УДК 004.8

ИНФОРМАЦИОННЫЙ ТРЕНАЖЕР ПО ОСНОВАМ МЕНДЕЛЕВСКОЙ ГЕНЕТИКИ

Гуров Д.А., Лукин В.В.

Предложен новый подход к построению информационного тренажера, обеспечивающий повышение качества и эффективности приобретения знаний и навыков по основам менделевской генетики.

A new approach to development of the computer simulator, which is intended for improving the quality and effectiveness of knowledge and skills acquisition in Mendelian genetics, is suggested.

1. ВВЕДЕНИЕ

Процесс обучения в той или иной предметной области состоит в изучении теоретического материала, его усвоении и приобретении практических навыков его использования. Одним из полезных и гибких инструментов помощи ученику и учителю в работе с предметной областью являются информационные тренажеры [1]. Преимущество их применения перед имитационными состоит в существенно меньшей трудоемкости реализации, более широкой вариабельности, возможности применения в предметной области, не требующей высокой степени наглядности для приобретения соответствующих навыков, как например, пилотирование воздушных судов.

Данная работа посвящена описанию модели информационного тренажера и практической его реализации применительно к менделевской генетике, т.е. в области, которая имеет ряд особенностей по сравнению с описываемыми в [1] предметными областями. Теоретические и прикладные задачи касаются изучения моделей наследования определенных признаков у живых организмов. Базовое положение генетики заключается в том, что наследственная информация передается лишь при помощи ДНК [2]. Исключение здесь составляют лишь вирусы, где носителем наследственной информации является РНК. Информация, находящаяся в ДНК преобразуется в белок или РНК различных типов. Белки различных типов являются участниками основных процессов в организме[1], а РНК является важнейшим компонентом большинства внутриклеточных процессов[2]. Определения гена постоянно уточняется, так как постоянно пополняются научные данные о функциональном значении генов и их пространственной организации. В рамках рассматриваемой предметной области примем такое определение: ген – участок ДНК, контролирующий тот или иной признак. Совокупность генов

[1] Синтез веществ и их обмен лежит в основе большинства процессов, происходящих в организме живых существ. Регуляцию синтеза и обмена веществ осуществляют белки, называемые ферментами. [Г. Р. Мутовин. Клиническая генетика, стр. 212]

[2] РНК является регулятором синтеза белка, регулятор процессов, связанных с ДНК клетки. [Г. Р. Мутовин. Клиническая генетика, стр. 46-51]

организма – генотип. Совокупность признаков организма – фенотип. Процесс синтеза белка или РНК на основе информации гена – экспрессия гена. В рамках модели Менделя предполагается равновероятный и независимый обмен генетическим материалом внутри генотипа (попадание каждого из аллелей гена в гаметы равновероятно и не зависит друг от друга). Эти утверждения соответствуют постулатам о парности наследственных единиц, доминантности и рецессивности признаков и постулату о расщеплении генов. Если рассматривается скрещивание организмов по группе признаков, то предполагается, что гены, определяющие данный признак, попадают в гаметы независимо[3]. Эти постулаты проверяются в исследованиях клеточных и молекулярных основ этих закономерностей, а также математическими методами. Генотипы родителей дают определенные пропорции фенотипов у потомков [3, 4]. На рис.1 приводится графовая модель предметной области в общем виде.

Менделевская генетика – узкая область, базирующаяся на устоявшихся правилах и проверенных методах их применения. С данным разделом сталкиваются все изучающие курс генетики, т.к. в нем рассматриваются практические все базовые понятия. Следует отметить, что в генетике уже предпринимаются успешные попытки создания обучающих систем, например, проект Rosalind [5], который представляет собой интерактивный сборник задач по биоинформатике, однако этот продукт относится скорее к системам самоконтроля, и его в полной мере нельзя назвать тренажером.

При изучении данной области учащиеся сталкиваются с рядом проблем:

- недостаточная общая теоретическая подготовка;
- отсутствие базовых знаний;
- недостаточная подготовка по предметам, лежащих в основе менделевской модели;
- необходимость приобретения практических навыков применения методов смежных дисциплин, например, статистического анализа данных, столь важных для обработки результатов практических экспериментов.

Все эти проблемы решаются практикой, которая предполагает наличие постоянной обратной связи, как например, занятиях с преподавателем.

Информационный тренажер – инструмент, который расширяет возможности освоения генетики за счет организации прямой и обратной связи субъект-субъектных отношений. Прямая связь – подача учебного материала, обратная –выполнение заданий для самоконтроля.

Подача материала подразумевает соблюдение нескольких принципов построения модели предметной области, ослабленных по сравнению с [1], но с добавлением условий организации обратной связи:

- порционность;
- ассоциативность;
- порционность;
- однозначность задач самоконтроля.

Базовый курс менделевской генетики соответствует определяемым принципам, следовательно, существует возможность построения информационного тренажера для его изучения.

2. МОДЕРНИЗАЦИЯ МОДЕЛИ ИНФОРМАЦИОННОГО ТРЕНАЖЕРА

Модель знаний представляется семантической сетью:

$$G =< \mathbf{P}, \mathbf{R}, \mathbf{T}_R > \quad (1)$$

[3] Постулат о независимом комбинировании признаков

$\mathbf{P}$ – информационные узлы, $\mathbf{R}$ – отношения между ними, $\mathbf{T}_R$ – множество типов отношений.

В рассматриваемой модели информационного тренажера предполагается организация промежуточного контроля, результатом которого будет либо продолжение изучения материалов, либо повторное обучение субъекта. Для обеспечения обратной связи используется промежуточный контроль знаний в виде закрытой формы тестирования. Модель тренажера [1] можно модифицировать, добавив условия перехода от вершины к вершине, что идентично компонентно-каркасному подходу [6]. Однако при этом существенно теряется наглядность модели представления знаний, т.к. выбор пути и возможности перехода будет представлять черный ящик, что неудобно для специалиста предметника, разрабатывающего информационный тренажер. В данной работе предлагается альтернативный подход. Пусть в графе знаний:

$$\mathbf{G} =< \mathbf{P}, \mathbf{R} > \tag{2}$$

необходимо осуществить промежуточный контроль между элементами знаний (темами)[4], которые можно представить в виде подграфов. Для этого введем операцию свертки подграфа в схему [2]. Заметим, что схема по своим свойствам соответствует классу эквивалентности. Из этого следует, что модель знаний (1) представима в виде (2). Для этого достаточно определить классы эквивалентности по любому типу отношений из $\mathbf{T}_R$.

Определим операцию свертки подграфа. Для этого рассмотрим произвольный подграф, как произвольный новый класс эквивалентности:

$$\forall G_i =< \mathbf{P}_i, \mathbf{R}_i > \exists C_i : \forall p_i \in \mathbf{P}_i \Rightarrow p_i \in C_i, i \in \mathbf{N} \tag{3}$$

Тогда, свертка подграфа определяется отображением множества его вершин в произвольную вершину этого подграфа:

$$\vec{G}_i : \mathbf{P}_i \to p_i \ \ \forall p_i \in \mathbf{P}_i, i \in \mathbf{N} \tag{4}$$

Заметим, что данная операция сохранит отношения вершин графа с вершинами подграфа. Определим операцию детализации вершины графа как отображение множества элементов класса эквивалентности, образованного вершиной:

$$p_i \to \mathbf{P}_i = \{p_i\}, C(p_i) = C_i, \forall p_i \in \mathbf{P}_i, i \in \mathbf{N} \tag{5}$$

Модель представления знаний представляет собой отношения классов эквивалентности[5]. Информационный тренажер определяет возможность определение порядка вложенности. Так, например, обучающийся может первоначально ознакомиться с базовыми понятиями курса целиком, потом более углубленно изучить какую-либо тему, или последовательно изучать все темы от начала и до конца, или выбрать какой-либо другой доступный порядок изучения предмета.

Обратная связь в информационном тренажере осуществляется на статическом и динамическом уровне. Статические формы нужны для улучшения модели представления знаний, а динамические формы – для организации самоконтроля. Промежуточное тестирование – реализация динамической формы, по сути, с помощью него осуществляется автоматизированное управление процессом обучения. Корректная и эффективная его реализация – важ-

[4] Разбиение тем на подтемы в контексте данной модели не рассматривается, т.к. далее вводится рекурсивная операция, позволяющая детализировать или обобщать знания.

[5] Например, иерархической вложенности

нейшее условие, определяющее качество информационного тренажера в целом. Важно наглядно представить системные события, возникающие после прохождения процедуры промежуточного контроля.

Тестовое задание закрытой формы с множественным выбором определяются следующими типами информационных составляющих (рис.1):

- условие задачи;
- задача;
- варианты ответов.

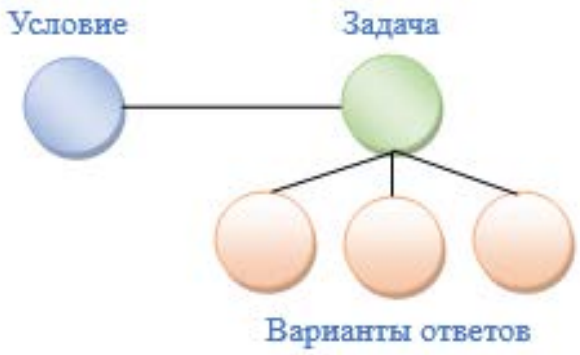

Рис. 1. Граф тестового задания

Для произвольной вершины графа знаний можно определить несколько условий тестовых заданий, которые, соответственно, будут принадлежать одному и тому же классу эквивалентности. Для каждого условия задания могут быть сформулированы несколько задач, которые также будут принадлежать тому же классу эквивалентности. Для информационного тренажера вводится следующее правило: класс эквивалентности, образованный темой должен быть однородным, т.е. содержит либо только задания, либо только темы.

3. АЛГОРИТМ РАБОТЫ ИНФОРМАЦИОННОГО ТРЕНАЖЕРА

1. Предъявляется тема.
   1.1. Если существует класс эквивалентности, образованный темой, проверяется его тип.
      1.1.1. Если класс эквивалентности содержит темы, они предъявляются в виде списка с возможностью выбора каждой из них.
         1.1.1.1. Выбранная тема отображается.
      1.1.2. Иначе:
         1.1.2.1. случайным образом выбирается задание промежуточного контроля;
         1.1.2.2. предъявляется вершины графа задания: условие, задача и варианты ответы в виде списка, с возможностью выбора любой из них;
         1.1.2.3. выбранный ответ фиксируется и отображается на экране;
2. Осуществляется переход к связной теме.

При отображении тем можно ограничить время отображения. В этом случае, применяется модель, в которой реализуются условия перехода [6]. Иногда типовые задачи требуют использования открытого ответа. В этом случае, открытый ответ формулирует значение, определяющее класс эквивалентности вложенного подграфа, по сути – фильтр данных. Если ответ правильный, осуществляется переход к следующей схеме, отображающей очередной элемент знаний, в противном случае осуществляется возврат на обобщающий уровень.

В рамках представляемой модели информационного тренажера возможны достаточно широкие расширения функциональности, например, реализация комментариев и пояснения к решению задачи, на основе вводимых субъектом ответов.

4. ПРЕДСТАВЛЕНИЕ МОДЕЛИ ЗНАНИЙ ПРЕДМЕТНОЙ ОБЛАСТИ

Рассмотрим модель знаний базового курса менделевской генетики. Теоретические и прикладные задачи касаются изучения моделей наследования определенных признаков у живых организмов. В рамках теории Менделя предполагается равновероятный и независимый обмен генетической информацией внутри генотипа (попадание каждого из аллелей гена в гаметы равновероятно и не зависит друг от друга). Эти утверждения соответствуют постулатам о парности наследственных единиц, доминантности и рецессивности признаков и постулату о расщеплении генов. Если рассматривается скрещивание организмов по группе признаков, предполагается, что гены, определяющий данный признак, попадают в гаметы независимо[6]. В настоящее время эти постулаты подтверждены с одной стороны исследованиями клеточных и молекулярных основ этих закономерностей, с другой – математическими методами, в т.ч. точные количественные параметры их проявления в популяции при разных генотипах с другой стороны. Генотипы родителей дают определенные пропорции фенотипов у потомков [3,4]. На рис.2 приводится графовая модель предметной области в общем виде.

Рис. 2. Графовая модель знаний предметной области

Приведем примеры типовых учебных задач, определяемых представленной моделью знаний предметной области:

- Известен генотип животного. Гаметы с каким генетическим материалом может произвести животное?
- Известен фенотип животного, определить вероятный генотип животного?
- Известен генотип животных поколения родителей, определить фенотипы и генотипы первого и второго поколения.
- Известны количество животных с различными фенотипами первого и второго поколения, определить фенотипы и генотипы родителей.
- Определить подчиняется ли группа признаков или признак менделевской модели наследования.

[6] Постулат о независимом комбинировании признаков

Выбранная предметная область выгодно отличается следующими признаками, позволяющими эффективно реализовать учебный курс в рамках представляемой модели информационного тренажера:

- *Структурированность знаний.* Элементы знаний предметной области представляются деревом (рис.2)
- *Возможность генерации решения.* Возможно генерировать типовые задачи, следовательно, существенно упростить модель тренажера, повысив качество обратной связи за счет исключения повторяемости.

5. РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОГО ТРЕНАЖЕРА

В качестве инструмента реализации системы выбран язык Python[7]. Свойства, определяющие выбор:

- прост, понятен, реализует парадигмы структурного, объектно-ориентированного, функционального программирования;
- открытая модель распространения самого интерпретатора и многих его расширений;
- в рамках расширений реализованы различные библиотеки: математические, статистические, обработки изображений, работы с интернет-службами;
- возможность работы с множеством встроенных структур данных (списки, словари, хеш-таблицы и т.д.), различными форматорами данных (CSV, XML, JSON и т.д.) и множеством СУБД (реляционных, объектно-ориентированных, графовых);
- существует множество надстроек для создания приложений;
- Политика лицензирования Python позволяет его свободно использовать как в коммерческих продуктах, так и в закрытых;
- Существуют прикладные решения, реализующие, например, семантическую сеть[8] или экспертную систему.

Проект предполагается реализовать как веб-сервис, используя технологию шаблонов проектирования MVC (Model-View-Controller) [7]. Это трехуровневое представление приложения: модель – часть приложения, которая получает данные и обрабатывает их, представление (вид) – часть приложения, отвечающая за отображение данных, контролер – часть приложения, отвечающая за обработку событий, в т.ч. действия пользователя. Важным моментом является то, что различные части MVC приложения должны решать лишь свою задачу и обращаться к интерфейсам других частей приложения при необходимости. Таким путем достигается большая структурированность и понижение связанности приложения. Шаблон MVC (производный шаблон MVP) реализован во многих средствах и средах разработки: фреймворк разработки настольных приложений Qt5, веб-фреймворк Django, фреймворк разработки настольных приложений от Microsoft WPF и другие. В частности, для реализации информационного тренажера выбран веб-фреймворк Flask[9], который поддерживает концепцию MVC, является компактным и оставляет многие детали реализации на усмотрение разработчика. Рассмотрим базовый набор инструментов, который предполагается применить для реализации MVC-уровней тренажера. Уровень модели реализуется при помощи реляционной СУБД PostgreSQL[10] и библиотеки sqlalchemy[11] как инструмента взаимодействия приложения и СУБД. Обработка данных на уровне модели реализуется собственными компо-

[7] Официальный сайт: http://www.python.org/about/
[8] http://www.strout.net/info/coding/python/ai/index.html
[9] Сайт проекта Flask: http://flask.pocoo.org/

[10] Сайт СУБД PostgreSQL: http://www.postgresql.org/
[11] Сайт SQLAlchemy: http://www.sqlalchemy.org/

нентами. PostgreSQL изначально проектировалась как СУБД для обработки больших объемов данных, она поддерживает:

- множество типов данных, помимо стандартных, например, SQL массивы, XML, геоданные;
- множество языков, как для клиентских приложений, так и для создания хранимых процедур, в т.ч. Python;
- правила, которые реализуются исполняемыми процедурами при разборе запроса;
- многоверсионность базы данных, как механизм многопользовательской работы, что позволяет минимизировать число блокировок;
- собственные типы данных;
- наследование структуры и свойств новых таблиц от существующих;
- реализацию собственных алгоритмов индексации и поиска на основе механизма GiST[12] (обобщенные деревья поиска).

PostgreSQL полностью поддерживает требования ACID, является открытой СУБД и может использоваться в коммерческих проектах без каких-либо ограничений.

Язык SQL является очень мощным инструментом обработки данных. Однако сам процесс интеграции SQL в программу часто сопряжен с множеством тонкостей, начиная от инициализации драйвера взаимодействия СУБД и заканчивая различными преобразованиями данных между системой типов SQL и языка программирования приложения. Для решения проблем взаимодействия программного приложения и СУБД во многих языках шли по пути стандартизации интерфейса драйверов к различным СУБД. В Python с этой целью была разработана спецификация DB-API 2.0, однако многие модули поддержки СУБД в Python все равно достаточно низкоуровневые. Сам процесс перехода разработчика из парадигмы ООП в парадигму реляционной модели данных достаточно сложен и сопряжен с ошибками. Работа с SQL так же подвержена проблемам безопасности и семантическим, техническим ошибкам. Одним из решений данной проблемы является механизм ORM (Object-relational mapping) [8] – технология создания виртуальной объектной базы данных, позволяющая организовывать доступ к реляционной базе данных используя объектно-ориентированную парадигму, при этом сущности рассматриваются как классы, атрибуты – как поля классов. Связи между сущностями реализуются как объекты, которые поддерживаются и создаются самой системой. Базовые операции языка SQL такие как, например, запрос данных по условию, агрегатные функции и другие ORM системой представляются как методы класса сущность. Вызовы метода система преобразует в корректный SQL-запрос. Например, используя данный подход можно вместо SQL-запроса количества пользователей с использованием агрегатной функции count(), вызвать класс users с методом count: users.count(*), который автоматически сгенерирует соответствующие блоки запроса. Для реализации этого подхода используется библиотека sqlalchemy. Кроме вышеописанных возможностей она поддерживает динамическое построение схемы базы данных в приложении.

При реализации уровня представления (вид) используется шаблонизатор[13] Jinja2[14]. Когда необходимо показать данные пользователю, шаблонизатор преобразует свои инструк-

[12] Сайт проекта GiST: http://gist.cs.berkeley.edu/

[13] При разработке веб-сайтов очень часто возникала проблема смешивания кода, отвечающего за логику приложения, с разметкой на языке HTML. В этом случае, для удобства разработчика и минимизации ошибок используется интерпретатор специального языка, который может быть скомбинирован с HTML или иным языком разметки, в него входят, как правило, работа с разными типами переменных, программные конструкции, например, операторы цикла и условия. Файл с кодом разметки и дополнительными инструкциями называется шаблоном, отсюда термин «шаблонизатор».

[14] Сайт проекта Jinja2: http://jinja.pocoo.org

ции в конкретные значения переменных и код HTML, затем передает полученный документ другим частям приложения.

Уровень контролера реализуется средствами Flask. При реализации слоя контролера для веб-сайта используют понятие маршрута – набора страниц или иных объектов, который посещает пользователь при выполнении той или иной задачи в приложении. Местоположение той или иной страницы или объекта задается при помощи URL. Контролер реализуется как набор функций, которые соотносятся с набором URL, обрабатываемым приложением. Flask позволяет использовать различные шаблоны по которым и будет происходить сопоставление URL-функция. Функция, обрабатывающая определенный URL, называется обработчиком, а механизм, который осуществляет сопоставление – маршрутизатором. Кроме механизма маршрутизации Flask поддерживает обработку данных, поступивших из форм. Для упрощения работы с формами используется расширение Flask – WTForms[15], которое осуществляет обмен данных между пользовательским интерфейсом и приложением.

Важной частью уровня модели нашего тренажера является компонент, генерирующий тестовые задачи различного типа, а также проверяющий их решения. Для формирования и проверки тестовых задач различного типа используется собственный набор функций и классов Python. Алгоритмическая часть проекта предполагает активное использование списков, словарей и других конструкций, поддерживаемых выбранным языком. В качестве примера рассмотрим реализацию одной из операций, используемой при формировании и автоматическом решении тестовых задач.

```
def crossing(self): #функция расчета результата скрещивания
        generation = []
      for gametA in gamets[parentA]:
          for gametB in gamets[parentB]:
                        generation.append(gametA+gametB)
       return generation
```

Фрагмент содержит конструкцию for…in, которая реализует поэлементный последовательный обход любой структуры данных, поддерживающей генераторы [9]. Генератор это специальная разновидность функций, которые реализуют «ленивые вычисления»[16] [10]. Когда в Python генератор работает с последовательностью, он не вычисляет или просматривает ее всю сразу, а берет только тот элемент, который будет необходим именно сейчас. Для этого он запоминает свое предшествующее состояние и на основе него вычисляет свое текущее состояние. В нашем случае это приводит к тому что, если на прошлом шаге генератор вернул *i*-ый элемент, то на текущем шаге он сразу же обратиться к *i+1*-му элементу или сообщит о конце последовательности и не будет повторно просматривать ее всю в поисках нужного элемента. Конструкция for…in автоматически задействует такую возможность данной конструкции, если она реализована.

6. РЕЗУЛЬТАТЫ

Предложенная модель информационного тренажера является не просто расширением [1, 6, 11]. Создание объектов типа «схема» не предполагает образование новых кортежей. Для их организации используется фильтр данных по типам связей между вершинами графа и значению их атрибутов. Реализация субъектно-объектной парадигмы [11] также предполагает создание только одного объекта – вершины.

[15] Сайт проекта WTForms: http://wtforms.readthedocs.org/en/latest/

[16] Такой подход к вычислениям предполагает вычислять значения только тогда, когда это необходимо.

Добавление возможности организация самоконтроля для обучающегося с заданием временных границ для решения задач, позволяет пользователям проводить более качественную самооценку приобретенным знаниям и навыкам решения практических задач.

Возможность повторного изучения теоретического материала при необходимости (по результатам тестирования) повышает качество приобретенных знаний и, следовательно, ценность информационного тренажера.

Практическая реализация тренажера в веб-интерфейсе с применением открытых технологий в качестве инструмента открывает широкие возможности и перспективы его использования, в т.ч. в других предметных областях.

ЛИТЕРАТУРА

1. Поминов Д.А., Лукин В. В. Проектирование информационных тренажеров для автоматизации учебного процесса. – М.: Нейрокомпьютеры, №9, 2013.
2. Crick, F. Central Dogma of Molecular Biology. Nature 227, 1970, p.561—563. PMID 4913914
3. Уильям С. Клаг, Майкл Р. Каммингс. Основы генетики. – М.: «Техносфера», 2009, с. 46-105.
4. Льюин Б. Гены. – М.: Мир, 1987, с. 8-21.
5. Сайт системы Rosalind: http://rosalind.info/
6. Лукин В.В., Лукин В.Н. Компонентно-каркасный подход к разработке программного обеспечения. –Моделирование и анализ данных, №1, 2011, с.131-140.
7. Курс лекций CSE 421 — Software Development in Java, Winter 2009, лекция 23: http://www.cse.ohio-state.edu/~rountev/421/lectures/lecture23.pdf
8. Scott W. Amber Mapping Objects to Relational Databases: O/R Mapping In Detail: http://www.agiledata.org/essays/mappingObjects.html
9. Hudak, Paul (September 1989). "Conception, Evolution, and Application of Functional Programming Languages". ACM Computing Surveys 21 (3): 383–385.
10. Соколов Е. Н. Механизмы памяти. – М.*:* Издательство Московского университета, 1969. 174 с.
11. В.В. Лукин, Е.О. Фесик. Графовая модель организации динамических пользовательских интерфейсов в экспериментальной психологии. – М.: Нейрокомпьютеры №9, 2012.

Работа поступила 4.02.2014