МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий
Кафедра параллельных вычислений

Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль): Программная инженерия и компьютерные науки

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Матус Натальи Евгеньевны

Тема работы:

РАЗРАБОТКА БАЗЫ АКТИВНЫХ ЗНАНИЙ ДЛЯ АВТОМАТИЧЕСКОГО КОНСТРУИРОВАНИЯ ПРОГРАММ АНАЛИЗА ВТОРИЧНОЙ СТРУКТУРЫ РНК

«К защите допущена»
Заведующий кафедрой,
д.т.н, профессор
Малышкин В.Э./……..
(ФИО) / (подпись)
«31» мая 2024 г.

Руководитель ВКР
д.т.н., профессор
зав. каф. ПВ ФИТ НГУ
Малышкин В.Э./……..
(ФИО) / (подпись)
«31» мая 2024 г.

Соруководитель ВКР
к.т.н.
ст. преп. каф. ПВ ФИТ НГУ
Перепёлкин В.А./……..
(ФИО) / (подпись)
«31» мая 2024 г.

Новосибирск, 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)
Факультет информационных технологий
Кафедра параллельных вычислений

Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль): Программная инженерия и компьютерные науки

УТВЕРЖДАЮ

Зав. кафедрой Малышкин В.Э
(фамилия, И., О.)
…….
(подпись)
«06» ноября 2023 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

Студенту*(ке)* Матус Наталье Евгеньевне, группы 20202
(фамилия, имя, отчество, номер группы)

Тема Разработка базы активных знаний для автоматического конструирования программ анализа вторичной структуры РНК
(полное название темы выпускной квалификационной работы)

утверждена распоряжением проректора по учебной работе от 03.11.2023 № 0413

Срок сдачи студентом готовой работы: 20 мая 2024 г.

Исходные данные (или цель работы): разработать базу активных знаний для автоматического конструирования программ анализа вторичной структуры РНК.

Структурные части работы: обзор предметной области, формулирование требований к составу базы активных знаний, анализ модулей анализа вторичной структуры РНК, разработка и реализация базы активных знаний, тестирование базы активных знаний.

Руководитель ВКР
зав. каф. ПВ ФИТ НГУ,
д.т.н, профессор
Малышкин В.Э./……..
(ФИО) / (подпись)
«06» ноября 2023 г.

Задание принял к исполнению
Матус Н.Е. /……..
(ФИО студента) / (подпись)
«06» ноября 2023 г.

Соруководитель ВКР
ст. преп. каф. ПВ ФИТ НГУ,
к.т.н.
Перепёлкин В.А./……..
(ФИО) / (подпись)
«06» ноября 2023 г.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Рибонуклеиновая кислота (РНК) — одна из основных макромолекул, которая содержится в клетках всех живых организмов; играет важную роль в кодировании, прочтении, регуляции и экспрессии генов.

Вычислительная модель – это двудольный ориентированный конечный граф, доли которого образуют два множества вершин, называемых множествами операций и переменных.

Вычислительный модуль – библиотечная подпрограмма и формальное описание этой подпрограммы (описание входных и выходных переменных, тип этих переменных, нефункциональные свойства подпрограммы и т. п.)

База активных знаний – совокупность вычислительных моделей и вычислительных модулей.

ВВЕДЕНИЕ

Компьютерные методы и информационные технологии анализа данных широко используются для интерпретации биологических процессов, связанных с биомолекулами, исследования последовательностей нуклеиновых кислот и белков, моделирования объемной структуры биомолекул и их взаимодействий между собой, например, при разработке лекарств или выявлении генетических мутаций [1].

В ходе анализа и обработки биологических данных применяется множество вычислительных алгоритмов и преобразований [2]. Часто требуется выполнить ряд таких преобразований, поэтому программа выглядит как упорядоченное прохождение входных данных через ряд вычислительных модулей, которые находятся на разных веб-ресурсах или представлены как программы из различных библиотек. Кроме того, выходные данные одного модуля и входные данные следующего могут требовать разных форматов. В этом случае в программу дополнительно нужно добавлять функции для преобразования одного формата в другой. При этом последовательность и перечень используемых модулей могут меняться в зависимости от задач предметной области, что требует дополнительных преобразований. Необходимость применять несколько различных программных инструментов для вычислительных экспериментов приводит к проблеме обеспечения их корректного совместного использования и предполагает наличие специальных компетенций у пользователя.

На данный момент существуют системы, позволяющие конструировать вычислительные конвейеры для работы с биологическими данными, такие как Galaxy – веб-платформа для геномных исследований [3] и GenePattern – веб-платформа для анализа экспрессии генов [4]. Однако в данных системах под каждую задачу необходимо вручную определять нужные вычислительные модули и устанавливать порядок их исполнения на конвейере. Также в данных системах не учитываются нефункциональные свойства модулей.

Решением данной проблемы может стать создание базы активных знаний [5], представляющей собой совокупность вычислительных моделей и модулей (подпрограмм различных библиотек) [6]. Такая база будет содержать в себе формальное описание задач предметной области, основанное на математическом аппарате вычислительных моделей. Система активных знаний, на основе которой создается база активных знаний, позволяет автоматически конструировать программы. Таким образом с пользователя снимается проблема поиска необходимых для решения задачи вычислительных модулей и обеспечение корректности порядка, в котором они будут исполняться.

Цель работы – разработка базы активных знаний для автоматического конструирования программ анализа вторичной структуры РНК.

Для достижения этой цели были поставлены следующие **задачи**:

1. Анализ библиотечных программ анализа вторичной структуры РНК на предмет их совместимости, технической возможности их включения в базу активных знаний, на предмет нефункциональных свойств.
2. Разработка способа описания нефункциональных свойств модулей.
3. Построение формального описания предметной области на основе математического аппарата вычислительных моделей.
4. Разработка алгоритмов системы активных знаний на основе существующих прототипов.
5. Экспериментальное исследование созданной базы активных знаний.

Новизна работы заключается в создании базы активных знаний для автоматического конструирования программ анализа вторичной структуры РНК. Впервые применен комплексный подход, заключающийся в учете функциональных и нефункциональных свойств модулей.

Практическая ценность работы определяется созданием удобного и доступного для широкого круга пользователей инструмента, позволяющего проводить за сокращенное время разносторонний и комплексный анализ вторичной структуры РНК без специальной подготовки. Изучение принципов пространственной организации РНК представляется важным и с точки зрения

теории (вопросы разнообразия и эволюции структур, идентификация ключевых структурных элементов РНК, задействованных в клеточных процессах) и с точки зрения практики (рациональный дизайн РНК с заданной структурой, таргетированное изменение структурных элементов, разработка лекарств, редактирование генома).

Структура работы содержит следующие разделы: Обзор существующих платформ для работы с биологическими данными и подходов к формализации предметной области, Проектирование базы активных знаний, Реализация и тестирование базы активных знаний.

1 Обзор существующих платформ для работы с биологическими данными и подходов к формализации предметной области

1.1 Платформы для работы с биологическими данными

Для работы с биологическими данными разработан ряд платформ, отличающихся уровнем порога вхождения, функциональностью и форматом представления данных. Целью первой части обзора является сравнительный анализ существующих платформ для работы с биологическими данными: GenePattern, Galaxy по следующим критериям: порог вхождения, функциональность, вариативность решения, способ хранения данных.

1.1.1 GenePattern

GenePattern – открытая веб-платформа для геномных исследований [3], специализирующаяся на анализе экспрессии генов. Пользователю предоставляется возможность составлять цепочки из уже имеющихся модулей анализа геномных данных. Пример создания такой цепочки представлен на рисунке 1. Также пользователь может интегрировать в платформу собственные модули путем описания их в конфигурационном файле. Важной особенностью платформы является то, что для ее использования пользователь может не обладать навыками программирования, а также разбираться в деталях реализации того или иного инструмента для работы с биологическими данными. К основным недостатками платформы можно отнести узкую функциональность системы, отсутствие удобного графического интерфейса и сложность экспорта-импорта данных.

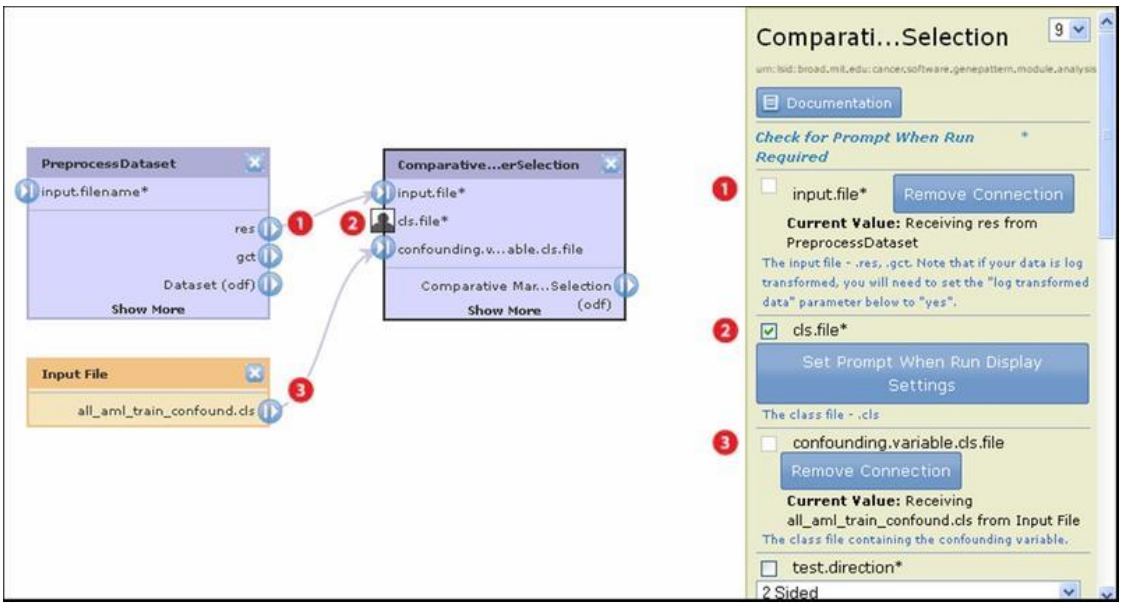

Рисунок 1 – Пример конвейера GenePattern

1.1.2 Galaxy

Galaxy является открытой веб-платформой для биомедицинских исследований [4]. С помощью графического интерфейса Galaxy (представлен на рисунке 2) пользователь может конструировать вычислительные конвейеры и запускать их. Платформа обладает низким порогом вхождения и схожей с GenePattern функциональностью, однако имеет ряд преимуществ, среди которых наиболее важными являются: предоставление пользователю облачного хранилища данных, что позволяет сохранять историю вычислений и повторно использовать результаты предыдущих экспериментов; возможность размещения своих конвейеров в публичных репозиториях; проведение вычислений на конвейерах других пользователей.

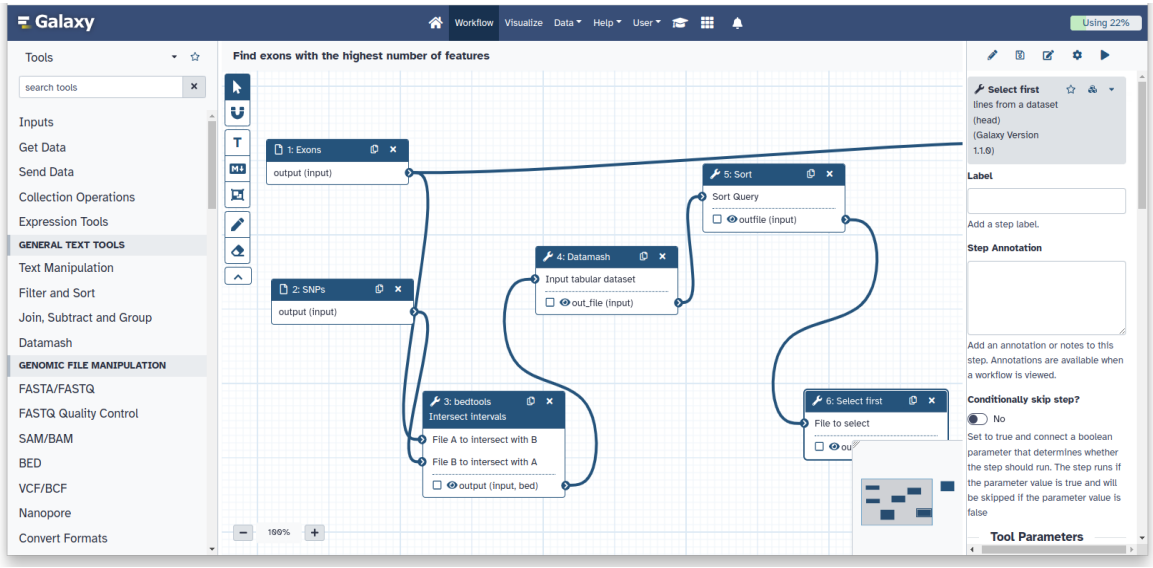

Рисунок 2 – Пример конвейера Galaxy

1.1.3 Сравнительный анализ платформ

В таблице 1 представлена сравнительная характеристика платформ: GenePattern, Galaxy. Видно, что рассмотренные платформы обладают низким порогом вхождения и схожей функциональностью. Однако платформа Galaxy обеспечивает более удобное хранение данных и результатов вычислений. Общим недостатком этих платформ является необходимость пользователю строго определять состав модулей и порядок их исполнения для каждой конкретной задачи.

Таблица 1 – Характеристика платформ

Платформа	Критерии сравнения			
	Низкий порог вхождения	Широкая функциональность	Вариативность решения	Хранение данных
GenePattern	+	–	–	–
Galaxy	+	±	–	+

1.2 Подходы к формализации предметной области

По мере накопления знаний в определенной предметной области, число понятий и связей между ними значительно увеличивается, что приводит к

необходимости структурировать информацию. Во второй части обзора приведено описание основных подходов к формализации предметной области, а также рассмотрены их преимущества и недостатки с точки зрения автоматического конструирования программ.

1.2.1 Библиотеки подпрограмм

Для проведения вычислительных экспериментов в области биоинформатики существуют стандартные библиотеки, содержащие в себе различные инструменты для обработки и анализа данных биологических исследований. Из наиболее популярных можно выделить BioPython – библиотека для вычислений в области молекулярной биологии [7], ViennaRNA Package – библиотека, используемая для прогнозирования и анализа вторичных структур РНК [8], Infernal – библиотека для поиска гомологичных нуклеотидных последовательностей [9]. Также существует множество автономных программ, представляющих собой исполняемый файл и отвечающих за одну операцию, например, Clustal Omega – программа для множественного выравнивания последовательностей [10], BLAST – программа для поиска гомологичных белков [11].

Часто для проведения всестороннего анализа возникает необходимость использования нескольких библиотек одновременно. В таком случае пользователь должен самостоятельно собрать и подключить к своей программе каждую из библиотек, обеспечить корректность порядка применения подпрограмм этих библиотек, а также провести промежуточные преобразования данных из одного формата в другой при несовпадении выходных данных одной подпрограммы и входных данных другой. Таким образом, пользователь должен обладать компетенциями как со стороны биологии – для того, чтобы обеспечить корректность применения функций библиотек с точки зрения задач предметной области, так и со стороны программирования – чтобы установить необходимые инструменты и написать программу, выполняющую необходимые вычисления.

1.2.2 Системы на основе онтологий

Онтология (в информатике) – это модель представления объектов в виде набора понятий, их определений, свойств и отношений между ними [12]. Онтологии позволяют создать универсальную терминологию и представить понятия предметной области, в т.ч. в пригодном для машинной обработки виде.

На данный момент в области биологии разрабатываются и активно используются различные онтологии [13]. Так, например, проект GO [14] (Gene Ontology) содержит более 40 тыс. терминов. Каждый термин представлен набором из идентификатора, названия термина, определения, синонимов и ссылок на связанные термины. Термины, содержащиеся в GO, служат для аннотации генома – процесса описания функциональных свойств элементов генома.

Недостатком онтологий с точки зрения автоматического конструирования программ является то, что для создания такого инструмента, помимо описания понятий предметной области, потребуется дополнительно описывать некоторые более низкоуровневые термины, такие как «программа», «операция», «переменная».

1.2.3 Экспертные системы

Экспертная система – это система, аккумулирующая в себе знания человека, обладающего компетенциями в определенной предметной области (эксперта) и способная частично заменить специалиста в решении задач [15]. Знания эксперта в такой системе представляются в виде набора правил, позволяющих делать логические выводы, например, набор правил вида «если…, то…». Взаимодействие пользователя с экспертной системой строится следующим образом: пользователь предоставляет системе набор входных данных (фактов), к которым система применяет логические правила и на выход отдает экспертный совет или оценку.

В качестве примера экспертной системы в области биоинформатики можно привести BETSY (Bioinformatics ExperT SYstem). Система позволяет генерировать конвейеры из вычислительных модулей для секвенирования

биополимеров [16]. В базе знаний BETSY содержатся вычислительные модули с описанием их характеристик. На вход системе пользователь подает входные данные и описание формата необходимых выходных данных. Используя обратный механизм вывода, система предлагает возможные конвейеры из модулей, применение которых приведет к желаемому результату. Далее пользователь указывает дополнительные параметры, например, выбирает предпочтительный алгоритм, что позволяет системе выбрать из множества конвейеров наилучший и вычислить результат.

Недостатком экспертных систем является то, что пользователю необходимо принимать решение в пользу тех или иных алгоритмов и параметров конвейера, что требует определенных компетенций и существенно повышает порог вхождения.

1.2.4 Семантические сети

Семантическая сеть – это способ представления информации в виде ориентированного графа [17]. Объекты предметной области представляются в виде вершин графа, а ребра, которыми они соединяются обозначаются отношения между объектами, такие, как, например, «is-a» (является) или «has-a» (имеет). Также объекты могут быть связаны отношением импликации, что позволяет определить причинно-следственную связь. Помимо представления знаний в виде графа, семантические сети предоставляют возможность дедуктивного вывода новых знаний.

Накопление знаний в виде семантической сети является эффективным способом организации информации в биоинформатике за счет высокой иерархичности биологических терминов [18]. Так, например, семантическая сеть, входящая в состав системы Unified Medical Language System (UMLS), содержит более 2.5 млн терминов биомедицины, разбитых на 127 категорий [19]. Семантическая сеть поддерживает не только иерархические связи между понятиями, но и ассоциативные связи, такие как, например «may-cause» (может вызвать) и «related-to» (связан с). UMLS используется, в основном, для обработки естественного языка при семантическом анализе текста.

Недостатком семантических сетей, так же, как и в случае с онтологиями, является то, что данный подход к формализации предметной области не ориентирован на автоматическое конструирование программ: в нем отсутствуют такие понятия, как «операция», «переменная» и т. п.

1.2.5 Активные знания

Концепция активных знаний – это подход, основанный на теории синтеза параллельных программ и систем на вычислительных моделях [6].

При таком подходе предметная область представляется как вычислительная модель. Вычислительная модель – это двудольный ориентированный конечный граф, доли которого образуют два множества вершин, называемых множествами операций и переменных. Схематическое представление вычислительной модели представлено на рисунке 3. Входящие в операцию и исходящие из неё дуги определяют входные и выходные переменные операции соответственно. Операциям вычислительной модели ставятся в соответствие вычислительные модули (одной операции могут соответствовать несколько модулей, совпадающих по множеству входных и выходных переменных). Вычислительный модуль представляет собой некоторую подпрограмму и формальное описание этой подпрограммы (описание входных и выходных переменных, тип этих переменных, нефункциональные свойства подпрограммы и т. п.) [20]. Совокупность вычислительных моделей и вычислительных модулей образует *базу активных знаний*.

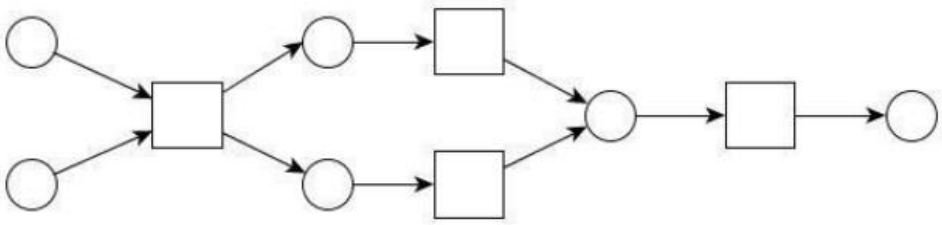

Рисунок 3 – Пример вычислительной модели; круги – переменные, квадраты – операции

Если на множестве переменных вычислительной модели выделить два подмножества – подмножество входных переменных некоторой задачи и

множество выходных переменных задачи, то задача нахождения способа вычисления выходных переменных из входных (построение плана вычислений) и последующее конструирование программы из модулей может осуществляться программно, т.е. автоматически. Система, выполняющая перечисленные выше функции, называется *системой активных знаний*. Стоит отметить, что у задачи может быть несколько вариантов путей исполнения. В таком случае, выбор оптимального плана вычисления обеспечивается системой активных знаний и может приниматься как исходя из доступных вычислительных ресурсов (количество ядер вычислителя, объем доступной памяти), так и исходя из нефункциональных свойств модулей (время исполнения, расход памяти, нагрузка на сеть, а также свойства, характерные для той или иной предметной области).

На данный момент уже существуют и активно разрабатываются прототипы систем активных знаний. Так, например, подсистема Aka системы HPC Community Cloud [21] является онлайн-IDE для разработки вычислительных моделей и позволяет создавать собственные вычислительные модели, описывать модули и вычислять задачи, описанные созданными вычислительными моделями.

1.3 Выводы

Сравнительный анализ существующих платформ для работы с биологическими данными (GenePattern, Galaxy) по различным критериям (порог вхождения, функциональность, вариативность решения, способ хранения данных) показал, что хотя данные инструменты существенно снижают порог вхождения (пользователю не нужны специальные навыки в области программирования), в то же время, они обладают рядом недостатков: отсутствие необходимых для анализа вторичных структур РНК модулей, необходимость пользователю строго определять состав модулей и порядок их исполнения для каждой конкретной задачи, отсутствие вариативности пути исполнения, отсутствие учета нефункциональных свойств модулей.

Проведенный обзор подходов к формализации предметной области (библиотеки подпрограмм, онтологии, экспертные системы, семантические сети, активные знания) показал, что с точки зрения автоматического решения прикладных задач биоинформатики, концепция активных знаний является наиболее подходящей за счет ориентированности данного подхода на синтез программ, а также за счет существования определенных наработок в данной области, в т. ч. прототипов систем активных знаний.

2 Проектирование базы активных знаний

2.1 Биоинформатика в решении задач анализа пространственных структур макромолекул

Биоинформатику можно определить как применение вычислительных инструментов для организации, анализа, понимания, визуализации и хранения информации, связанной с биологическими макромолекулами [22]. Это междисциплинарная наука, в основном включающая молекулярную биологию и генетику, информатику, математику и статистику [23]. Анализы в биоинформатике преимущественно сосредоточены на трех типах больших наборов данных, доступных в молекулярной биологии: структурах макромолекул, последовательностях генома и результатах экспериментов по функциональной геномике (например, данных об экспрессии). Биоинформатика использует широкий спектр вычислительных методов, включая выравнивание последовательностей и структур, интеллектуальный анализ данных, макромолекулярную геометрию, построение филогенетического дерева, предсказание структуры и функции белка, поиск генов и кластеризацию данных об экспрессии. Акцент делается на подходах, объединяющих различные вычислительные методы и разнородные источники данных.

Биоинформатика – это практико-ориентированная наука, которая находит широкое применение в различных областях, таких как биохимия, биофизика, фармакология, экология. Биоинформационные инструменты полезны для анализа и интерпретации геномных данных, технологии обнаружения бактерий, устойчивых к антибиотикам [24].

Одним из широко используемых приложений биоинформатики является идентификация трехмерных структур макромолекул, необходимая для предсказания их функции и разработки биомедицинских препаратов для лечения многих сложных заболеваний человека [25]. В частности, функциональные свойства РНК тесно связаны с ее способностью образовывать сложные иерархические структуры [26]. Структуру РНК можно описать иерархически: первичная, вторичная, третичная и четвертичная структуры (рисунок 4).

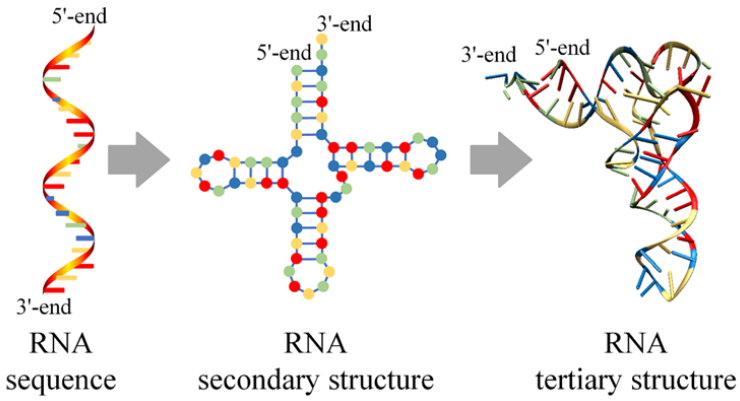

Рисунок 4 – Первичная (слева), вторичная (в центре) и третичная структуры РНК (справа) [27]

Первичная структура описывает нуклеотидную последовательность, вторичная структура описывает взаимодействия комплементарных нуклеотидов, третичная структура — укладку цепи в пространстве, а четвертичная — взаимодействие РНК с другими биомолекулами. Поскольку РНК состоит из нуклеотидов, которые формируют как одноцепочечные, так и двухцепочечные структуры, образуется разнообразный набор элементов вторичной структуры, включая стебли, шпильки, выпуклости, внутренние петли, соединения и псевдоузлы. Эти элементы вторичной структуры могут взаимодействовать друг с другом, приводя к образованию более сложных третичных структур. Функции РНК зависят от того, как меняется их трехмерная структура в ответ на разнообразный набор клеточных состояний. Понимание структуры РНК имеет важное значение для понимания их функций и разработки фармацевтических препаратов [28].

2.2 Задача предсказания вторичной структуры РНК

Начальный этап анализа структуры РНК включает получение информации о вторичной структуре, которая служит основой для изучения соответствующей третичной структуры. Именно от вторичной структуры РНК зависят ее функциональные свойства.

Рентгеновская кристаллография и ядерный магнитный резонанс (ЯМР) являются прямыми и наиболее точными подходами к определению структур РНК. Однако оба метода характеризуются высокой стоимостью и низкой

пропускной способностью, ограничивающие их широкое использование. Кроме того, молекулы РНК очень нестабильны и трудно кристаллизуются. Фактически на сегодняшний день лишь для очень небольшого процента (<0,001%) известных РНК структура была определена экспериментально.

Помимо экспериментальных методов, вторичную структуру РНК можно предсказать вычислительно с помощью прогнозирования или с использованием методов машинного обучения [27]. Данные подходы имеют большие преимущества: они могут предсказать структуру для абсолютно любой последовательности, не требуя наличия экспериментальных данных. С 1970-х годов был разработан ряд методов прогнозирования, основанных на различных гипотезах механизма сворачивания РНК. Большинство из этих методы пытаются идентифицировать структуру с минимальной свободной энергией (МСЭ), в соответствии с гипотезой о том, что молекула РНК, вероятно, существует в состоянии МСЭ, как и белок [29]. С использованием этих методов было разработано множество программных приложений, включая методы на основе глубокого машинного обучения.

Вычислительные методы предсказания вторичной структуры РНК можно разделить на два крупных класса:

- Предсказание вторичной структуры РНК на основе одной последовательности.

Данные вычислительные методы прогнозируют вторичную структуру молекулы, основываясь на первичной структуре этой молекулы. Наиболее распространенные алгоритмы основываются на минимизации свободной энергии молекулы, на определении вероятностей спаривания нуклеотидов, на поиск субоптимальных структур.

- Предсказание вторичной структуры РНК на основе нескольких последовательностей.

Данные вычислительные методы предсказывают общую вторичную структуру для множества схожих последовательностей РНК. Как правило, алгоритмы предсказания общей структуры для ряда последовательностей

требуют предварительного выравнивания этих последовательностей. Выравнивание последовательностей представляет собой процесс записи последовательностей в виде строк матрицы, таким образом, чтобы схожие участки РНК находились друг под другом. Такой способ представления данных позволяет увидеть функциональные, структурные и эволюционные взаимосвязи последовательностей.

2.3 Требования, предъявляемые к базе активных знаний

В данном разделе сформулированы требования, предъявляемые к разрабатываемой базе активных знаний. Требования можно разделить на две категории: требования, предъявляемые к библиотечным подпрограммам анализа вторичной структуры РНК (которые впоследствии будут включаться в базу активных знаний в виде вычислительных модулей) и требования, предъявляемые к вычислительным моделям базы активных знаний.

2.3.1 Требования, предъявляемые к библиотечным подпрограммам анализа вторичной структуры РНК

В данном разделе сформулированы требования к библиотекам работы с РНК и подпрограммам из этих библиотек, которые будут использоваться при создании базы активных знаний (включены в базу в качестве вычислительных модулей).

На основании ранее выполненных работ в Лаборатории молекулярно-генетических систем Института цитологии и генетики СО РАН [1, 30] сформулированы следующие требования к составу модулей базы активных знаний (функциональные требования):

1. Предсказание вторичной структуры РНК по первичной структуре РНК.
2. Множественное выравнивание последовательностей первичных структур РНК.
3. Предсказание общей вторичной структуры РНК по множеству первичных структур РНК.

Также были сформулированы желательные нефункциональные характеристики:

1. Возможность установки библиотек на операционную систему Linux.

2. Совместимость подпрограмм по формату данных.

2.3.2 Требования, предъявляемые к вычислительным моделям базы активных знаний

Сформулированы следующие требования к вычислительным моделям, которые будут включены в базу активных знаний:

1. Возможность вариативного исполнения (вычислительная модель содержит несколько альтернативных вариантов вычисления переменных).

2. Параллельное применение модулей.

Выполнение этих требований позволит конструировать более эффективную программу по вычислительной модели.

2.4 Анализ библиотечных подпрограмм анализа вторичной структуры РНК

В данном разделе представлено описание библиотечных подпрограмм, выбранных в качестве модулей базы активных знаний на основании ранее выполненных работ в Лаборатории молекулярно-генетических систем Института цитологии и генетики СО РАН [1, 30]. Приведен результат анализа этих подпрограмм на соответствие требованиям, сформулированным в п. 2.3.1.

2.4.1 Форматы файлов для работы со структурами РНК

Одним из критериев анализа является исследование совместимости подпрограмм из различных библиотек по формату данных.

Форматы для записи первичных структур РНК

Так как первичная структура РНК отражает порядок чередования нуклеотидов в полинуклеотидной цепи, то в основе всех форматов лежит строка, состоящая из букв. Каждая буква в строке соответствует нуклеотиду. В случае РНК, таких нуклеотидов 4 – аденин (A), гуанин (G), цитозин (C) и урацил (U).

Рассмотрим наиболее распространенные форматы записи первичных структур РНК: FASTA и CLUSTAL.

FASTA

Файл в формате FASTA (расширение .fa), кроме самой последовательности нуклеотидов содержит строку-титр, начинающуюся с символа «>». Первое слово в строке содержит название последовательности, остальные – описание последовательности. После строки-титра идет запись самой последовательности. Пример файла в формате FASTA представлен на рисунке 5.

```
>NG_008679.1:5001-38170 Homo sapiens paired box 6 (PAX6)
ACCCTCTTTTCTTATCATTGACATTTAAACTCTGGGGCAGGTCCTCGCGTAGAACGCGGCTGTCAGATCT
GCCACTTCCCCTGCCGAGCGGCGGTGAGAAGTGTGGGAACCGGCGCTGCCAGGCTCACCTGCCTCCCCGC
CCTCCGCTCCCAGGTAACCGCCCGGGCTCCGGCCCCGGCCCGGCTCGGGGCCCGCGGGGCCTCTCCGCTG
CCAGCGACTGCTGTCCCCAAATCAAAGCCCGCCCCAAGTGGCCCCGGGGCTTGATTTTTGCTTTTAAAAG
GAGGCATACAAAGATGGAAGCGAGTTACTGAGGGAGGGATAGGAAGGGGGGTGGAGGAGGGACTTGTCTT
TGCCGAGTGTGCTCTTCTGCAAAAGTAGCAAAATGTTCCACTCCTAAGAGTGGACTTCCAGTCCGGCCCT
GAGCTGGGAGTAGGGGGCGGGAGTCTGCTGCTGCTGTCTGCTAAAGCCACTCGCGACCGCGAAAAATGCA
GGAGGTGGGGACGCACTTTGCATCCAGACCTCCTCTGCATCGCAGTTCACGACATCCACGCTTGGGAAAG
TCCGTACCCGCGCCTGGAGCGCTTAAAGACACCCTGCCGCGGGTCGGGCGAGGTGCAGCAGAAGTTTCCC
GCGGTTGCAAAGTGCAGATGGCTGGACCGCAACAAAGTCTAGAGATGGGGTTCGTTTCTCAGAAAGACGC
```

Рисунок 5 – Пример файла в формате FASTA [31]

CLUSTAL

Формат CLUSTAL (расширение. aln) служит для записи нескольких выровненных первичных структур РНК. Он представляет собой набор блоков. В блоке содержится несколько строк, каждая из которых отвечает одной последовательности. Кроме самой последовательности, в строке указывается ее наименование и число идентичных остатков в последовательностях блока. В конце каждого блока добавляется строка, описывающая степень сходства последовательностей. Пример файла в формате CLUSTAL представлен на рисунке 6.

```
FOSB_MOUSE      LPGSTSAKEDGFGWLLPPPPPPPLPFQSSRDAPPNLTASLFTHSEVQVLGDPFPVVSPSY 300
FOSB_HUMAN      LPGSAPAKEDGFSWLLPPPPPPPLPFQTSQDAPPNLTASLFTHSEVQVLGDPFPVVNPSY 300
                ****:.******.**************:*:*************************.***
```

Рисунок 6 – Пример файла в формате CLUSTAL

Форматы для записи вторичных структур РНК

Плоский рисунок

Представляет собой рисунок, отображающий расположение нуклеотидов и связь между ними. Представлен на рисунке 7.

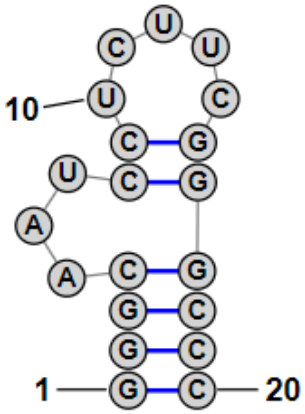

Рисунок 7 – Вторичная структура РНК в формате плоского рисунка

Dot-Bracket Notation (DBN)

Формат Dot-Bracket Notation (расширение .dot, .bracket, .dbn) является текстовым отображением вторичной структуры РНК. По аналогии с форматом FASTA, файл DBN начинается со строки-титра, описывающей последовательность. Далее записывается первичная структура РНК в виде буквенной последовательности. Под строкой с первичной структурой идет строка, описывающая вторичную последовательность. Эта строка состоит из символов точки и скобок: точка соответствует неспаренному нуклеотиду, а открытая и закрытая скобка – левому и правому спаренному нуклеотиду соответственно. Пример файла в формате DBN представлен на рисунке 8.

```
>A stem-loop structure (with a bulge)
GGGCAAUCCUCUUCGGGCCC
((((...((.....))))))
```

Рисунок 8 – Файл в формате DBN

Квадратная матрица взаимодействий

Представляет собой квадратную матрицу, в которой на пересечении i-ой строки и j-ого столбца стоит 0, если i-ый и j-ый нуклеотид в последовательности неспаренные, и 1, если спаренные. Пример такой матрицы представлен на рисунке 9.

	A	A	G	U	A	A	C	U	U
A				1				1	1
A				1				1	1
G							1		
U	1	1			1	1			
A				1				1	1
A				1				1	1
C			1						
U	1	1			1	1			
U	1	1			1	1			

Рисунок 9 – Квадратная матрица взаимодействий

2.4.2 RNAfold

RNAfold – подпрограмма из библиотеки The ViennaRNA Package на языке C, разработанная в институте теоретической химии на базе Венского университета [8]. Для каждой подпрограммы данной библиотеки имеется возможность скачать предварительно скомпилированный двоичный файл (для дистрибутивов Linux) или исполняемый файл (для Windows и MacOS), содержащий необходимую подпрограмму. Таким образом, существует два способа запуска RNAfold:

1. Написание программы на C с подключением библиотеки The ViennaRNA Package.
2. Запуск исполняемого файла через командную строку.

RNAfold реализует один из алгоритмов предсказания вторичной структуры РНК по первичной – алгоритм Цукера. Алгоритм Цукера основывается на предположении, что вторичная структура РНК в состоянии равновесия обладает минимальной свободной энергией. Благодаря использованию методов динамического программирования, алгоритм является наиболее распространенным при решении задач предсказания вторичной структуры РНК. Он обладает высокой скоростью вычислений (временная сложность алгоритма составляет $O(n^3)$, где n – число последовательностей первичных структур РНК) и для использования алгоритма фактически отсутствуют ограничения по длине молекулы.

На вход RNAfold принимает последовательность первичной структуры РНК либо в текстовом формате, либо в специализированном формате для нуклеотидных последовательностей – формате FASTA. На выход RNAfold выдает последовательность вторичной структуры РНК в формате DBN (Dot-Bracket Notation) и свободную энергию этой последовательности в ккал/моль.

2.4.3 Clustal Omega

Иногда анализируют не единичную структуру РНК, а множество схожих между собой последовательностей. С целью установления структурных, функциональных и эволюционных отношений между последовательностями первичных структур РНК применяют метод выравнивания последовательностей. Последовательности записывают в виде строк матрицы, причем в некоторых последовательностях добавляются разрывы цепочки, что позволяет расположить все последовательности таким образом, чтобы схожие участки находились друг под другом.

Одной из самых широко используемых инструментов для множественного выравнивания последовательностей является Clustal (Cluster alignment). Последняя версия программы, Clustal Omega, является одной из наиболее эффективных программ (временная сложность O(N*log(N)), где n – число последовательностей первичных структур РНК) [10].

Clustal Omega поставляется как исполняемый файл (Linux, Windows, MacOS) для запуска через командную строку. На вход Clustal Omega принимает список первичных последовательностей РНК в следующих форматах: Fasta, Clustal, Msf, Phylip, Selex, Stockholm, Vienna. На выход Clustal Omega выдает список выровненных первичных последовательностей РНК. Список поддерживаемых форматов выходного файла совпадает со списком форматов входного файла.

2.4.4 RNAalifold

RNAalifold так же, как и RNAfold, является подпрограммой библиотеки The ViennaRNA Package. RNAalifold также является алгоритмом предсказания вторичной структуры РНК, использующий минимизацию свободной энергии.

Однако в отличие от RNAfold, RNAalifold позволяет построить вторичную структуру РНК, являющуюся общей для некоторого набора первичных последовательностей.

RNAalifold так же, как и RNAfold, имеет два варианта запуска: через подключение библиотеки и через командную строку.

На вход RNAalifold принимает список выровненных последовательностей первичных структур РНК в следующих форматах: CLUSTAL, Stockholm, FASTA, MAF. На выход RNAalifold выдает общую последовательность вторичной структуры РНК в формате DBN (Dot-Bracket Notation) и свободную энергию этой последовательности в ккал/моль.

2.4.5 Выводы

В таблице 2 приведены результаты анализа библиотечных подпрограмм. Видно, что выбранный перечень библиотечных подпрограмм покрывает требуемую функциональность. Также анализ показал, что все три библиотечные подпрограммы могут быть установлены на одной операционной системе и работать через командную строку. Также видно, что все подпрограммы поддерживают работу с форматом FASTA.

Таким образом, все рассмотренные библиотечные подпрограммы соответствуют сформулированным в п. 2.3 требованиям и могут быть включены в базу активных знаний.

Таблица 2 – Результат анализа библиотечных подпрограмм

	RNAfold	Clustal Omega	RNAalifold
Функциональность	Предсказание ВС РНК	Множественное выравнивание	Предсказание общей ВС РНК
Поддерживаемые ОС	Linux, Windows, MacOS	Linux, Windows, MacOS	Linux, Windows, MacOS
Варианты запуска	Подключение библиотеки/командная строка	Командная строка	Подключение библиотеки/командная строка

Продолжение таблицы 2

Формат входных данных	FASTA	FASTA, CLUSTAL, MSF, Phylip, Selex, Stockholm, Vienna	CLUSTAL, Stockholm, FASTA, MAF
Формат выходных данных	DBN	FASTA, CLUSTAL, MSF, Phylip, Selex, Stockholm, Vienna	DBN

2.5 Анализ нефункциональных свойств программ анализа вторичной структуры РНК

Нефункциональные свойства можно разделить на две категории: с точки зрения биологии и с технической точки зрения. С точки зрения биологии, например, два модуля могут обладать одинаковой функциональностью, однако один модуль будет давать более точные результаты при определенном типе входных данных, а другой – при другом.

С технической точки зрения, модули могут обладать следующими нефункциональными свойствами: алгоритмическая сложность по времени, точность вычислений, количество нужной памяти, возможность выполнения на графическом ускорителе, возможность параллельного исполнения.

В таблице 3 приведен результат анализа библиотечных подпрограмм, выбранных в качестве модулей для включения в базу активных знаний, на нефункциональные свойства. Видно, что, например, модуль RNAalifold не может применяться параллельно к нескольким последовательностям (т. к. RNAalifold вычисляет общую для всех входных последовательностей вторичную структуру), в то время как модуль RNAfold может независимо вычислять вторичную структуру РНК для разных последовательностей, т. е. может исполняться параллельно (что обеспечивает соответствие требованию, сформулированному в п. 2.3.2).

Таблица 3 – Результат анализа нефункциональных свойств библиотечных подпрограмм

	RNAfold	Clustal Omega	RNAalifold
Временная сложность алгоритма	$O(n^3)$	O(N*log(N)	$O(n^3)$
Возможность параллельного исполнения	+	–	–
Возможность исполнения на GPU	–	–	–

2.6 Разработка вычислительных моделей

В ходе анализа библиотечных подпрограмм анализа вторичной структуры РНК (п. 2.4), были выделены подпрограммы, которые будут включены в вычислительные модели базы активных знаний в качестве реализации операций. Для построения вычислительных моделей необходимо также определить переменные моделей и задать последовательность операций.

Так как решается задача предсказания вторичной структуры РНК по первичной (п. 2.2), переменными моделей будут множество последовательностей первичных структур РНК, множество последовательностей вторичных структур РНК и общая вторичная структура РНК. Так как операции предсказания вторичной структуры РНК на основе одной последовательности и на основе нескольких последовательностей являются независимыми, то вычислительные модели будут иметь две независимые ветки исполнения. А так как операция предсказания вторичной структуры РНК на основе нескольких последовательностей требует предварительного выравнивания последовательностей первичных структур РНК, то перед применением данной операции будет находиться операция множественного

выравнивания, принимающая на вход последовательности первичных структур РНК и выдающая на выход выровненные последовательности.

2.6.1 Вычислительная модель для предсказания вторичной структуры РНК

Построенная вычислительная модель, представленная на рисунке 10, состоит из трех операций: операция предсказания вторичной структуры РНК на основе одной последовательности, множественное выравнивание последовательностей РНК и предсказание общей вторичной структуры РНК на основе нескольких последовательностей.

Рисунок 10 – Вычислительная модель для предсказания вторичной структуры РНК

Входная переменная вычислительной модели – множество первичных структур РНК. Проходя по верхней ветке модели к каждой последовательности из входных данных применяется операция предсказания вторичной структуры РНК. Проходя по нижней ветки модели, ко всем последовательностям применяется операция множественного выравнивания, а затем, к выровненным последовательностям применяется операция предсказания общей вторичной структуры РНК. Таким образом, выходными переменными данной вычислительной модели является множество вторичных структур РНК, соответствующих входным первичным, а также общая для всех первичных структур вторичная структура РНК.

2.6.2 Вычислительная модель для предсказания вторичной структуры РНК с возможностью параллельного исполнения

Как показал проведенный в п. 2.5 анализ нефункциональных свойств, операция предсказания вторичной структуры РНК на основе одной последовательности позволяет разделить входные данные на части и независимо применять модуль к каждой из частей. Такая возможность была реализована в вычислительной модели, представленной на рисунке 11. Две нижние ветки модели совпадают с вычислительной моделью из п. 2.5.1. Проходя по верхней ветке, к входящим последовательностям применяется операция разделения на две равные части, а затем, к каждой части применяется модуль предсказания вторичной структуры РНК. После этого последовательности полученных вторичных структур объединяются в одно выходное значение – вторичные структуры РНК.

Возможность параллельного исполнения модуля предсказания вторичной структуры РНК на основе одной последовательности в данной вычислительной модели будет реализоваться в случае, если система активных знаний имеет поддержку параллельного исполнения независимых веток модели.

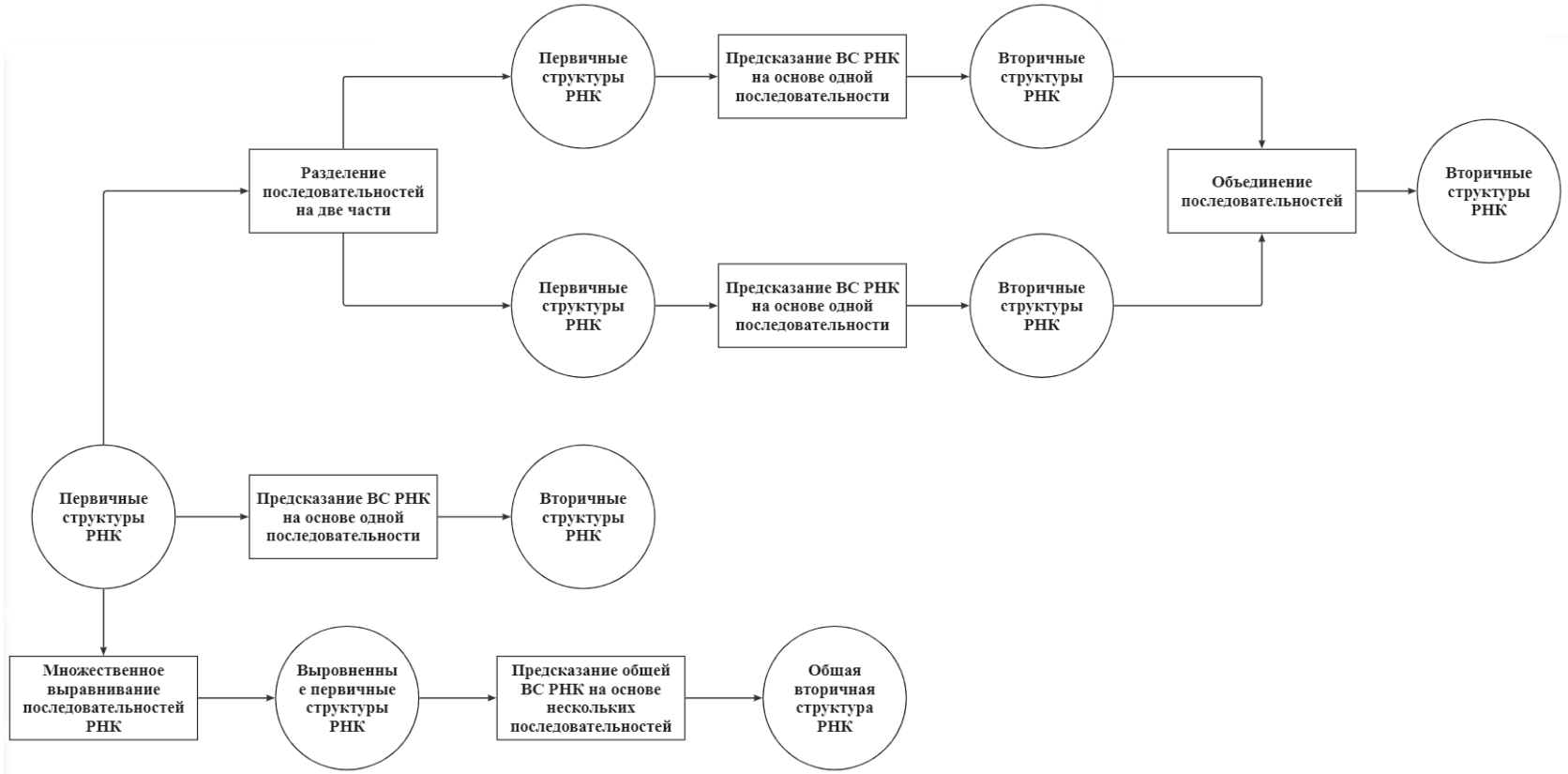

Рисунок 11 – Вычислительная модель для предсказания вторичной структуры РНК с возможностью параллельного исполнения

2.6.3 Рекурсивная вычислительная модель для предсказания вторичной структуры РНК с возможностью параллельного исполнения

В общем случае, количество частей, на которые можно делить входные данные операции предсказания вторичной структуры РНК для последующего параллельного исполнения, может быть больше чем 2. Таким образом, вместо применения операции предсказания к частям входных данных можно применить ту же самую вычислительную модель, которая, в свою очередь, поделит часть входных последовательностей еще на две и так далее. Такая рекурсивная вычислительная модель представлена на рисунке 12.

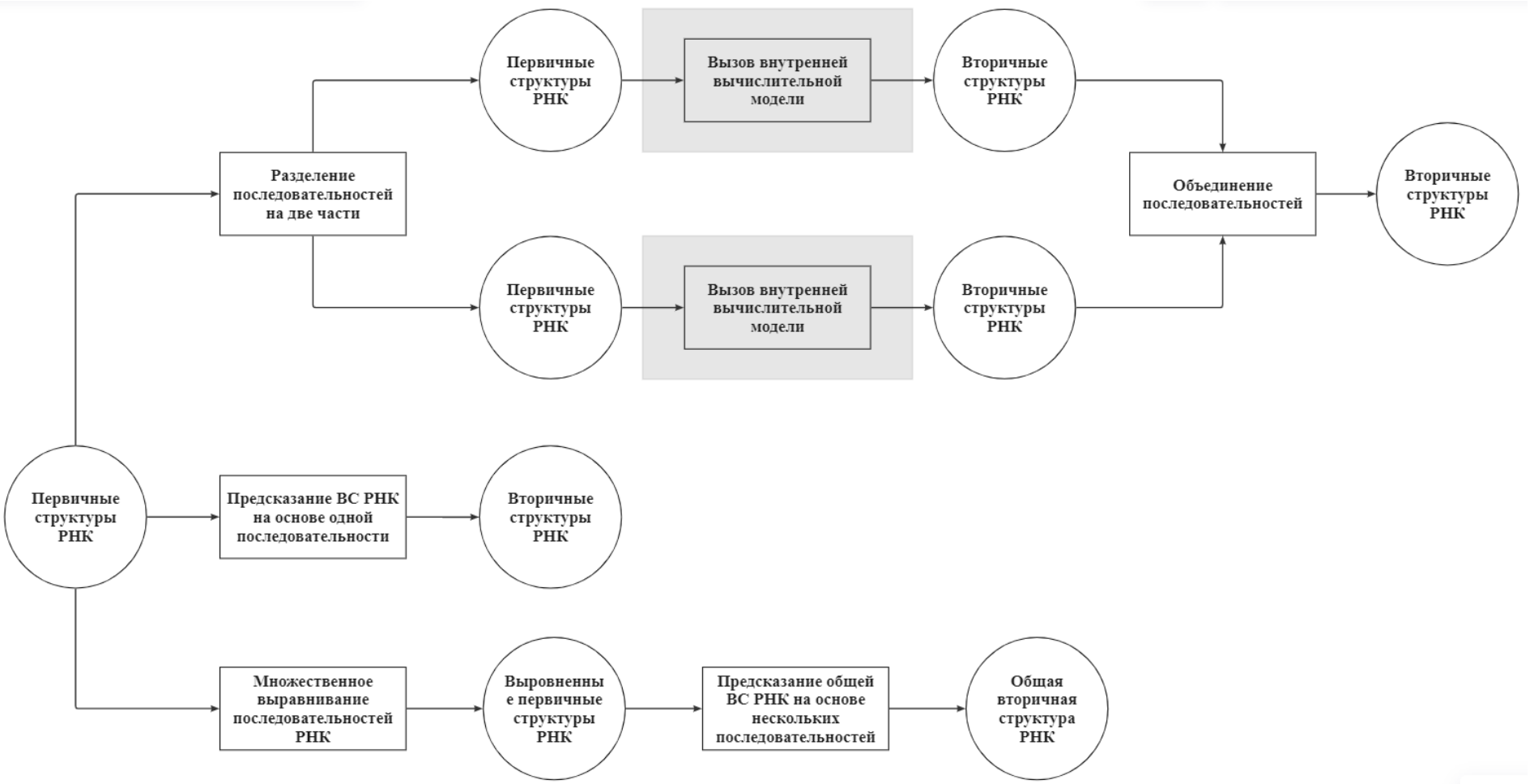

Рисунок 12 – Рекурсивная вычислительная модель для предсказания вторичной структуры РНК с возможностью параллельного исполнения

2.7 Доработка алгоритма построения плана вычислений с целью учета нефункциональных свойств модулей

При вычислении задачи, поставленной на вычислительной модели, система активных знаний строит план вычислений – множество операций вычислительных моделей, применение которых к входным данным позволит вычислить результат. В общем случае, планов вычислений может быть несколько. Например, для вычислительной модели, представленной на рисунке 11, система может построить два альтернативных плана: первый, при котором вторичные структуры РНК будут вычисляться последовательно с применением модуля предсказания вторичной структуры РНК на основе одной последовательности, и второй, при котором входящие последовательности сначала делятся на две части, а затем к каждой применяется модуль предсказания. В текущих алгоритмах построения плана вычислений, при наличии нескольких возможных путей вычисления переменной, системы активных знаний выбирают этот путь произвольно. Для того, чтобы при построении плана вычислений имелась возможность учитывать нефункциональные свойства модулей, реализующих операцию, а также, для того, чтобы система активных знаний строила план вычислений с учетом этих нефункциональных свойств, потребовалось расширить модель описания модуля, включив в него описание нефункциональных свойств и модифицировать алгоритм построения плана вычислений.

За основание был взят алгоритм системы HPC Community Cloud (HPC2C) [21] (данная система более детально рассмотрена в п. 3.1). В листинге 1 представлен алгоритм построения плана вычислений системой HPC2C с внесенным в него изменением (выделено серым цветом).

Исходный алгоритм является модифицированным алгоритмом Дейкстры нахождения кратчайшего пути во взвешенном графе. Для нахождения оптимального пути алгоритм находит путь с минимальной суммой меток вершин графа – операций и переменных. До внесения изменений в алгоритм система присваивала всем операциям и переменным метку, равную 1.

Для того, чтобы при построении плана вычислений система учитывала нефункциональные свойства модулей, была определена функция *Cost*, которая каждой операции вычислительной модели сопоставляет произвольную функцию, принимающую на вход входные переменные операции. Теперь при вычислении метки для операции учитывается значение функции *Cost* для этой операции.

Листинг 1 – Модифицированный алгоритм построения плана вычислений на псевдокоде

```
Вход:
   V:  множество входных переменных;
   W: множество выходных переменных;
   F(v): функция, принимающая переменную v и возвращающая список операций,
         имеющих в списке входных переменных v
Выход:
   P: множество операций, составляющих план вычислений

Алгоритм:
 1   For v ∈ V:          // Всем входным переменным присваиваем метку, равную 0
 2       marks[v] = 0
 3
 4   While W ⊄ V:
 5       nearest = null
 6
 7       For x ∈ marks:     // Ищем ближайшую непосещенную переменную
 8           If x ∈ visited:
 9               continue
10           If nearest = null OR marks[nearest] > marks[x]:
11               nearest = x
12
13       For operation ∈ F(nearest):        // Проходим по всем операций, среди входных переменных
14           operation_mark = null          // которых есть найденная переменная
15
16           For input ∈ In(operation):      //Для всех входных переменных операции
17               If input ∉ marks:           // обновляем метку
18                   break
19               If operation_mark = null OR marks[input] > operation_mark:
20                   operation_mark = marks[input]
21
22               operation_mark = operation_mark + Cost[operation]          // К метке прибавляем
23                                                                          значение функции Cost
24
25           For output ∈ Out(operation):            // Для всех выходных переменных операции
26               If output ∈ visited:                // обновляем метку и наилучшую операцию
27                   continue
28               If output ∉ marks OR marks[output] > operation_mark:
29                   marks[output] = operation_mark
30                   best_operation[output] = operation
31
32       visited[nearest] = true          // Помечаем переменную как посещенную
33
34   queue.push(W)                        // Записываем все выходные переменные в очередь
35
36   While queue.length > 0:
37       w = queue.pop()                              // Достаем переменную из очереди
38       w_best_operation = best_operation[w]         // и добавляем в план наилучшую для
39                                                    // вычисления переменной операцию
40       plan.add(best_operation[w])
41
42       For v ∈ In(w_best_operation):     // Добавляем в очередь входные переменные операции
43           queue.push(v)
44
45   Return plan
```

Для реализации функции *Cost* было расширена модель представления модуля в HPC2C. Теперь при описании модуля, помимо входных и выходных переменных, пользователь может задать свойство *cost*, описывающее произвольную функцию от входных переменных модуля. На основании этого свойства система активных знаний осуществляет выбор того или иного плана вычислений. Таким образом, пользователь получает возможность задать произвольные условия, описывающие, в каких ситуациях использование того или иного модуля будет более оптимальным.

Так, с помощью свойства *cost*, в модели, представленной на рисунке 13, можно задать условия, при которых вычисление вторичной структуры РНК будет производиться последовательно или параллельно в зависимости от количества входных последовательностей первичных структур.

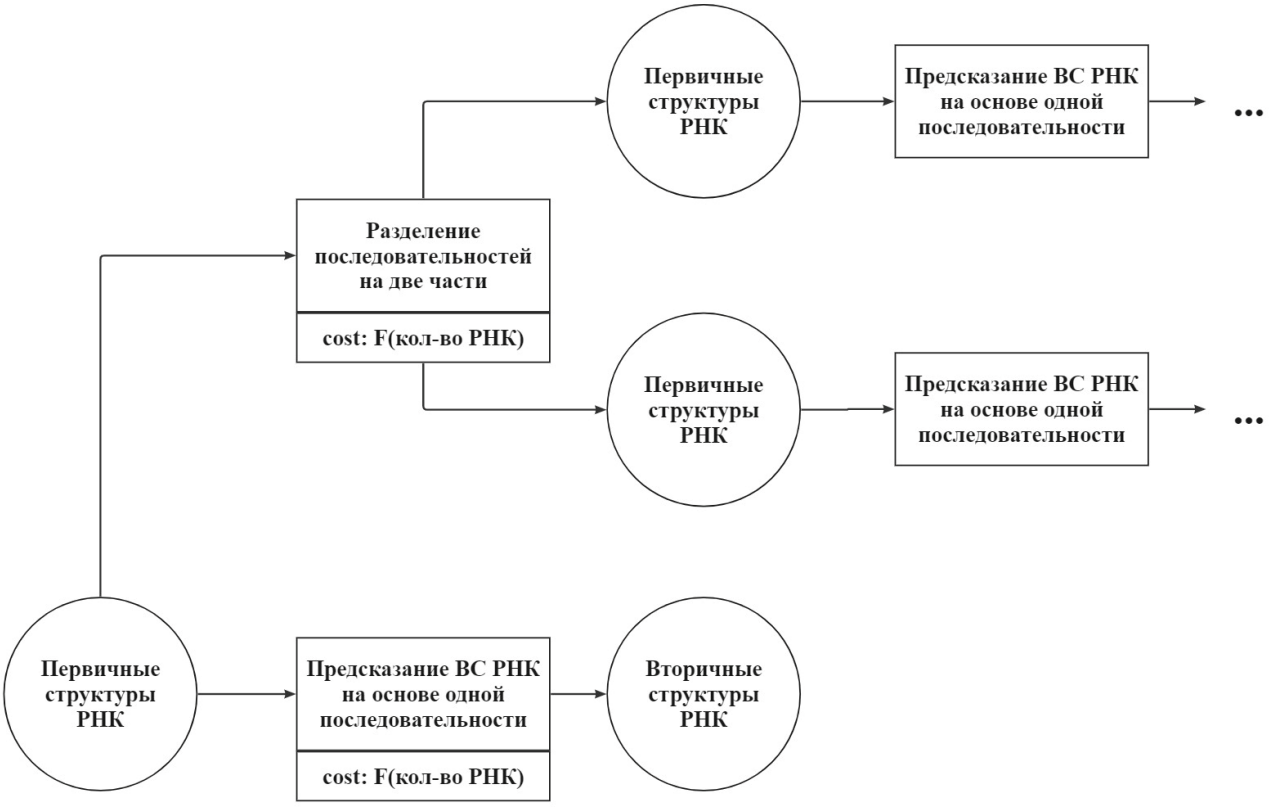

Рисунок 13 – Вычислительная модель, учитывающая нефункциональные свойства модулей

2.8 Характеристика решения

Проведенный в п. 2.4 анализ библиотечных подпрограмм показал, что выбранные для включения в базу активных знаний подпрограммы соответствуют требованиям, сформулированным в п. 2.3.1.

Разработанные в п. 2.6 модели обладают вариативностью исполнения и позволяют конструировать параллельную программу за счет параллельного применения модуля RNAfold, а предложенная в п. 2.7 доработка алгоритма построения плана вычислений позволит системе активных знаний делать выбор в пользу того или иного плана вычисления, основываясь на нефункциональных свойствах модулей. Таким образом, вычислительные модели, разработанные в п. 2.6, соответствуют требованиям, сформулированным в п. 2.3.2.

3 Реализация и тестирование базы активных знаний

3.1 HPC Community Cloud

В качестве системы активных знаний была выбрана подсистема Aka системы HPC Community Cloud [21], так как она позволяет конструировать вычислительные модели и реализует алгоритм построения плана вычислений по заданной вычислительной модели. Преимуществом данной системы является также то, что она является системой с открытым исходным кодом, что позволяет дорабатывать ее алгоритмы.

Данная подсистема состоит из двух модулей, которые в совокупности выполняют две основных функции:

1. Позволяет конструировать вычислительные модели (HPC2C Aka).
2. По вычислительной модели конструирует план вычислений и исполняет его (HPC2C Server).

Пример интерфейса и вычислительной модели, созданной в Aka, представлен на рисунке 14.

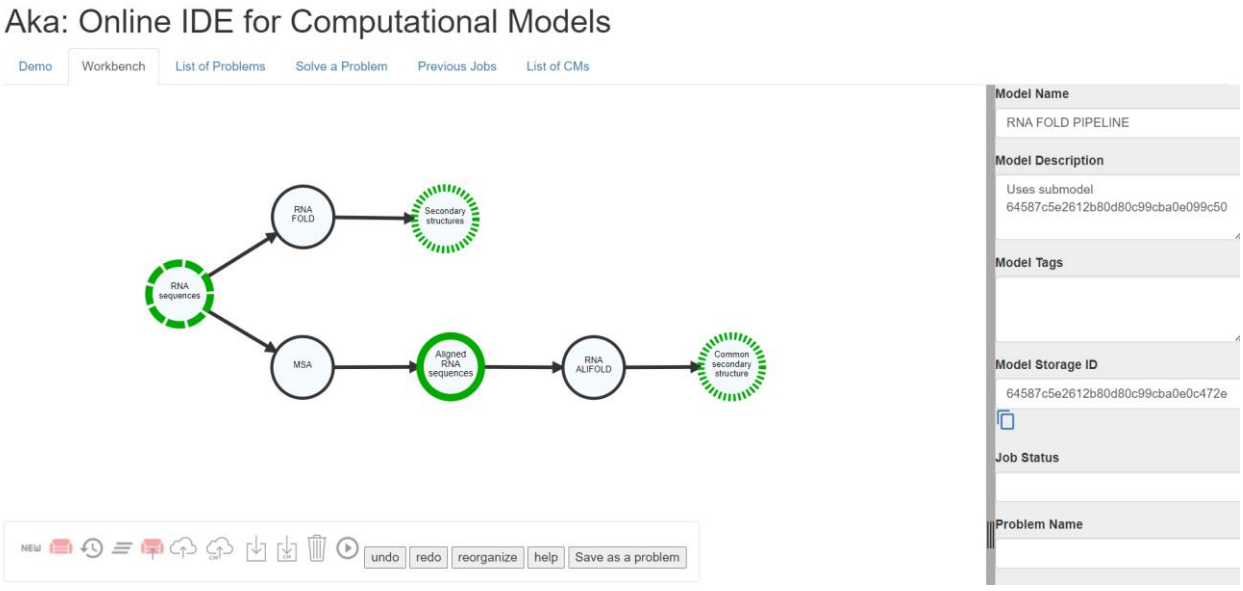

Рисунок 14 – Интерфейс HPC2C Aka

Операции вычислительной модели представляются черными кругами, переменные – зелеными. Входные и выходные переменные модели выделяются пунктиром и могут быть изменены пользователем.

Как было отмечено в п. 1.2.5, модули являются реализацией операции вычислительной модели, т. е. одной операции могут соответствовать несколько

модулей. Эти модули могут различаться по реализации и могут быть представлены разном виде (в виде программы на том или ином языке программирования, в виде HTTP-запроса и т. д.). Поэтому, для того чтобы система активных знаний могла исполнять модули, ей необходима дополнительная информация о типе модуля.

В описание модуля в HPC2C Aka входит название модуля, тип модуля и специфические поля, зависящие от типа модуля. Всего в HPC2C на данный момент поддерживается 8 типов модулей:

1. команда ОС Linux;
2. программа на JavaScript;
3. программа на Python;
4. вызов другой вычислительной модели;
5. HTTP-метод GET;
6. HTTP-метод POST;
7. HTTP-метод PUT;
8. HTTP-метод DELETE.

На рисунке 15 описывается модуль для сложения двух переменных. Тип модуля – javascript, поэтому в описание модуля добавлено еще одно поле – javascript, в котором описывается программа на языке JavaScript.

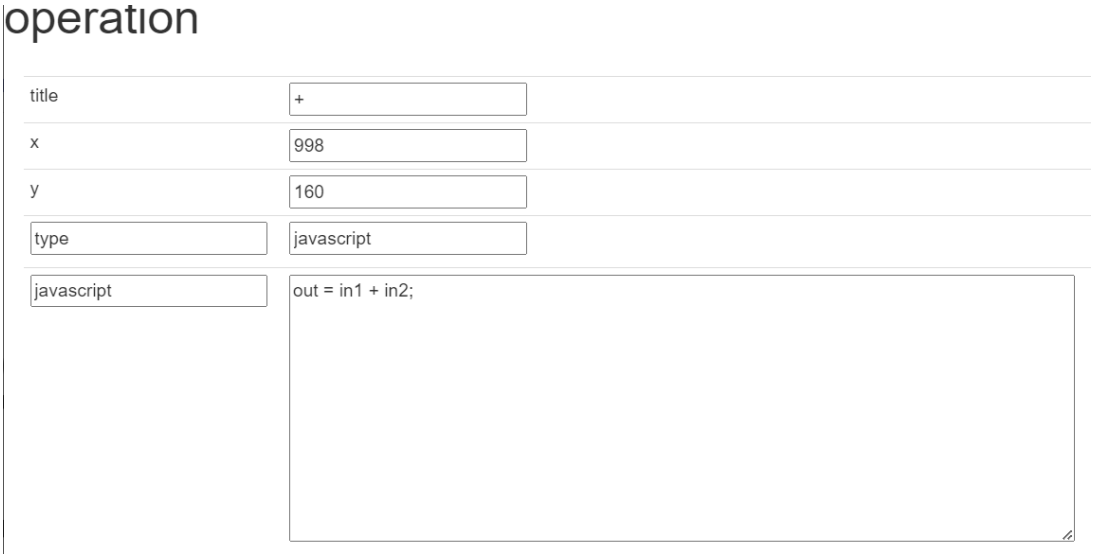

Рисунок 15 – Описание модуля в HPC2C Aka

HPC2C Aka конвертирует созданный пользователем граф вычислительной модели в формат json для дальнейшей передачи на сервер HPC2C. Такой же формат может использоваться пользователями для работы без графического интерфейса, напрямую через API HPC2C. Пример вычислительной модели в формате json представлен на рисунке 16.

```
{
    "variables": {
        "name": {},
        "greeting": {}
    },
    "operations": {
        "generate_greeting": {
            "inputs": ["name"],
            "outputs": ["greeting"],
            "type": "javascript",
            "javascript": "greeting='Hello, '+name+'!';",
            "mapping": {
                "name": "name",
                "greeting": "greeting"
            }
        }
    }
}
```

Рисунок 16 – Пример простейшей вычислительной модели в формате json

По ключу *variables* находится полный список переменных модели. По ключу *operations* находится описание всех операций. В описание операции входит список входных и переменных, тип операции и специфические для типа поля.

3.2 Доработка алгоритма построения плана вычисления HPC2C Aka

Для того, чтобы при конструировании вычислительных моделей пользователь имел возможность задавать нефункциональные свойства модулей, которые в дальнейшем учитывались бы системой активных знаний при построении плана вычислений, была расширена модель представления модуля в HPC2C, а также произведена модификация алгоритма построения плана вычисления, описанная в п. 2.7.

На рисунке 17 представлен JavaScript-класс, описывающий модуль в HPC2C. До расширения модели у класса было два свойства: свойство *inputs*, содержащее информацию о входных переменных модуля и свойство *outputs*,

содержащее информацию о выходных переменных модуля. После внесения изменений в класс было добавлено свойство *cost*.

```
operation = {
  inputs: ["a", "b"],
  outputs: ["c"]
};
```

```
operation = {
  inputs: ["a", "b"],
  outputs: ["c"],
  cost: "10 * (a + b)"
};
```

Рисунок 17 – Внутреннее представление вычислительного модуля в HPC2C до (слева) и после (справа) внесения изменений

Чтобы задать свойство *cost* у модуля в графическом интерфейсе HPC2C Aka пользователь должен добавить поле *cost* в окне описания модуля и вписать в него произвольное выражение на языке JavaScript, использующую в качестве переменных входные переменные описываемого модуля, как показано на рисунке 18.

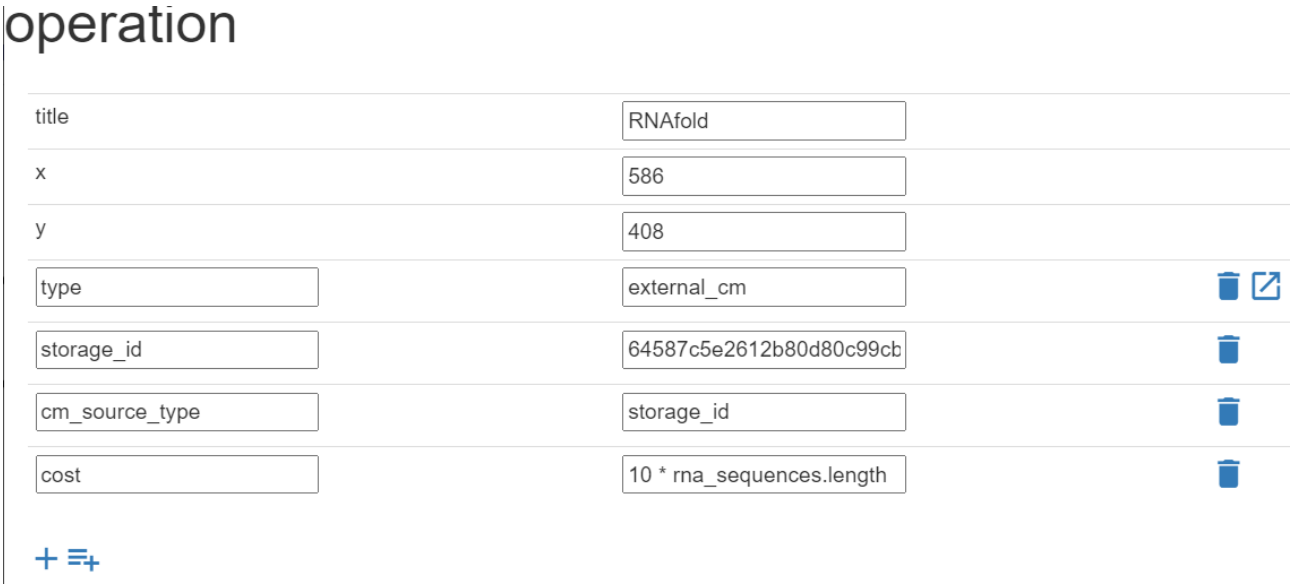

Рисунок 18 – Описание модуля RNAfold со свойством *cost*

Стоит отметить, что при построении плана вычислений системой HPC2C используется словарь с набором пар переменная/значение переменной, с помощью которого также вычисляется и значение JavaScript-выражения *cost*.

3.3 Разработка архитектуры решения

Так как HPC2C поддерживает HTTP-методы как тип операций, было решено вынести выбранные для включения в базу активных знаний модули на

отдельный сервер и предоставить API для вызова исполнения того или иного модуля. Такое решение обладает следующими преимуществами:

1. Расширяемость: при необходимости можно легко добавить еще один модуль.

2. Упрощение создания новых операций в HPC2C Aka: описания операций будут иметь единообразный вид, отличаться будет только url, по которому будет отправляться HTTP-запрос.

Архитектура решения представлена на рисунке 19 и будет содержать 3 модуля:

1. HPC2C Aka: графический интерфейс, где пользователь будет выбирать вычислительные модели для исполнения (или создавать свои), а также вводить входные данные и получать выходные.

2. HPC2C Server: сервер, который принимает описание вычислительной модели от HPC2C Aka, строит план вычислений и ставит модули на исполнение (в данном случае, отправляет запрос на сервер с модулями).

3. Сервер с модулями: сервер, который обрабатывает запросы на исполнение того или иного модуля от HPC2C Server и непосредственно исполняет модуль.

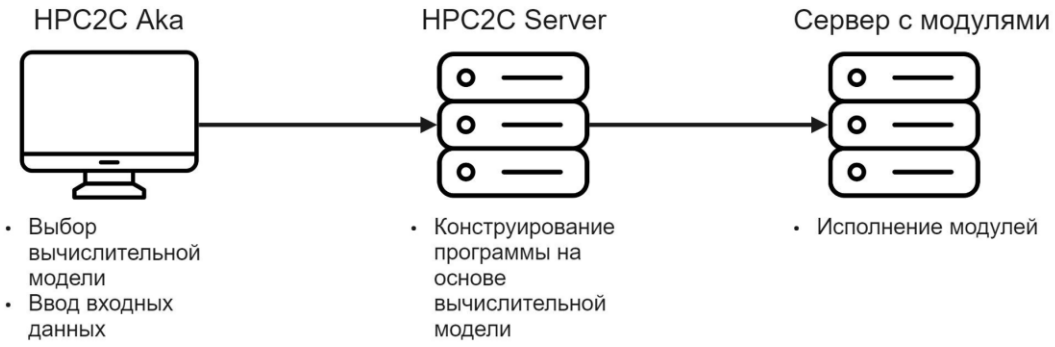

Рисунок 19 – Архитектура решения

3.4 Реализация сервера с модулями

Для создания единой точки доступа к программным модулям базы активных знаний был создан веб-сервер на языке JavaScript с использованием платформы Node.js.

Реализован следующий API:

1. POST /rnafold – запрос на предсказание множества ВС РНК модулем RNAfold. В теле запроса принимает множество последовательностей первичных РНК, в теле ответа присылает множество последовательностей вторичных РНК.
2. POST /rnaalifold – запрос на предсказание общей ВС РНК модулем RNAalifold. В теле запроса принимает множество выровненных последовательностей первичных РНК, в теле ответа присылает общую последовательностей вторичных РНК.
3. POST /clustalo – запрос на выравнивание множества первичных последовательностей РНК модулем Clustal Omega. В теле запроса принимает множество последовательностей первичных РНК, в теле ответа присылает выровненных последовательностей первичных РНК.

Примеры HTTP-запросов приведены в Приложении А.

Также на сервере реализовано логирование, что позволяет отслеживать, какие запросы были получены сервером с отметкой времени получения запроса. На рисунке 20 представлена запись об обработке запроса /rnafold в лог-файле.

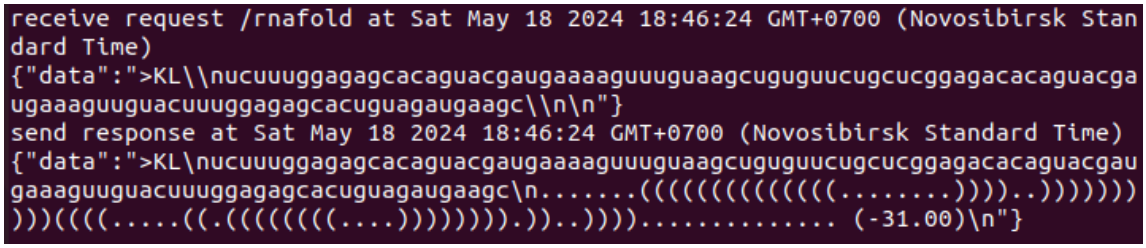

Рисунок 20 – лог-файл сервера с модулями

Описание программы и руководство оператора для сервера с модулями представлено в Приложении Г и Приложении Д соответственно.

3.5 Реализация и тестирование вычислительных моделей на базе HPC2C Aka

В данном разделе представлены реализации вычислительных моделей, разработанных в п. 2.6 на базе HPC2C Aka. Приведены результаты функционального тестирования и тестирования производительности моделей.

3.5.1 Вычислительные модели для запроса на сервер с модулями

Вычислительные модели, описанные в п. 2.6, используют одни и те же операции: операция предсказания вторичной структуры РНК на основе одной последовательности, операция предсказания вторичной структуры РНК на основе нескольких последовательностей, операция множественного выравнивания последовательностей. При реализации данных операций на базе HPC2C Aka эти операции будут являться модулями, отправляющими HTTP-запрос на сервер, описанный в п. 3.3. Для того, чтобы в каждой из вычислительных моделей не описывать одни и тот же модуль запроса на сервер, было решено вынести модули HTTP-запросов на сервер в отдельные вычислительные модели. Это позволит скрыть технические детали реализации от пользователя, что упростит восприятие вычислительных моделей, а также упростит конструирование вычислительных моделей из п. 2.6 за счет переиспользования.

Таким образом, было создано три вычислительные модели: *RNAfold POST request* (для запроса на исполнение модуля RNAfold), *RNAalifold POST request* (для запроса на исполнение модуля RNAalifold), *Clustal Omega POST request* (для запроса на исполнение модуля Clustal Omega). Вычислительная модель *RNAfold POST request* представлена на рисунке 21.

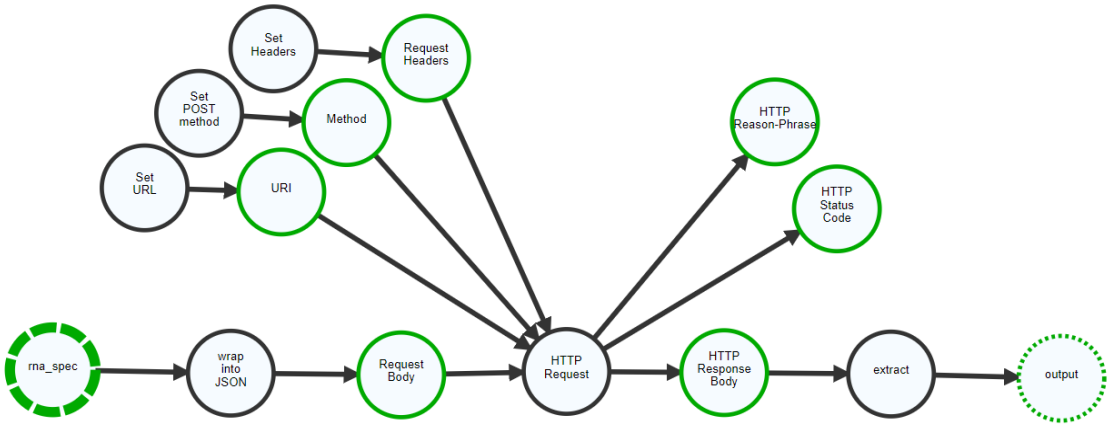

Рисунок 21 – Вычислительная модель *RNAfold POST request*

На вход вычислительной модели подается список последовательностей РНК. Далее этот список преобразуется в тело HTTP-запроса и производится POST-запрос на заданный в модели URL. На выход модель выдает ответ, полученный от сервера.

Описание модели *RNAfold POST request* в формате json представлено в Приложении Б.

3.5.2 Вычислительная модель Sequential RNA prediction model

На рисунке 22 представлена вычислительная модель *Sequential RNA prediction model*, реализующая последовательное предсказание вторичной структуры РНК двумя методами: на основе одной последовательности с помощью модуля RNAfold и на основе нескольких последовательностей с помощью модуля RNAalifold с предварительным выравниванием последовательностей (модуль Clustal Omega). Представление модели *Sequential RNA prediction model* во внутреннем формате HPC2C приведено в Приложении В.

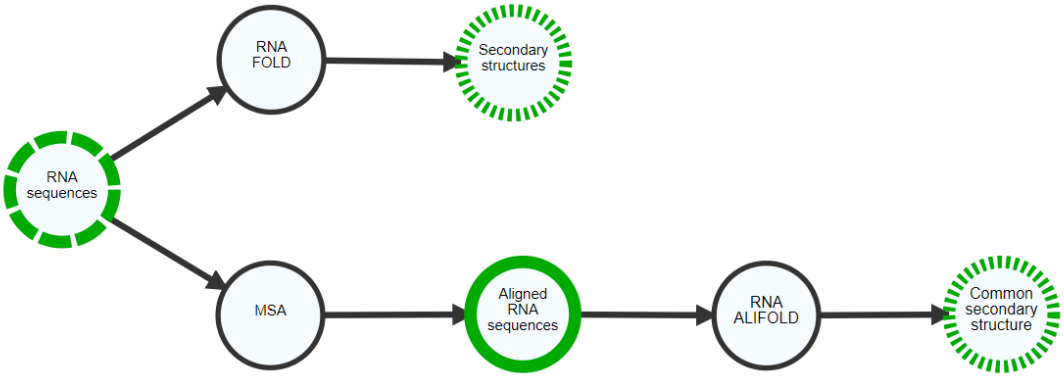

Рисунок 22 – Вычислительная модель *Sequential RNA prediction model*

Функциональное тестирование данной модели производилось на тестовом наборе гомологичных последовательностей из базы некодирующих РНК Rfam [32], для которых уже известны их вторичные структуры. На рисунке 23 представлена часть последовательностей из тестового набора.

```
>KL402727.1:c195254-185141
ucuuuggagagcacaguacgaugaaaguuguaagcuguguuc
>CM001220.2:482268-492375
ucuuuggagagcacaguacgaugaaaguuguaagcuguguuc
>JN005831.1:84928-95041
ucuuuggagagcacaguacgaugaaaguuguaagcuguguuc
```

Рисунок 23 – Фрагмент тестового набора последовательностей РНК в формате FASTA

Полученные выходные данные (вторичные структуры РНК, рисунок 24; общая вторичная структура РНК, рисунок 25) совпали с ожидаемыми эталонными.

```
>KL402727.1:c195254-185141
ucuuuggagagcacaguacgaugaaaguuguaagcuguguuc
........(((((((((((((....)))))..)))))))))
>CM001220.2:482268-492375
ucuuuggagagcacaguacgaugaaaguuguaagcuguguuc
........(((((((((((((....)))))..)))))))))
>JN005831.1:84928-95041
ucuuuggagagcacaguacgaugaaaguuguaagcuguguuc
```

Рисунок 24 – Фрагмент предсказанных моделью *Sequential RNA prediction model* вторичных структур РНК в формате DBN

```
UCUUUGGAGAGCACAGUACGAUGAAAGUUGUAAGCUGUGUUC
((....))(((((((((((((....)))))..)))))))))
```

Рисунок 25 – Предсказанная моделью *Sequential RNA prediction model* общая вторичная структура РНК в формате DBN

3.5.3 Вычислительная модель Parallel RNA prediction model

На рисунке 26 представлена вычислительная модель *Parallel RNA prediction model*, позволяющая применять модуль RNAfold параллельно. Перед

применением модуля RNAfold, входные последовательности делятся на две части в модуле split sequences, затем к каждой из частей применяется модуль RNAfold, а после этого выходные данные обоих модулей объединяются в общую выходную переменную Secondary structures, проходя через операцию объединения concat sequences.

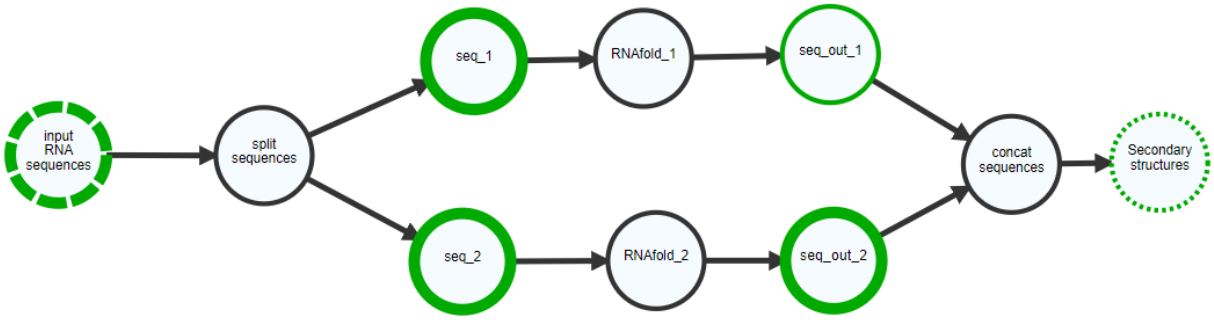

Рисунок 26 – Вычислительная модель *Parallel RNA prediction model*

Корректность работы данной модели была также проверена на реальных данных.

Также было проведено тестирование производительности модели с параллельным исполнением в сравнении с последовательным исполнением модели *Sequential RNA prediction model*. На рисунке 27 представлен график зависимости времени вычисления вторичных структур от количества входных последовательностей для параллельного и последовательного исполнения. На рисунках 28 и 29 представлены графики ускорения и эффективности параллельного исполнения. Ускорение рассчитывалось по формуле (1). Эффективность рассчитывалась по формуле (2).

$$S = \frac{T(1)}{T(n)}, \qquad (1)$$

где T(1) – время работы последовательной программы, T(n) – время работы параллельной программы на n потоках.

$$E = \frac{S(n)}{n} \times 100, \qquad (2)$$

где S(n) – ускорение параллельной программы на n потоках, n – число потоков.

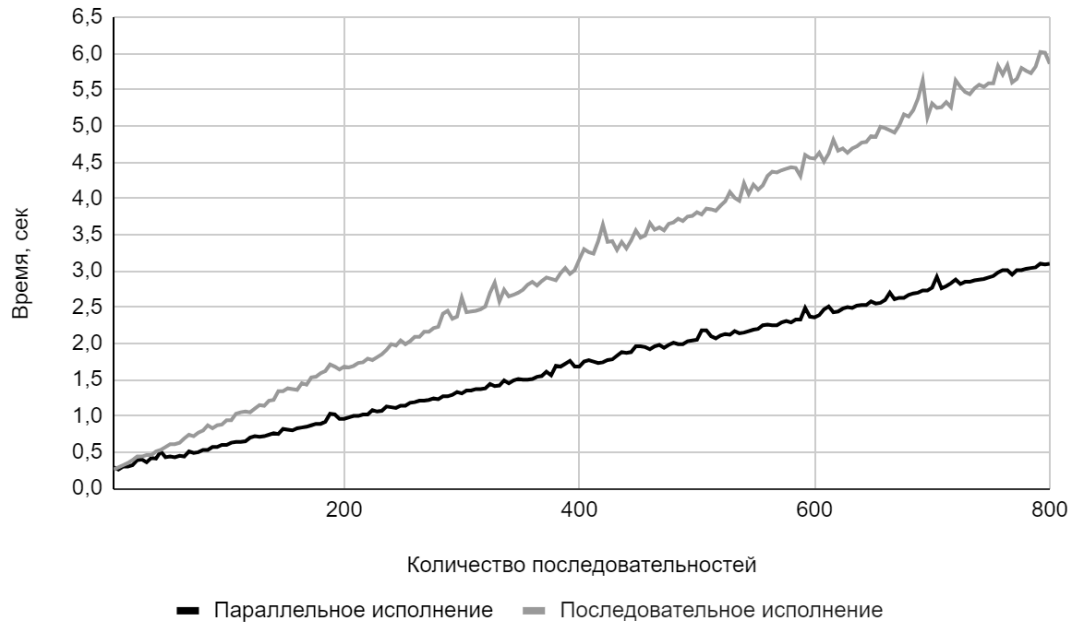

Рисунок 27 – Зависимость времени работы программы от количества последовательностей

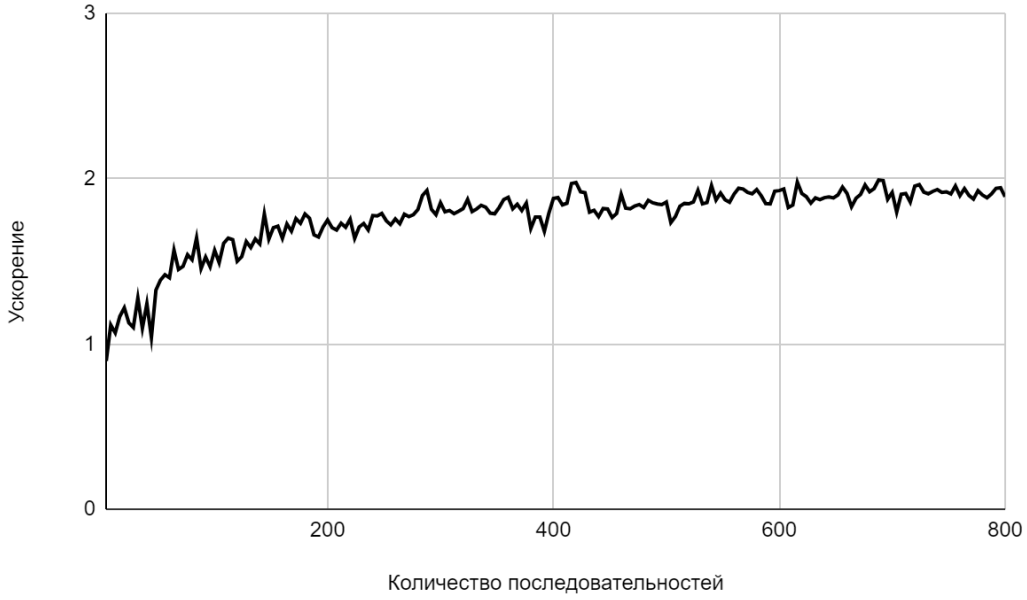

Рисунок 28 – Зависимость ускорения параллельной программы от количества последовательностей

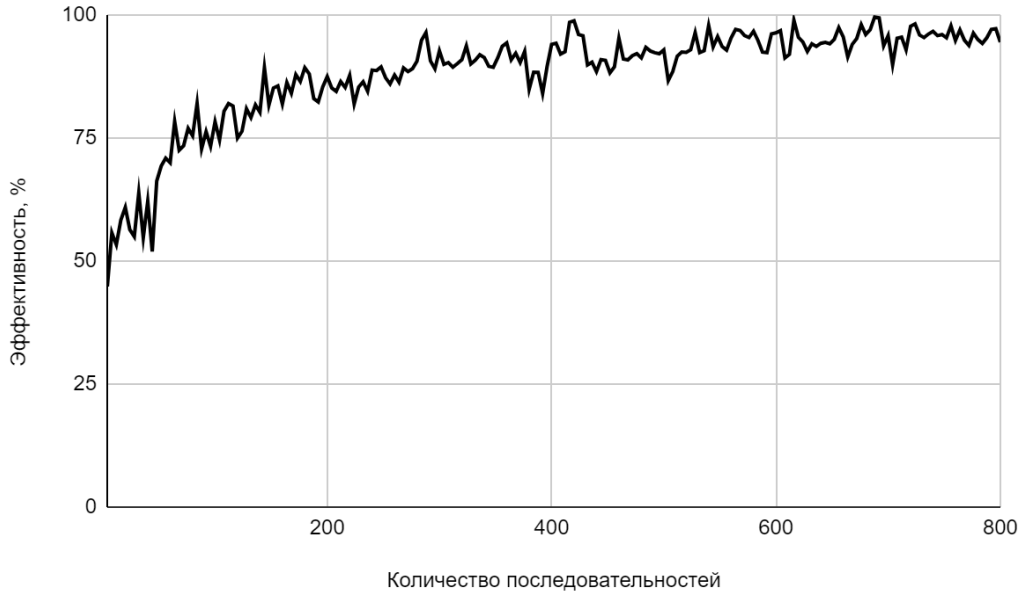

Рисунок 29 – Зависимость эффективности параллельной программы от количества последовательностей

Видно, что параллельное исполнение становится эффективным уже на количестве входных последовательностей, равном 8, а при количестве последовательностей 300 эффективность параллельной программы достигает 90 % и выше.

3.5.4 Вычислительная модель RNA prediction model with performance variation

На рисунке 30 представлена модель, поддерживающая как последовательное, так и параллельное применение модуля RNAfold. Выбор того или иного плана вычисления в данной модели обеспечивается за счет дополнительных операций проверки количества входящих последовательностей (на рисунке 30 – операции «>=2» и «<2»). При количестве последовательностей больше либо равном 2, исполнение идет по верхней ветке, при количестве, меньшем 2 – по нижней.

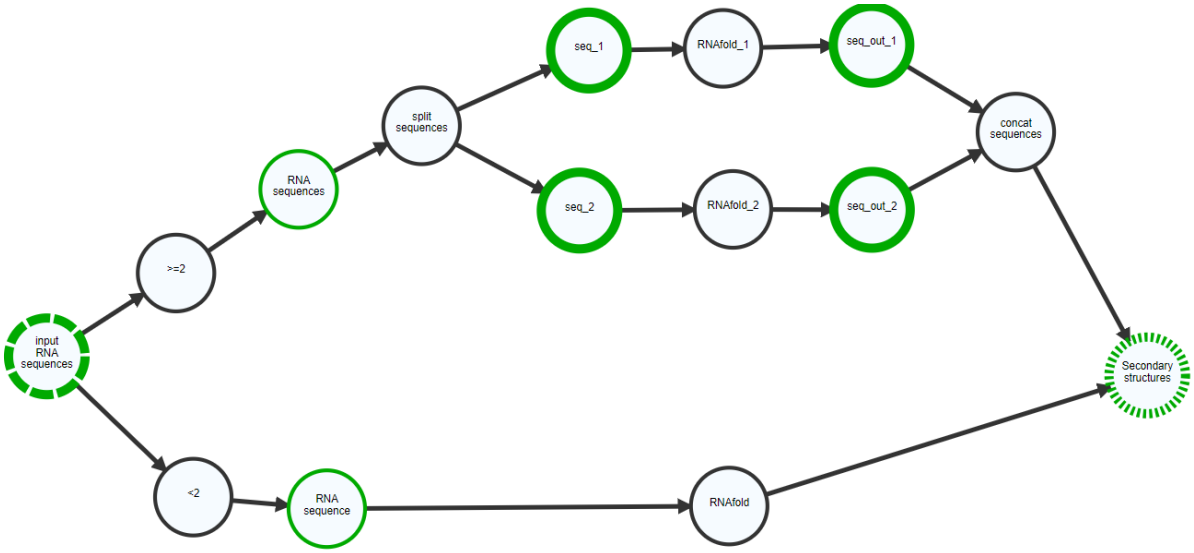

Рисунок 30 – Вычислительная модель *RNA prediction model with performance variation*

Данная модель была также протестирована на реальных данных. Проведенный анализ лог-файлов сервера HPC2C показал корректность выбора той или иной ветки исполнения при количестве последовательностей более двух и менее двух.

3.5.5 Вычислительная модель RNA prediction model with module cost

После доработки алгоритма HPC2C, описанном в п. 2.7 и п. 3.2, была реализована вычислительная модель, представленная на рисунке 31 и использующая при описании модулей поле *cost.*

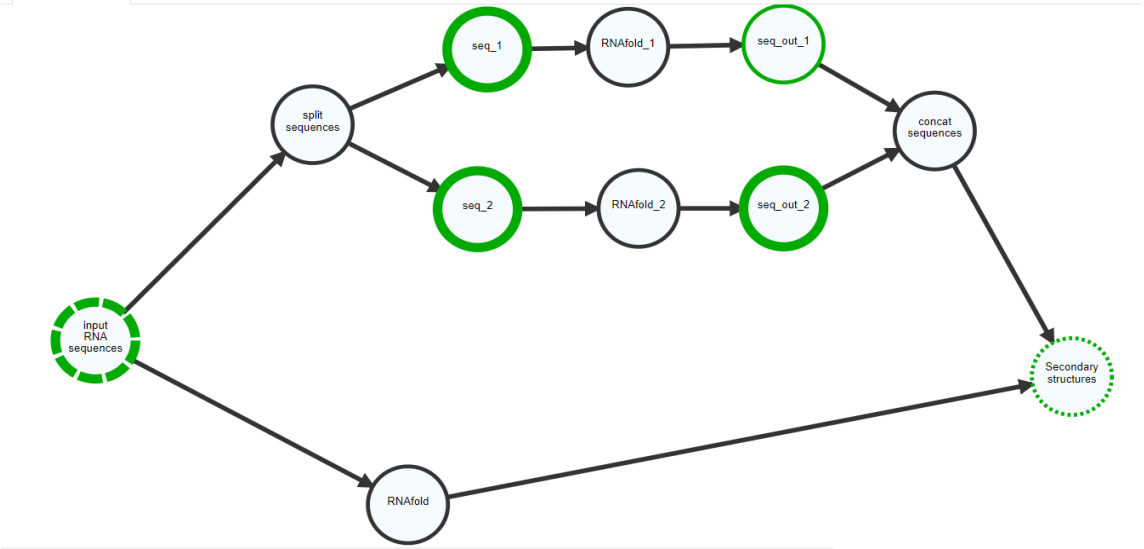

Рисунок 31 – Вычислительная модель *RNA prediction model with module cost*

Описание модулей RNAfold и split sequences в HPC2C Aka с полем *cost* представлены на рисунке 32.

Рисунок 32 – Описание модулей RNAfold и split sequences в HPC2C Aka

Как показало экспериментальное исследование производительности параллельного исполнения модуля RNAfold в п. 3.5.4, параллельное исполнение является эффективным при количестве последовательностей, большем 8. Выражение в поле *cost* модулей RNAfold и split sequences составлено таким образом, что при количестве последовательностей, меньшем 8, исполнение идет по нижней ветке (последовательное исполнение), а при количестве, большем 8, по верхней (параллельное исполнение).

По аналогии с моделью из п. 3.5.4, было проведено функциональное тестирование и тестирование, подтверждающее корректность выбора плана вычислений.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы разработана база активных знаний для автоматического конструирования программ анализа вторичной структуры РНК, учитывающая функциональные и нефункциональные свойства модулей.

Проведен анализ библиотечных программ анализа вторичной структуры РНК на предмет их совместимости, технической возможности их включения в базу активных знаний, на предмет нефункциональных свойств. Разработан способ описания нефункциональных свойств модулей. Разработаны и реализованы вычислительные модели предсказания вторичной структуры РНК на базе HPC Community Cloud: Sequential RNA prediction model, Parallel RNA prediction model, RNA prediction model with performance variation, RNA prediction model with module cost. Расширено внутреннее представление вычислительных модулей и модифицирован алгоритм построения плана вычислений HPC Community Cloud. Проведено функциональное тестирование созданных вычислительных моделей на наборе гомологичных последовательностей из базы некодирующих РНК Rfam. Проведено тестирование производительности модели Parallel RNA prediction model.

Положения, выносимые на защиту:

1. Разработана база активных знаний для автоматического конструирования программ анализа вторичной структуры РНК.

2. Разработан способ описания нефункциональных свойств модулей базы активных знаний.

3. Произведена доработка алгоритма построения плана вычислений системы HPC Community Cloud, позволяющая учитывать нефункциональные свойства модулей.

4. Проведено экспериментальное исследование созданной базы активных знаний.

Направления дальнейшего развития:

1. Расширение базы активных знаний новыми модулями и вычислительными моделями.

2. Учет нефункциональных свойств, специфичных для предметной области.

Выпускная квалификационная работа выполнена мной самостоятельно и с соблюдением правил профессиональной этики. Все использованные в работе материалы и заимствованные принципиальные положения (концепции) из опубликованной научной литературы и других источников имеют ссылки на них. Я несу ответственность за приведенные данные и сделанные выводы.

Я ознакомлена с программой государственной итоговой аттестации, согласно которой обнаружение плагиата, фальсификации данных и ложного цитирования является основанием для не допуска к защите выпускной квалификационной работы и выставления оценки «неудовлетворительно».

Матус Наталья Евгеньевна _____

ФИО студента Подпись студента

«20» мая 2024 г.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Titov, I. Bioinformatic Method for Identifying Group II Introns in Organella Genomes / I. Titov, N. Kobalo, D. Vorobyev, A. Kulikov // Frontiers in Genetics.– 2019.– Vol. 10.– Art. 1135.
2. Kobalo, N. Prediction of RNA Secondary Structure Based on Optimization in the Space of Its Descriptors by the Simulated Annealing Algorithm / N. Kobalo, A. Kulikov, I. Titov // Perspectives of System Informatics. Lecture Notes in Computer Science.– 2019. – Vol. 11964. – P. 116–124.
3. Reich, M. GenePattern 2.0. / M. Reich, T. Liefeld, J. Gould, J. Lerner, P. Tamayo, J. P. Mesirov // Nature Genetics.– 2006.– Vol. 38.– P. 500–501.
4. Goecks, J. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences / J. Goecks, A. Nekrutenko, J. Taylor // Genome Biology.– 2010.– Vol. 11.– Art. 86.
5. Malyshkin, V. Active Knowledge, LuNA and Literacy for Oncoming Centuries / V. Malyshkin // Programming Languages with Applications to Biology and Security. Lecture Notes in Computer Science.– 2015. – Vol. 9465. – P. 292–303.
6. Вальковский, В. А. Синтез параллельных программ и систем на вычислительных моделях / В. А. Вальковский, В. Э. Малышкин.– Издательство "Наука", Сибирское отд-е, Новосибирск, 1988.– 129 с.
7. Cock, P. A. Biopython: freely available Python tools for computational molecular biology and bioinformatics / P. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, M. J. L. de Hoon // Bioinformatics.– 2009.– Vol. 25.– P. 1422–1423.
8. Lorenz, R. Vienna RNA package 2.0. / R. Lorenz, S. H. Bernhart, C. H. zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, I. L. Hofacker // Algorithms for Molecular Biology.– 2011.– Vol. 6.– Art. 26.
9. Nawrocki, E. P. Infernal 1.1: 100-fold faster RNA homology searches / E. P. Nawrocki, S. R. Eddy // Bioinformatics.– 2013.– Vol. 29.– P. 2933–2935.

10. Sievers, F. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega / F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, J. D. Thompson, D. G. Higgins // Molecular Systems Biology.– 2011.– Vol. 7.– Art. 539.
11. Altschul, S. F. Basic local alignment search tool / S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. Lipman // Journal of Molecular Biology.– 1990.– Vol. 2015.– P. 403–410.
12. Gruber, T. R. Toward principles for the design of ontologies used for knowledge sharing / T. R. Gruber // International Journal Human-Computer Studies.–1995.– Vol. 43.– P. 907–928.
13. Подколодный, Н. Л. Онтологии в биоинформатике и системной биологии / Н. Л. Подколодный, О. А. Подколодная // Вавиловский журнал генетики и селекции.– 2015.– № 19.– С. 652–660.
14. Suzi, A. A. The Gene Ontology knowledgebase in 2023 / A. A. Suzi, J. Balhoff, S. Carbon, J. M. Cherry, H. J. Drabkin, D. Ebert, M. Feuermann, P. Gaudet, N. L. Harris // Genetics.– 2023.– Vol. 224. Art. 031.
15. Jackson, P. Introduction To Expert Systems / P. Jackson.– 1998.– Addison Wesley.– p. 560.
16. Chen, X. Planning bioinformatics workflows using an expert system / X. Chen, J. T. Chang // Bioinformatics.– 2017.– Vol. 33.– P. 1210–1215.
17. Lehmann, F. Semantic networks / F. Lehmann // Computers & Mathematics with Applications.– 1992.– Vol. 23.– P. 1–50.
18. Hsing, M. Integration of Biological Data with Semantic Networks / M. Hsing, A. Cherkasov // Current Bioinformatics.– 2006.– Vol. 1. – P. 273–290.
19. Bodenreider, O. The Unified Medical Language System (UMLS): Integrating Biomedical Terminology/ O. Bodenreider // Nucleic acids research.– 2004.– Vol. 32.– P. 267-270.
20. Артюхов, А. А. Прототип базы активных знаний на основе вычислительных моделей / А. А. Артюхов // Проблемы информатики.– 2021.– № 4.– С. 55–66.

21. Gorodnichev, M. Semantic tools for development of high-level interactive applications for supercomputers / M. Gorodnichev, D. Lebedev // The Journal of Supercomputing.– 2021.– Vol. 77.– P. 11866–11880.
22. Luscombe, N. M. What is bioinformatics? A proposed definition and overview of the field / N. M. Luscombe, D. Greenbaum, M. Gerstein // Methods of Information in Medicine.– 2001.– Vol. 40.– P. 346–358.
23. Can, T. Introduction to Bioinformatics / T. Can // miRNomics: MicroRNA Biology and Computational Analysis. Methods in Molecular Biology.– 2014.– Vol. 1107.– P. 51–71.
24. Yamin, D. Current and Future Technologies for the Detection of Antibiotic-Resistant Bacteria / D. Yamin, V. Uskoković, A. M. Wakil, M. D. Goni; S. H. Shamsuddin, F. H. Mustafa, W. A. Alfouzan, M. Alissa, A. Alshengeti, R. H. Almaghrabi // Diagnostics.– 2023.– Vol. 13.– Art. 3246.
25. Abdurakhmonov, I. Y. Bioinformatics: Basics, Development, and Future / I. Y. Abdurakhmonov // Bioinformatics – Updated Features and Applications. 2016.– InTech.– p. 268.
26. Deng, J. RNA structure determination: From 2D to 3D / J. Deng, X. Fang, L. Huang, S. Li, L. Xu, K. Ye, J. Zhang, K. Zhang, Q. C. Zhang // Fundamental Research.– 2023.– Vol. 3.– P. 727–737.
27. Zhao, Q. Review of machine learning methods for RNA secondary structure prediction / Q. Zhao, Z. Zhao, X. Fan, Z. Yuan, Q. Mao, Y. Yao // PLOS Computational Biology.– 2021.– Vol. 17.– Art. 1009291.
28. Ganser, L. R. The roles of structural dynamics in the cellular functions of RNAs / L. R. Ganser, M. L. Kelly, D. Herschlag, H. M. Al-Hashimi // Nature Reviews Molecular Cell Biology.– 2019.– Vol. 20.– P. 474–489.
29. Anfinsen, C. B. Principles that govern the folding of protein chains / C. B. Anfinsen // Science.– 1973.– Vol. 181.– P. 223–230.
30. Шевелев, Е. И. Поиск и анализ паттернов вторичной структуры РНК // Выпускная квалификационная работа.– 2023.– НГУ. Новосибирск.

31. Akalin, A. Computational Genomics with R // A. Akalin.– 2020.– Chapman and Hall/CRC.
32. Kalvari, I. Rfam 14: expanded coverage of metagenomic, viral and microRNA families / I. Kalvari, E. P. Nawrocki, N. Ontiveros-Palacios, J. Argasinska, K. Lamkiewicz, M. Marz, S. Griffiths-Jones, C. Toffano-Nioche, D. Gautheret, Z. Weinberg, E. Rivas, S. R. Eddy, R. D. Finn, A. Bateman, A. I. Petrov // Nucleic Acids Research.– 2021.– Vol. 49.– P. 192–200.

ПРИЛОЖЕНИЕ А

Примеры запросов к серверу с модулями

1. Пример запроса на исполнение модуля RNAfold:

```
POST /rnafold HTTP/1.1
Host: localhost:8000
Content-Type: application/json
Content-Length: 88

{
  "data":">KL4027271:c195254-
185141\\nucuuuggagagcacaguacgaugaaaguuguaagcuguguuc"
}
```

2. Пример запроса на исполнение модуля RNAalifold:

```
POST /rnaalifold HTTP/1.1
Host: localhost:8000
Content-Type: application/json
Content-Length: 161

{
  "data":">KL4027271:c195254-
185141\\nucuuuggagagcacaguacgaugaaaguuguaagcuguguuc\\n>CM001220.2:482268
-492375\\nucuuuggagagcacaguacgaugaaaguuguaagcuguguuc"
}
```

3. Пример запроса на исполнение модуля Clustal Omega:

```
POST /clustalo HTTP/1.1
Host: localhost:8000
Content-Type: application/json
Content-Length: 161

{
  "data":">KL4027271:c195254-
185141\\nucuuuggagagcacaguacgaugaaaguuguaagcuguguuc\\n>CM001220.2:482268
-492375\\nucuuuggagagcacaguacgaugaaaguuguaagcuguguuc"
}
```

ПРИЛОЖЕНИЕ Б

Вычислительная модель для запроса на сервер с модулями в формате json

```
{
    "rnafold": {
        "variables": {
            "URI": {
            },
            "Request Body": {
            },
            "HTTP Status Code": {
            },
            "Method": {
            },
            "HTTP Reason-Phrase": {
            },
            "HTTP Response Body": {
            },
            "Request Headers": {
            },
            "rna_spec": {
                "input": true
            },
            "output": {
                "output": true
            }
        },
        "operations": {
            "HTTP Request": {
                "inputs": [
                    "URI",
                    "Request Body",
                    "Method",
                    "Request Headers"
                ],
                "outputs": [
                    "HTTP Status Code",
```

```
            "HTTP Response Body",
            "HTTP Reason-Phrase"
        ],
        "type": "http_post",
        "mapping": {
            "URI": "url",
            "Request Body": "data",
            "Method": "method",
            "Request Headers": "headers",
            "HTTP Status Code": "response_code",
            "HTTP Response Body": "response_data",
            "HTTP Reason-Phrase": "response_reason"
        }
    },
    "Set POST method": {
        "inputs": [],
        "outputs": [
            "Method"
        ],
        "javascript": "result = \"POST\";",
        "type": "javascript",
        "mapping": {
            "Method": "result"
        }
    },
    "wrap into JSON": {
        "inputs": [
            "rna_spec"
        ],
        "outputs": [
            "Request Body"
        ],
        "javascript": "y = JSON.stringify({data: x});",
        "type": "javascript",
        "mapping": {
            "rna_spec": "x",
            "Request Body": "y"
```

```
        }
    },
    "extract": {
        "inputs": [
            "HTTP Response Body"
        ],
        "outputs": [
            "output"
        ],
        "type": "javascript",
        "javascript": "out = JSON.parse(json).data;",
        "mapping": {
            "HTTP Response Body": "json",
            "output": "out"
        }
    },
    "Set URL": {
        "inputs": [],
        "outputs": [
            "URI"
        ],
        "type": "javascript",
        "javascript": "x='http://192.168.12.24:8000/rnafold';",
        "mapping": {
            "URI": "x"
        }
    },
    "Set Headers": {
        "inputs": [],
        "outputs": [
            "Request Headers"
        ],
        "type": "javascript",
        "javascript": "x='Content-Type: application/json';",
        "mapping": {
            "Request Headers": "x"
        }
```

```
                    }
                }
            }
        }
```

ПРИЛОЖЕНИЕ В

Вычислительная модель Sequential RNA prediction model в формате json


```
{
    "sequential RNA prediction model": {
        "variables": {
            "RNA sequences": {
                "input": true
            },
            "Secondary structures": {
                "output": true
             },
            "Common secondary structure": {
                "output": true
            },
            "Aligned RNA sequences": {
             }
        },
        "operations": {
            "RNA FOLD": {
                "inputs": [
                    "RNA sequences"
                ],
                "outputs": [
                    "Secondary structures"
                ],
                "type": "external_cm",
                "storage_id": "64587c5e2612b80d80c99cba0e099c50",
                "cm_source_type": "storage_id",
                "mapping": {
                    "RNA sequences": "rna_spec",
                    "Secondary structures": "output"
                }
            },
            "RNA ALIFOLD": {
                "inputs": [
                    "Aligned RNA sequences"
                ],
                "outputs": [
                    "Common secondary structure"
                ],
                "type": "external_cm",
                "storage_id": "ec52ae6e7dc1819a3e58c98f71000ead",
                "cm_source_type": "storage_id",
                "mapping": {
```

```
                    "Common secondary structure": "output",
                    "Aligned RNA sequences": "rna_spec"
                }
            },
            "MSA": {
                "inputs": [
                    "RNA sequences"
                ],
                "outputs": [
                    "Aligned RNA sequences"
                ],
                "type": "external_cm",
                "storage_id": "ec52ae6e7dc1819a3e58c98f71020b74",
                "cm_source_type": "storage_id",
                "mapping": {
                    "RNA sequences": "rna_spec",
                    "Aligned RNA sequences": "output"
                }
            }
        }
    }
}
```

ПРИЛОЖЕНИЕ Г

ВЕБ-СЕРВЕР ДЛЯ АНАЛИЗА ВТОРИЧНОЙ СТРУКТУРЫ РНК

Описание программы

Листов 10

Новосибирск, 2024

АННОТАЦИЯ

В данном документе приведено описание веб-сервера для анализа вторичной структуры РНК. Программа написана на языках JavaScript и bash. Программа решает класс задач, связанных с предсказанием вторичной структуры РНК по первичной.

Оформление программного документа «Описание программы» произведено по требованиям ГОСТ 19.402-78 «ЕСПД. Описание программы» и ГОСТ 19.105-78 «Единая система программной документации (ЕСПД). Общие требования к программным документам (с Изменением № 1)».

СОДЕРЖАНИЕ

1. Общие сведения

1.1. Обозначение и наименование программы

Полное наименование программы – «Веб-сервер для анализа вторичной структуры РНК».

Краткое название – «Веб-сервер для анализа ВС РНК».

1.2. Программное обеспечение, необходимое для функционирования программы

Для функционирования программы необходима среда выполнения кода JavaScript Node.js.

1.3. Языки программирования

Программа написана на языках JavaScript и bash с использованием технологий Node.js.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Программа решает следующий класс задач:

1. Предсказание вторичной структуры РНК на основе одной последовательности (применение модуля RNAfold).
2. Предсказание вторичной структуры РНК на основе нескольких последовательностей (применение модуля RNAalifold).
3. Множественное выравнивание последовательностей первичных структур РНК (применение модуля Clustal Omega).

3. Описание логической структуры

3.1. Алгоритм программы

1. Пользователь посылает HTTP-запрос на сервер с входными данными для вычислений в формате json.
2. Сервер обрабатывает запрос и применяет к входным данным необходимый модуль.
3. Сервер посылает в ответ выходные данные в формате json.

3.2. Структура программы

Структура программы:

1. Основной файл веб-сервера app.js, принимающий HTTP-запросы.

2. Три файла rnafold.js, rnaalofold.js, clustalo.js, обрабатывающие запрос и формирующие ответ.

3. Три файла с bash-скриптами rnafold.sh, rnaalofold.sh, clustalo.sh, содержащие непосредственный вызов соответствующих модулей.

4. Исполняемые файлы с модулями RNAfold, RNAalifold, Clustal Omega.

4. Используемые технические средства

Программа эксплуатируется на персональном компьютере или вычислительном кластере.

Устройство, на котором запускается программа, должно иметь:

1. операционную систему Ubuntu версии 20.04 и выше;
2. оперативную память объемом 2 Гб или выше;
3. процессор с количеством ядер 1 или выше;
4. жесткий диск объемом 20 Гб или выше.

5. Вызов и загрузка

Запуск сервера происходит из командной строки. Для запуска сервера необходимо:

1. Перейти в директорию, где расположены файлы сервера.
2. Выполнить команду: nohup node app.js.

6. Входные данные

Входные данные должны представляться через тело POST-запроса в формате json и должны быть представлены в виде строки по ключу “data”, как на рисунке Г.1. Строка должна содержать последовательности РНК в формате FASTA.

Модуль, который необходимо применить к входным данным, указывается в URL.

```
{
    "data": ">example\\ngccugacug"
}
```

Рисунок Г.1 – Пример json с входными данными

7. Выходные данные

Выходными данными является результат применения модуля к входным данным из запроса. Данные представляются в формате json. По ключу “data” находится строка с результатами вычислений (в формате FASTA или DBN, в зависимости от модуля). Пример json с ответом сервера представлен на рисунке Г.2.

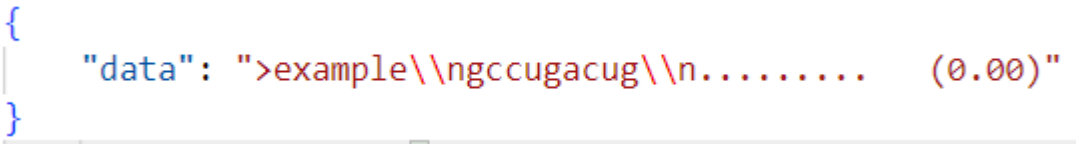

Рисунок Г.2 – Пример json с выходными данными

ПРИЛОЖЕНИЕ Д

ВЕБ-СЕРВЕР ДЛЯ АНАЛИЗА ВТОРИЧНОЙ СТРУКТУРЫ РНК
Руководство оператора
Листов 7

Новосибирск, 2024

АННОТАЦИЯ

В данном документе приведено руководство оператора веб-сервера для анализа вторичной структуры РНК.

Оформление программного документа «Руководство оператора» произведено по требованиям ГОСТ 19.505-79 «ЕСПД. Руководство оператора» и ГОСТ 19.105-78 «Единая система программной документации (ЕСПД). Общие требования к программным документам (с Изменением N 1)».

СОДЕРЖАНИЕ

1. Назначение программы

1.1. Функциональное назначение программы

Программа предназначена для проведения анализа вторичной структуры РНК и проведения расчетов, связанных с предсказанием вторичной структуры РНК.

1.2. Эксплуатационное назначение программы

Программа предназначена для эксплуатации биологами и биоинформатиками в ходе проведения анализа вторичной структуры РНК.

1.3. Состав функций

1. Предсказание вторичной структуры РНК на основе одной последовательности (применение модуля RNAfold).
2. Предсказание вторичной структуры РНК на основе нескольких последовательностей (применение модуля RNAalifold).
3. Множественное выравнивание последовательностей первичных структур РНК (применение модуля Clustal Omega).

2. Условия выполнения программы

2.1. Минимальный состав аппаратных средств

1. Операционную систему Ubuntu версии 20.04 и выше.
2. Оперативная память объемом 2 Гб или выше.
3. Процессор с количеством ядер 1 или выше.
4. Жесткий диск объемом 20 Гб или выше.

2.2. Минимальный состав программных средств

1. Операционная система Ubuntu версии 20.04 и выше.
2. Среда выполнения кода JavaScript Node.js.

2.3. Требования к оператору

Конечный пользователь программы (оператор) должен обладать практическими навыками использования интерфейса командной строки.

3. ВЫПОЛНЕНИЕ ПРОГРАММЫ

3.1. Загрузка и запуск программы

Запуск программы происходит из командной строки. Для запуска сервера необходимо:

1. Перейти в директорию, где расположены файлы сервера.
2. Выполнить команду: nohup node app.js.

3.2. Выполнение программы

В процессе обработки запросов от пользователей, программа генерирует временные директории и папки, которые затем автоматически удаляются. Дополнительный действий со стороны оператора не требуется.

3.3. Завершение работы программы

Для завершения работы программы необходимо:

1. Выполнить команду: ps -C node, чтобы узнать номер процесса.
2. Выполнить команду: kill [pid], где pid – номер процесса из п. 1.

4. Сообщения оператору

При обработке запроса от пользователя программа пишет в лог-файл следующую информацию:

1. URL полученного запроса.
2. Время, в которое был получен запрос.
3. Входные данные запроса.
4. Время, в которое был отправлен ответ.
5. Выходные данные запроса.

При возникновении внутренних ошибок программа пишет сообщение об ошибке и детали ошибки.