565

Таблица 5.1. Сравнение результатов.

В результате работы алгоритма на графах были получены такие же кодирующие последовательности, как и в результате работы алгоритма для целого генома (см. таблицу 5.1).

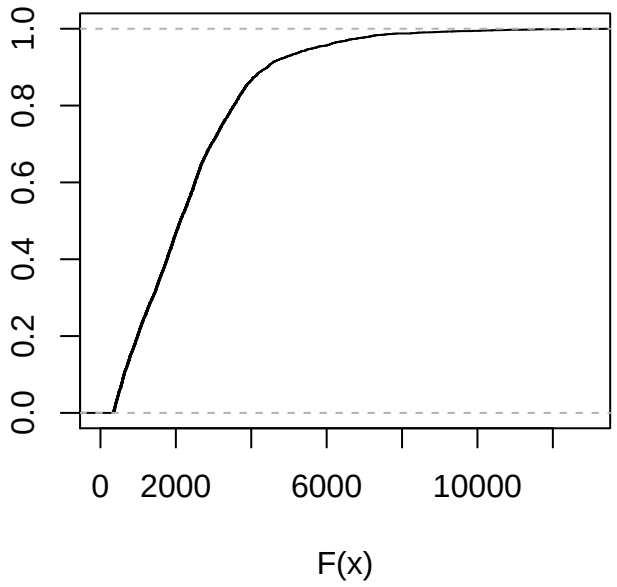


Рис. 5.2. Эмпирическая функция распределения значимости кодирующих последовательностей для *E. coli*.

Построим график эмпирической функции распределения значимости кодирующих последовательностей *Escherichia coli*, рисунок 5.2 (4565 последовательностей). Как видим, значимости имеют довольно большой разброс, у распределения присутствует тяжелый «хвост». Теперь, когда получена эмпирическая функция распределения значимости, можно ввести ещё одно определение.

Определение 5.2. *Значимостью (p-value) последовательности* $S = s_0 s_1 \dots s_N$ *называется* $\mathsf{P}(Score > S^*) = 1 - F(S^*)$, *где* S^* *— score строки S, а* F *— функция распределения score (эмпирическая) для генома, из которого взята данная последовательность.*

5.1.2. Удаления, вставки

На том же графе был проведен тест на корректную обработку сдвигов рамки считения (при наличие вставок/удалений). Первые 4200 нуклеотидов были расположены вдоль прямой, на каждом ребре по 1050 нуклеотидов. В таком графе было найдено три кодирующих последовательности: с 337 по 2799, с 2801 по 3733 и с 3734 по 4198 нуклеотид со scores, равными 6610, 2509 и 1231 соответственно. Первые две последовательности являются правильными, последняя же имеет правильную координату начала, но прерывается, правильная последовательность заканчивается на 5020-ой позиции.

Возьмём параметры для геномов, в которых вероятности перехода из состояния совпадения в состояние вставки/удаления равна 0.01 и изменим исходную последовательность, пусть каждый символ может быть удалён с той же вероятностью 0.01. В результате выполнено 47 удалений, получилось четыре кодирующих последовательности: с 341 по 636, с 640 по 1397, с 1440 по 2768 и с 3007 по 4152 нуклеотид. В первую кодирующую последовательность попало два удаления, во вторую и последнюю — по девять, а в третью — 15. Первые три последовательности — это участки первой правильной кодирующей последовательности, верно найдена координата её конца (с учётом 31 удаления). Вторая последовательность объединяет в себе участок второй и третью истинные последовательности. Отметим, что шесть удалений были успешно найдены моделью. Таким образом, большая часть истинных кодирующих последовательностей была обнаружена (пропущен небольшой участок длиной примерно 200 нуклеотидов).

Проведем аналогичный эксперимент со вставками: пусть между любыми двумя символами строки может оказаться вставлен символ с вероятностью 0.01. В результате работы алгоритма выполнено 38 вставок, отмечено две кодирующих последовательностей: с 342 по 3769, с 3770 по 4236 символ. До 342-го нуклеотида осуществлено пять вставок, поэтому начало первой кодирующей последовательности совпадает с истинным. В первую последовательность попала 31 вставка, (программой было найдено 18), с учетом этого получается, что эта последовательность является объединением первых двух истинных последовательностей. Вторая последовательность совпала с третьей ис-

тинной, а обе вставки (позиции 3792 и 4150) были корректно обнаружены и отмечены в скрытом пути.

Итак, вставки и удаления, конечно, вызывают проблемы при решении задачи поиска генов, но реализованный алгоритм на графе наследует одно из свойств метода FragGeneScan и успешно находит кодирующие последовательности на строках со вставками лишь с незначительной потерей качества.

5.2. Граф с развилками и соединениями

Для проверки корректности обработки программой структуры графа рассмотрим следующий граф (Рис 5.3).

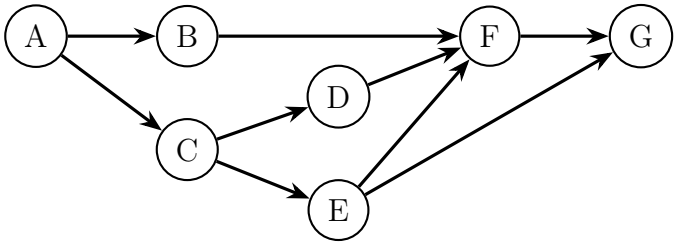


Рис. 5.3

ABFG (4200)			ACDFG (5600)			ACEG (4200)		
Начало	Конец	p-value	Начало	Конец	p-value	Начало	Конец	p-value
337	2799	0.02	1	165	0.96	1	165	0.96
2801	3733	0.39	181	804	0.6	181	804	0.6
3734	4198	0.72	964	1413	0.72	979	1131	0.97
			1512	1826	0.82	1215	2318	0.29
			1874	2797	0.5	2329	3063	0.78
			3426	4202	0.51	2707	3999	0.21
			4201	5133	0.39			
			5134	5598	0.72			

Таблица 5.2. Сравнение результатов на разных путях, в скобках указана общая длина пути.

На пути $ABFG$ последовательно расположены первые 4200 нуклеотидов генома *Escherichia coli* (по 1400 на каждом ребре), на остальных ребрах помещены случайные

участки того же генома. Результаты работы алгоритма отражены в таблице 5.2 (без пути $ACEFG$). Первые две кодирующие последовательности в пути ABFG — правильные, последняя — участок правильной, ограничен длиной взятой последовательности. Пути $ACDFG$ и $ACEG$ имеют общее ребро AC, на нём и были найдены два первых пути. Аналогично, так как последнее ребро FG совпадает у путей $ABFG$ и $ACDFG$ нашлись две попавшие на эти ребра последовательности (на пути $ACDFG$ их координаты сдвинуты на 1400). Наиболее значимыми (с меньшими p-value) были отмечены истинные последовательности на $ABFG$, а также последовательности 1215-2318, 2707-3999 на пути $ACEG$. Последняя последовательность на пути с участком настоящего генома $ABFG$ не получила ожидаемых значений, вероятно из-за того, что последовательность вынуждено прервалась.

Возьмём high-GC геном *Rhodobacter sphaeroides*, его GC-состав равен 68.97%. Всего в геноме найдена 2247 кодирующая последовательность со средней значимостью, равной 3105.6. Возьмём граф, как на рисунке 5.3 и сравним распределения значимостей кодирующих последовательностей найденных в геноме *R. sphaeroides* и на графе, строки на ребрах которого случайно взяты из генома. В результате получено 1970 кодирующих последовательностей, средняя значимость равна 702.8. Расположим на ребрах случайные строки с таким же GC-составом и такой же длины, что и в предыдущем эксперименте. Найдено значительно меньше, 486 кодирующих последовательностей, средняя значимость равна 510.6. Сопоставим результаты трёх экспериментов, построив график эмпирических функций распределения значимостей получившихся кодирующих последовательностей (Рис 5.4).

Проведём аналогичный эксперимент на геноме *Staphylococcus aureus*, его GC-состав равен $32,7\%$ (low-GC). Его результаты представлены на Рис 5.5. Как видно на графиках после проведённых экспериментов, значимости кодирующих последовательностей, полученных на графах, в которых строки были взяты случайно из генома или просто сгенерированы с тем же GC-составом в среднем меньше, чем значимости исходного генома, а значит кодирующие последовательности с малым значением значимости должны быть более тщательно проверены, и, возможно, исключены из множества кодирующих.

Выполнена проверка алгоритма на простых примерах. На графе, состоящем из нескольких последовательно идущих ребрах (Рис 5.1) работа алгоритма аналогична FragGeneScan на строке, результаты полностью сопадают. Вставки и удаления симво-

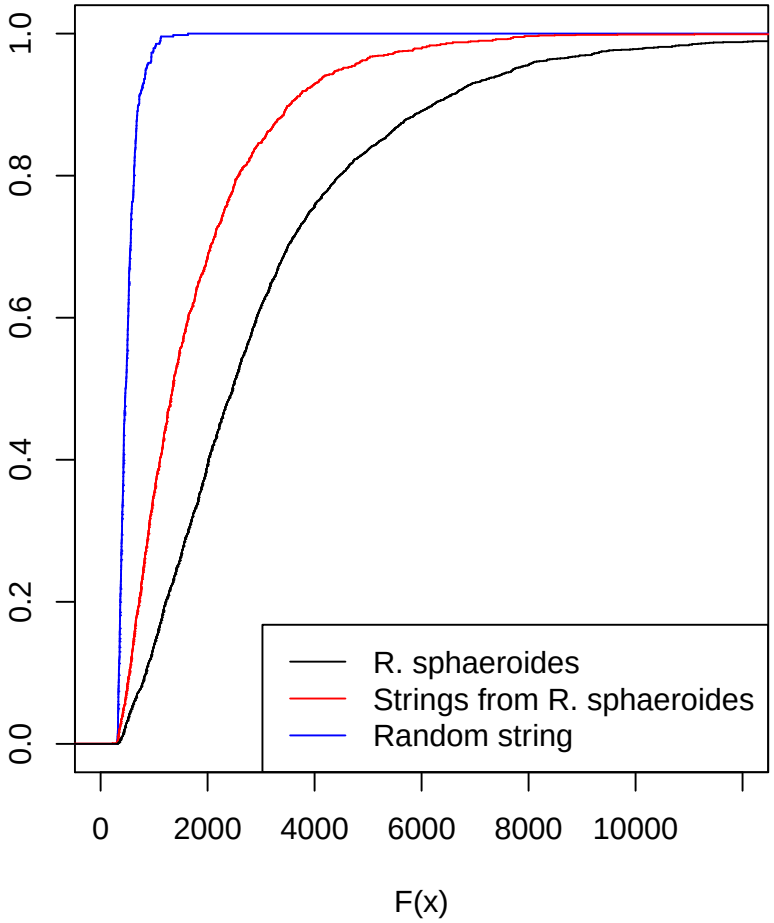


Рис. 5.4. Эмпирические функции распределения значимости кодирующих последовательностей для *Rhodobacter sphaeroides*: 1) черная линия — значимости генома, 2) красная — значимости на графе, строки на котором получены случайно из *R. sphaeroides*, 3) синяя — значимости на графе со случайно сгенерированными строками с тем же GC-составом, что и у генома.

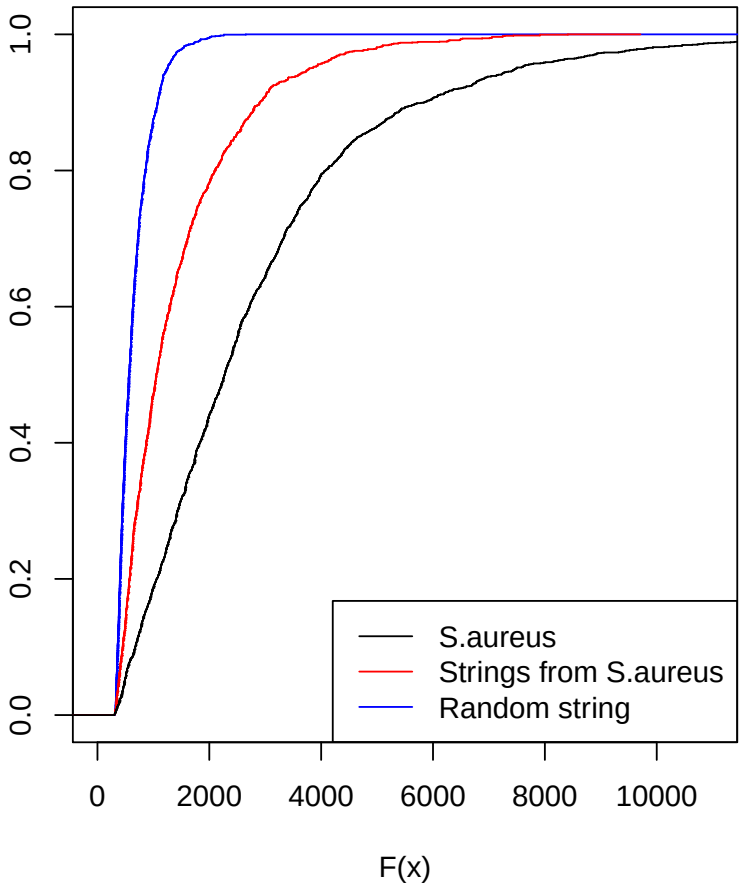


Рис. 5.5. Эмпирические функции распределения значимости кодирующих последовательностей для *Staphylococcus aureus*: 1) черная линия — значимости генома (2683 последовательности), 2) красная — значимости на графе, строки на котором получены случайно из *S. aureus* (2250 последовательности), 3) синяя — значимости на графе со случайно сгенерированными строками с тем же GC-составом, что и у генома (2384 последовательностей).

лов строки могут разделить полученные кодирующие последовательности на несколько частей, либо, напротив, объединиться в более длинные. При усложнении структуры графа алгоритмом корректно находятся все пути, которые затем размечаются на кодирующие последовательности. Таким образом, результаты работы алгоритма на графах соответствуют структуре графа и биологическим особенностям, и алгоритм может быть применён для более сложных графов.

Заключение

В ходе исследования была изучена задача предсказания генов и методы её формализации. Введено понятие скрытой марковской модели, рассмотрен алгоритм Витерби как способ нахождения наиболее оптимального пути скрытых состояний. Также описан способ решения задачи предсказания генов при помощи алгоритма Витерби. Была изучена скрытая марковская модель FragGeneScan и основанный на ней алгоритм решения задачи для целой нуклеотидной последовательности (генома). Описана задача предсказания генов на графах, разработан алгоритм для её решения на ориентированных ацикличных графах, основанный на модификации алгоритма Витерби.

Удалось выполнить реализацию описанного алгоритма на языке программирования C++ и показать, что, как на простых графах, так и на графах с развилками и соединениями алгоритм даёт разумные результаты. Такой алгоритм позволяет наследовать свойства FragGeneScan и корректно обрабатывать сдвиги рамки считения, вызванные ошибками при секвенировании. Было проведено несколько экспериментов на графах, на которых были расположены как сами геномы *Escherichia coli*, *Staphylococcus aureus*, *Rhodobacter sphaeroides*, так и их случайные участки, либо случайно сгенерированные последовательности. Таким образом, основным результатом данной работы является то, что удалось показать перспективность решения задачи предсказания генов на графах в обход сборки генома.

В дальнейшем планируется расширить класс графов, к которому может быть применён алгоритм (в первую очередь на графы с циклами), а также применить алгоритм на практике при решении задачи предсказания генов на реальных графах сборки.

Список литературы

1. Improved microbial gene identification with GLIMMER / Arthur L. Delcher, Douglas Harmon, Simon Kasif et al. // Nucleic Acids Research. — 1999. — 12. — Vol. 27, no. 23. — P. 4636–4641.
2. Noguchi H., Park J., Takagi T. MetaGene: prokaryotic gene finding from environmental genome shotgun sequences // Nucleic Acids Research. — 2006. — 10. — Vol. 34, no. 19. — P. 5623–5630.
3. Rho M., Tang H., Ye Y. FragGeneScan: predicting genes in short and error-prone reads // Nucleic Acids Research. — 2010. — 08. — Vol. 38, no. 20. — P. e191–e191.
4. Prodigal: prokaryotic gene recognition and translation initiation site identification / Doug Hyatt, Gwo-Liang Chen, Philip F LoCascio et al. // BMC Bioinformatics. — 2010. — Mar. — Vol. 11, no. 1. — Access mode: https://doi.org/10.1186/1471-2105-11-119.
5. Besemer J., Borodovsky M. GeneMark: web software for gene finding in prokaryotes, eukaryotes and viruses // Nucleic Acids Research. — 2005. — Jul. — Vol. 33, no. Web Server. — P. W451–W454. — Access mode: https://doi.org/10.1093/nar/gki487.
6. Introduction to Mathematical Methods in Bioinformatics. — Springer Berlin Heidelberg, 2006. — Access mode: https://doi.org/10.1007/978-3-540-48426-4.
7. Popov V. Fraggenescan modification for graphs. — 2021. — May. — Access mode: https://doi.org/10.5281/zenodo.4812807.