Т. С. Стефанова

ОТБОР СОДЕРЖАНИЯ ОБУЧЕНИЯ НЕКЛАССИЧЕСКИМ ВЫЧИСЛИТЕЛЬНЫМ МОДЕЛЯМ

Работа представлена кафедрой информатики.
Научный руководитель – доктор педагогических наук, профессор М. В. Швецкий

Данная статья описывает подход к отбору содержания обучения классическим и неклассическим вычислительным моделям, что, возможно, может привести к формированию у обучаемых нефундаментальной трактовки понятия «*вычислимость*» и демонстрирует возможный вариант реализации нефундаментального подхода к обучению информатике.

The article describes the approach to selection of materials on teaching classical and non-classical computation models, which may result in forming of a nonfundamental interpretation of the «computability» concept and which shows a possible way of realisation of a nonfundamental approach to computer science training.

Метод построения и изучения моделей широко используется в науке. Сущность этого метода состоит в следующем. Создается модель объекта, а затем изучаются свойства этой модели. Полученные свойства объявляются свойствами самого объекта. Это верно в том случае, когда модель адекватна изучаемому объекту (т. е. правильно отражает его существенные черты). Адекватность модели проверяется экспериментально.

Вычислительными моделями будем называть математически точные понятия, моделирующие понятие «*алгоритм*», или одно из связанных с ним понятий, опирающееся на понятие «*процесс*»: «*вычислительный процесс*», «*алгоритмический процесс*».

Выделим два основных класса вычислительных моделей:

1) классические вычислительные модели;

2) неклассические вычислительные модели.

К *классическим вычислительным моделям* будем относить следующие вычислительные модели:

1) *представительные вычислительные модели*: нормальные алгорифмы Маркова (НАМ), машина Тьюринга (МТ), многоленточная машина Тьюринга, обратимая машина Тьюринга, машина Поста–Успенского (МПУ), машина Минского, машина с конечным числом переменных, машина с неограниченными регистрами (МНР), равнодоступная адресная машина (РАМ), машина с параллельным доступом к памяти (ПМНР);

2) *непредставительные вычислительные модели*: конечные автоматы, магазинные автоматы.

Неклассическими вычислительными моделями будет считать:

1) *представительные вычислительные модели*: генетические алгоритмы, нейронные сети, генетическое программирование, равнодоступная адресная машина с хранимой программой (РАСП), автомат Неймана, ДНК-вычисления;

2) *непредставительные вычислительные модели*: конечные автоматы Уотсона–Крика, магазинные автоматы Уотсона–Крика, муравьиные алгоритмы, клеточные автоматы (одномерные и двухмерные).

В соответствии с предложенным вариантом разделения вычислительных моделей, приведем логическую структуру содержания обучения (рис. 1).

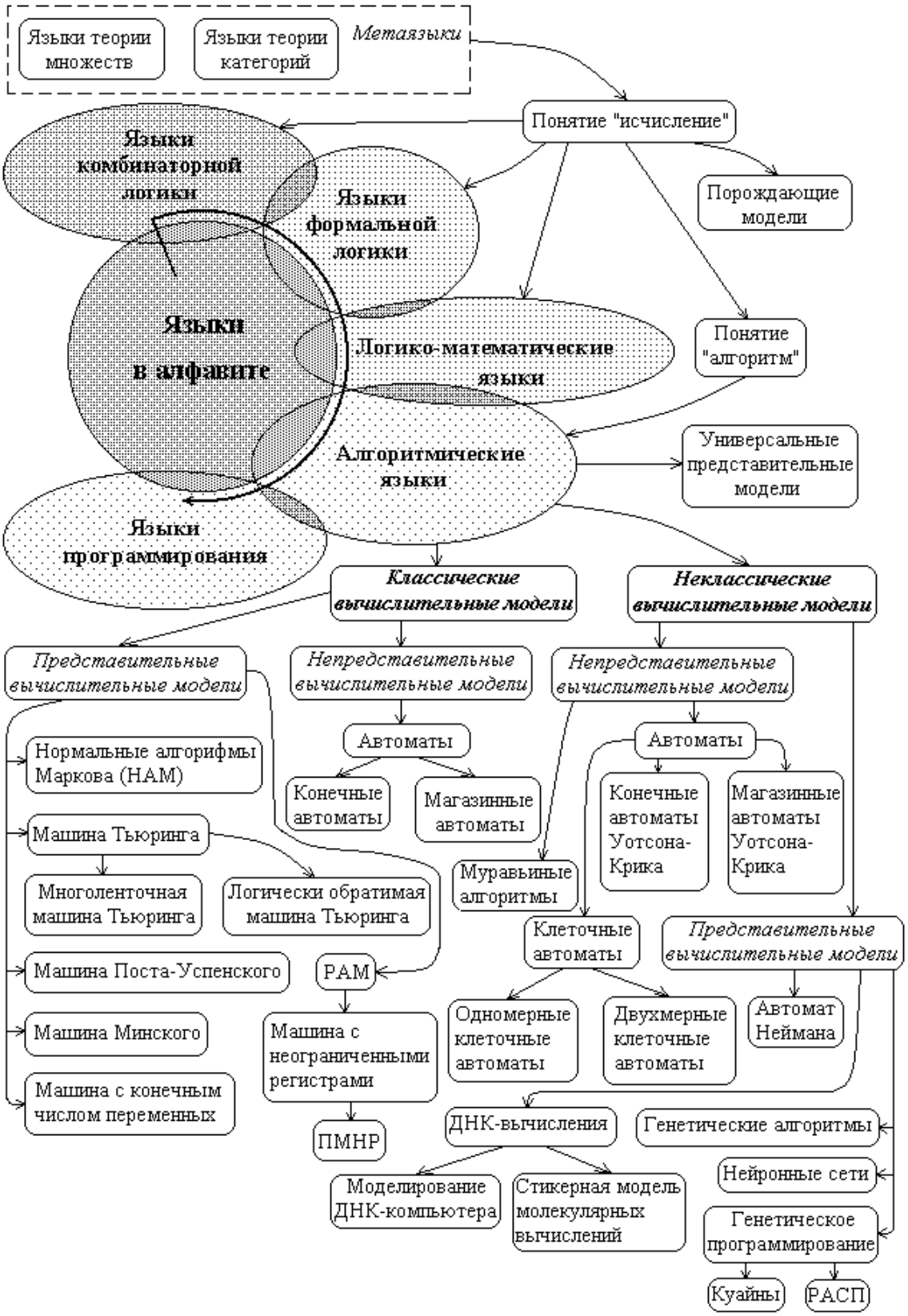

Рис. 1. Логическая структура содержания образования

В настоящее время иногда понятие «неклассические вычислительные модели» определяется с помощью обобщающего понятия «природные вычисления».

1. Уточнение понятия «природные вычисления». Понятие, смысл которого мы собираемся уточнить, содержит два слова: «природные» и «вычисления».

Вначале остановимся на первом термине. Известен эмпирически установленный *принцип неисчерпаемости Природы*: природа имеет средства для осуществления любой корректно сформулированной человеком задачи[1].

Приведем высказывания ученых, писателей, философов о том, что природа может являться источником новых идей для решения тех или иных задач:

1. Оставьте трудиться напрасно, стараясь извлечь из одного разума всю мудрость; спрашивайте *природу*, она хранит все истины и на вопросы ваши будет отвечать вам непременно и удовлетворительно (Ф. Бэкон).

2. Гони *природу* в дверь, она влетит в окно… (Ж. де Лафонтен. Кошка, превращенная в женщину).

3. *Природа* – это исчислимое (О. Шпенглер. Закат Европы. Т. 1).

4. *Природа* мерещится при взгляде на эту картину в виде какого-то огромного, неумолимого и немого зверя или, вернее, гораздо вернее сказать, хотя и странно, – в виде какой-нибудь громадной машины новейшего устройства, которая бессмысленно захватила, раздробила и поглотила в себя, глухо и бесчувственно, великое и бесценное существо – такое существо, которое одно стоило всей природы и всех законов ее, всей земли, которая и создавалась-то, может быть, единственно для одного только появления этого существа! (Ф. М. Достоевский. Идиот. Ч. 3. Гл. 5).

5. Может ли *природа* быть информационным процессором?[2]

6. В *природе* есть возможность делать унитарные преобразования, т. е. действовать на бесконечном множестве[3].

Вторым словом является термин «вычисление». Таким образом, видимо, Природа каким-то образом может помочь человеку в реализации вычислительных процессов.

Ясно, что фраза «Природа вычисляет» является эллипсисом: более точно говорить, что имеет смысл искать (и находить) новые вычислительные модели в многочисленных моделях реальности (например, в биолого-химических и квантово-механических).

Таким образом, мы смотрим через призму моделей реальности на понятие «вычислимость»; в результате появляется новый класс моделей вычислений, которые называются «неклассические вычислительные модели».

Интересно сравнить сказанное с классической вычислительной моделью – машиной Тьюринга. А. Тьюринг спроецировал на свою машину собственное отношение к миру, а принципы обработки информации, заложенные в нее, несомненно, являются результатом интроспекции (внутреннего самонаблюдения). Таким образом, машина Тьюринга – это прежде всего модель самого Тьюринга, своеобразный памятник собственному интеллекту; иными словами, особенности мировосприятия Тьюринга как человека наиболее точным образом представляются его машиной.

Приступим к более точному описанию содержания понятия «природные вычисления», которое объединяет следующие разделы:

А. *Эволюционное программирование*:

1) клеточные автоматы;

2) генетические алгоритмы:

а) муравьиные алгоритмы;

б) генетическое программирование.

Б. нейросетевые вычисления.

В. ДНК-вычисления.

Г. квантовые вычисления.

Определим основные понятия этого раздела, относящиеся только к эволюционному программированию.

1. *Эволюционное программирование* – это область знаний, исследующая сообщества связанных друг с другом *конечных автоматов* (*клеточных автоматов*) и анализирующая их взаимодействие[4].

Эволюционное программирование является разновидностью парадигмы программирования, которая называется *программированием от состояний* (или *автоматным программированием*).

2. *Клеточный автомат* – это семейство *конечных автоматов*, демонстрирующее *эмерджентное* (*проявляющееся*) *поведение* при взаимодействии элементов этого семейства.

Примером реализации клеточных автоматов является игра «Жизнь».

3. *Генетическими алгоритмами* называют формальные алгоритмические аналоги процесса эволюции, для реализации которых используются точные определения *эволюционных операций над элементами популяции*[5].

4. *Генетическое программирование* – это программирование с использованием *генетических операторов*, предназначенное для моделирования процесса эволюции компьютерных программ.

Поскольку генетическое программирование является методом автоматического генерирования компьютерных программ, то его можно отнести к области *автоматического программирования*.

5. *Муравьиные алгоритмы* (или оптимизация по принципу муравьиной колонии) обладают специфическими свойствами, присущими муравьям, и используют их для ориентации в физическом пространстве. Природа предлагает различные методики для оптимизации некоторых процессов.

Муравьиные алгоритмы особенно интересны потому, что их можно использовать для решения не только статических, но и динамических проблем, например маршрутизации в меняющихся сетях.

Подведем итоги.

Исследователи обращаются к природным механизмам, которые миллионы лет обеспечивают адаптацию биоценозов к окружающей среде. Одним из таких механизмов, имеющих фундаментальный характер, является *механизм наследственности*. Его использование для решения задач оптимизации привело к появлению *генетических алгоритмов*.

В живой природе особи в биоценозе конкурируют друг с другом за различные ресурсы, такие как пища или вода. Кроме того, особи одного вида и популяции конкурируют между собой, например, за привлечение брачного партнера. Те особи, которые более приспособлены к окружающим условиям, будут иметь больше шансов на создание потомства. Слабо приспособленные либо не произведут потомства, либо их потомство будет очень немногочисленным. Это означает, что гены от высокоприспособленных особей будут распространяться в последующих поколениях. Комбинация хороших характеристик от различных родителей иногда может приводить к появлению потомка, приспособленность которого даже больше, чем приспособленность его родителей.

Итак, вид в целом развивается, все лучше и лучше приспосабливаясь к среде обитания.

Приведенное выше уточнение понятия «природные вычисления» мы используем для развития содержания обучения вычислительным моделям, в частности выделением класса, который мы называем «неклассические вычислительные модели».

2. Отбор содержания обучения генетическим алгоритмам и генетическому программированию.

Далее мы рассмотрим, как происходит отбор содержания обучения генетическим алгоритмам и генетическому программированию бакалавров физико-математического образования.

Дж. Коза[6] высказал предположение, что удачная компьютерная программа может

эволюционировать за счет применения генетических операторов. При этом структурами, к которым они применяются, являются *иерархически организованные фрагменты компьютерных программ*. Другими словами, алгоритм обучения работает с популяцией программ-кандидатов.

В связи с этим необходимо определить понятия «генетический алгоритм» и «генетической программирование».

Генетическими алгоритмами будем называть формальные алгоритмические аналоги процесса эволюции, для реализации которых используются точные определения *эволюционных операций над элементами популяции*[7].

Генетическое программирование – это программирование с использованием *генетических операторов*, направленное на описание процесса эволюции компьютерных программ.

Генетическое программирование – это направление эволюционного поиска, ориентированное на решение задач автоматического синтеза программ. *Хромосомы* (компьютерные программы различной величины и сложности) автоматически генерируются с помощью *генетических операторов*[8].

Генетическое программирование представляет собой модификацию генетического алгоритма с учетом возможностей компьютерных программ. При использовании этого метода популяция состоит из закодированных соответствующим образом программ, которые подвергаются воздействию генетических операторов скрещивания и мутации для нахождения оптимального решения, которым считается программа, наилучшим образом решающая поставленную задачу. Программы оцениваются относительно определенной специальным образом *функции приспособленности*[9].

Теперь рассмотрим содержание обучения генетическим алгоритмам и генетическому программированию. В зависимости от уровня начальных знаний студентов, целей обучения и продолжительности курса может быть выделена следующая последовательность тем.

1. Генетические алгоритмы. Генерирование начальной популяции. Основные операторы кроссинговера.

Основные понятия:

• «генетический алгоритм», функция приспособленности (целевая функция), понятие генетического оператора;

• стратегии создания начальной популяции: принцип «одеяла», «дробовика», «фокусировки», «комбинирования»;

• основные операторы кроссинговера: простой (одноточечный) оператор кроссинговера, двухточечный оператор кроссинговера, многоточечный оператор кроссинговера, упорядоченный оператор кроссинговера, частично-соответствующий оператор кроссинговера, циклический оператор кроссинговера, универсальный оператор кроссинговера, «жадный» оператор кроссинговера.

2. Основные операторы мутации.

Основные понятия:

• «мутация», «оператор мутации»;

• основные операторы мутации: простой оператор мутации, мутация обмена, одноточечный оператор мутации, оператор мутации на основе правила «золотого сечения», двухточечный оператор мутации, оператор инверсии, оператор транслокации, оператор транспозиции, оператор сегрегации, оператор удаления, оператор вставки.

3. Селекция (операторы репродукции).

Основные понятия:

• «селекция», «оператор репродукции», «отбор», «естественный отбор», «бессознательный отбор», «методический отбор», «расстояние Хэмминга»;

• основные виды селекции: оператор селекции на основе рулетки, оператор селекции на основе заданной шкалы, оператор элитной селекции, оператор турнирной селекции, оператор «близкого родства», оператор «дальнего родства».

4. Построение простого генетического алгоритма.

Основные понятия:

• «классический генетический алгоритм», «инициализация», «оценка приспособленности», «генетический микроалгоритм»;

• алгоритм Д. Холланда, простой генетический алгоритм Д. Гольдберга, простой генетический алгоритм Л. Девиса.

5. Генетическое программирование.

Основные понятия:

• «генетическое программирование»;

• генетические операторы в пространстве программ: воспроизводство, оператор кроссинговер (скрещивание), мутация, перестановка.

Данный отбор содержания обучения предполагает выбор средств обучения. Будем считать, что основным средством обучения являются языки программирования С и LISP. Причем выбор таких языков программирования связан непосредственно с возможностями, предоставляемыми этими языками. Например, реализация задач, связанных с темой «генетическое программирование», наиболее удобна на языке программирования LISP, так как он обладает следующими особенностями:

1) эквивалентность представления программ и данных в языке, что позволяет интерпретировать структуры данных как программы и модифицировать программы как данные;

2) применение в качестве основной управляющей структуры не итерации (как в языках императивного программирования), а рекурсии;

3) широкое использование структуры данных «список», в силу чего обработка списков является основой многих алгоритмов;

4) компактный код программ.

На основе построенной структуры содержания обучения определим требования, предъявляемые к начальным знаниям студентов:

1) основные понятия программирования на языках С и LISP (переменная, оператор присваивания, оператор условного перехода, циклы, функции, рекурсия, массивы, строки, линейные списки);

2) умение конструировать программы за приемлемое время;

3) отлаживать и тестировать программы;

4) грамотно интерпретировать полученные результаты решения задачи с помощью компьютера.

Таким образом, студент изначально должен иметь навыки программирования на языке С или LISP.

Приведем примеры некоторых типов задач.

1. Напишите функцию на языке программирования С, моделирующую одноточечный оператор кроссинговера с помощью структуры данных «список».

2. Напишите функцию на любом языке программирования, моделирующую генетический оператор кроссинговера над программами.

В результате обучения генетическим алгоритмам и генетическому программированию на языках программирования С и LISP студенты должны уметь решать задачи, связанные с использованием основных понятий генетического программирования и генетических операторов, а также применение этих понятий при эвристическом способе решения задач математического содержания.

3. ДНК-компьютеры и способы их моделирования. *ДНК-компьютером* будем называть физические устройства, выполняющие операции над цепочками ДНК, основанные на принципе комплиментарности и массовом параллелизме нуклеотидов.

Схематично работа ДНК-компьютера может быть представлена как последовательность трех операций:

1) запись (приготовление) начального состояния;

2) вычисление с применением операций над цепочками ДНК;

3) вывод результата (измерение).

Важно отметить, что, даже если надежда построить ДНК-компьютер не сбудется (например, из-за чувствительности к ошибкам[10]), возможной альтернативой могло бы стать *воплощение этой вычислительной парадигмы в рамках силиконовых технологий*[11]. В связи с этим мы попытаемся промоделировать ДНК-компьютер на «обычном» компьютере, заимствовав уникальные способности ДНК (заложенные природой) по шифрованию и дешифрованию информации, а также организации параллельной обработки данных.

Далее мы продемонстрируем два способа компьютерного моделирования ДНК-компьютера, а именно:

1) моделирование ДНК-компьютера: опыт Эдлмана;

2) моделирование ДНК-компьютера: стикерная модель молекулярных вычислений.

Мы осуществим отбор содержания обучения по данным темам для бакалавров физико-математического образования.

1. *Компьютерное моделирование ДНК-компьютера: опыт Эдлмана.*

При отборе содержания обучения необходимо учитывать минимальный уровень знаний по курсу «Основы дискретной математики», а именно:

1) элементы теории множеств («множество», «мультимножество» и операции над ними);

2) элементы теории графов (определение графа, вершины, ребра, типы графов).

Эти вспомогательные математические понятия наиболее полно отражены в учебном пособии[12].

Также была проанализирована литература по молекулярной биологии[13], что позволило выделить минимальный набор знаний, который необходим при изучении данной темы.

Далее опишем более детально способ моделирования ДНК-компьютера. Его структура такова.

Одинарную цепочку ДНК будем моделировать цепочкой в алфавите {А, Ц, Г, Т}.

Молекулу ДНК (*двойную цепочку ДНК*) будем моделировать ассоциативным списком языка LISP вида $((A_1 . B_1) (A_2 . B_2) \ldots (A_n . B_n))$, где $A_1A_2...A_n$ и $B_1B_2...B_n$ – комплиментарные одинарные цепочки ДНК.

Пробирку (*реальную*), *содержащую одинарные цепочки ДНК*, будем моделировать *мультимножеством*, содержащим цепочки в алфавите {А, Ц, Г, Т}.

В пробирке одинарные цепочки ДНК присутствуют с некоторыми кратностями, т. е. там может содержаться несколько копий одной и той же цепочки.

Важно отметить, что в реальности двойные и одинарные цепочки ДНК являются *процессорами*, способными к взаимодействию с другими цепочками. Это предположение является *основной идеей* при построении ДНК-компьютера, работающего с пробирками.

Для работы с пробирками, содержащими молекулы ДНК, используются следующие операции[14]:

1. *Операция слить* объединяет содержимое двух пробирок в одну пробирку.

2. *Операция размножить* позволяет изготовить две копии данной и той же пробирки.

3. *Операция обнаружить* возвращает значение «*ложь*», если пробирка пуста и «*истина*» – в противном случае.

4. *Операция разделить* позволяет по заданной пробирке N и цепочке w в алфавите {А, Ц, Г, Т} изготовить две пробирки $+(N, w)$ и $-(N, w)$ такие, что $+(N, w)$ состоит из всех цепочек, находящихся в пробирке N и содержащих w в качестве подцепочки, а $-(N,w)$ не обладает таким свойством.

5. *Операция разделить по длине* позволяет по данной пробирке N и натуральному числу n изготовить пробирку $(N,\leq n)$, состоящую из всех цепочек из пробирки N, длина которых не больше n.

6. *Операция разделить по префиксу* позволяет по заданной пробирке N и цепочке

w в алфавите {А, Ц, Г, Т} изготовить пробирку *B*(*N*, *w*), состоящую из всех цепочек из пробирки *N*, *префикс* которых совпадает с *w*.

7. *Операция разделить по суффиксу* позволяет по заданной пробирке *N* и цепочке *w* в алфавите {А, Ц, Г, Т} изготовить новую пробирку *E*(*N*, *w*), состоящую из всех цепочек из пробирки *N*, *суффикс* которых совпадает с *w*.

Наглядно продемонстрируем одну из операций над пробирками, содержащими одинарные молекулы ДНК.

1. Операция *разделить*.

Исходная пробирка *N*: ЦЦТГ, ТТАЦГ, АЦ, ТГТАГА, ТАЦАТ.

Цепочка *w*: ТА.

Результат:

+(*N*, *w*): ТТАЦГ, ТГТАГА, ТАЦАТ.

-(*N*, *w*): ЦЦТГ, АЦ.

2. Операция *разделить по длине*.

Исходная пробирка *N*: ЦЦТГ, ТТАЦГ, АЦ, ТГТАГА, ТАЦАТ.

Натуральное число *n*: *n* = 4.

Результат: (*N*,≤4): ЦЦТГ, АЦ.

Операции над пробирками являются основной для ДНК-программирования, т. е. эти операции позволяют составлять *программы*, отвечающие на вопросы о наличии или отсутствии цепочек, обладающих определенными свойствами.

Приведем программу на метаязыке ДНК-операций, которая позволяет решать *задачу коммивояжера* (о поиске гамильтонова пути в графе).

1. *Ввести* (*N*)
2. $N \leftarrow B(N, s_0)$
3. $N \leftarrow E(N, s_0)$
4. $N \leftarrow (N, \leq 140)$
5. Для *i* от 1 до 5 выполнить $N \leftarrow +(N, s_i)$
6. *Обнаружить* (*N*)

По существу, этот алгоритм несет в себе полный перебор. В подходе Эдлмана нежелательный детерминизм обходится за счет *массового параллелизма* цепочек ДНК. *Комплементарность Уотсона–Крика* используется, чтобы обеспечить построение таких последовательностей ребер, которые действительно являются путями в графе *G*.

Недостатком этого алгоритма является то, что нахождение ответа для графа умеренных размеров может потребовать «большого» количества компьютерного времени, т. е. увеличение количества вершин в графе приводит к тому, что задача становится очень трудоемкой и требует использования суперкомпьютеров.

На основе анализа работы данного алгоритма можно сделать вывод, что молекулярный алгоритм, использованный в этом опыте, является малоэффективным и, как все известные алгоритмы, сводится к полному перебору. Как и для традиционных компьютеров, оптимизация алгоритма может расширить область применимости метода.

Оптимизация этого алгоритма может быть осуществлена с помощью различных *эвристических методов*, но, конечно, методы полного перебора, подобные вышеприведенному, применимы лишь тогда, когда можно эффективно получать все возможные пути, поэтому в опыте число молекул должно экспоненциально расти с увеличением размера задачи (в нашем примере – с числом вершин).

2. *Компьютерное моделирование ДНК-компьютера: стикерная модель молекулярных вычислений*.

Рассмотрим модель молекулярных вычислений, которая была введена S. Roweis и названа *стикерной моделью*.

Стикерная модель базируется на *парадигме комплементарности Уотсона–Крика*, благодаря которой цепочки ДНК можно использовать как физический носитель информации. По существу, стикерная модель представляет собой *память с произвольным доступом*, причем не требующей удлинения цепочек и допускающей (по крайней мере, теоретически) многократное использование.

Опишем метод представления информации цепочками ДНК (кодирование информации), основанный на принципе компли-

ментарности. В этом методе используется два основных типа одноцепочечных молекул ДНК: 1) запоминающие цепочки; 2) цепочки-наклейки (стикеры).

Запоминающей цепочкой длиной n нуклеотидов называется цепочка ДНК, содержащая k неперекрывающихся подцепочек, каждая из которых имеет длину m нуклеотидов. Таким образом, должно выполняться неравенство $n \geq mk$.

Подцепочки запоминающей цепочки должны существенно отличаться одна от другой, т. е. в любых двух из них, по крайней мере, на нескольких позициях должны стоять разные нуклеотиды. Это необходимо для того, чтобы каждый бит, который мы хотим закодировать, мог быть с уверенностью идентифицирован.

Стикером называется цепочка, состоящая из m нуклеотидов и комплементарная в точности одной из k подцепочек запоминающей цепочки.

Каждая подцепочка запоминающей цепочки может быть либо включена, либо выключена.

Будем говорить, что *подцепочка запоминающей цепочки включена*, если к ней присоединен соответствующий ей стикер. В противном случае, если к подцепочке не присоединен стикер, то *подцепочка выключена*[15].

Другими словами, существует правило, по которому стикер не может присоединиться к участку, перекрывающемуся с двумя выбранными подцепочками. Такое присоединение приводит к ошибке (например, изменению количества подцепочек в запоминающей цепочке).

Запоминающими комплексами[16] называются частично сдвоенные цепочки ДНК.

В дальнейшем для запоминающих цепочек, в которых включены некоторые подцепочки, будем использовать термин «запоминающий комплекс».

Запоминающие комплексы позволяют кодировать числа в двоичной форме записи:

1) включенные подцепочки соответствуют цифре 1;

2) выключенные подцепочки соответствуют цифре 0.

Таким образом, осуществляется кодирование информации с помощью запоминающих комплексов в стикерной модели.

В действительности для любого серьезного применения стикерной модели важно тщательное изучение наиболее подходящего кодирования с целью так скомбинировать микробиологическую реализуемость и теоретические преимущества, чтобы достичь оптимального компромисса между ними.

В опыте Эдлмана используют короткие одинарные цепочки, которые соединяются шаг за шагом так, что после каждого шага остаются липкие концы. Получающиеся при этом двойные цепочки не должны иметь одноцепочечных промежутков.

Однако в стикерных системах основой являются длинные одинарные цепочки, к которым присоединяются стикеры, создавая запоминающие комплексы.

При сравнении основных структур данных в опыте Эдлмана и в стикерных системах можно выделить идею конструирования двойных цепочек. В обоих случаях работает *парадигма комплементарности*.

Кроме основные структур данных выделяются операции, используемые в стикерных системах[17].

1. *Операция слить* имеет следующий смысл: две пробирки объединяются в одну. При этом запоминающие комплексы из двух пробирок на входе без изменений присоединенных к ним стикеров переходят в новое мультимножество, являющееся объединением двух данных.

2. *Операция разделить* создает по данной пробирке N и целому числу i, $1 \leq i \leq k$, две новых пробирки $+(N, i)$ и $-(N, i)$. Пробирка $+(N, i)$ (соответственно $-(N, i)$) состоит из всех запоминающих комплексов исходной пробирки N, в которых включена (соответственно выключена) i-я подцепочка.

3. *Операция включить* для данной пробирки N и целого i, $1 \le i \le k$, создает новую пробирку (N, i), в которой у каждого запоминающего комплекса из N включена i-я подцепочка. Это означает, что подходящий стикер присоединяется к запоминающему комплексу, если i-я подцепочка этого комплекса была выключена. Если i-я подцепочка уже была включена, то комплекс не изменяется.

4. *Операция очистить* для данной пробирки N и целого i, $1 \le i \le k$, создает новую пробирку (N, i), в которой у каждого запоминающего комплекса из N выключена i-я подцепочка. Это означает, что соответствующие стикеры удаляются из тех комплексов, где они имелись.

С помощью определенных выше операций можно организовать *вычисление в стикерной модели*.

Таким образом, данная *вычислительная парадигма*, ассоциированная со стикерной моделью, состоит в решении *трудных задач* путем полного перебора всех данных начальной пробирки (причем обработка данных осуществляется параллельно).

Одним из наиболее интересных примеров применения стикерной модели является атака на криптосистему *DES* (*DES* – классическая симметричная криптосистема с закрытым ключом).

Данный отбор содержания обучения предполагает выбор методов, форм и средств обучения.

В качестве основного метода обучения будем использовать метод демонстрационных примеров.

Формами обучения являются лекции и практические занятия (лабораторные работы, семинары).

Будем считать, что основным средством обучения для всех выделенных тем являются языки программирования LISP и C, причем выбор данных языков программирования связан с возможностями, предоставляемыми этими языками.

В качестве контроля умений и навыков студентов будем использовать контрольные работы, требующие использование только компьютера или сочетающие работу за компьютером с выполнением математических вычислений на бумаге (проверка контрольной работы осуществляется в процессе ее проведения).

В заключение отметим, что:

1) генетические алгоритмы часто используются совместно с нейронными сетями для решения определенных типов задач, поэтому имеет объединить эти темы в одном учебном курс;

2) выделенные вычислительные модели позволяют учить студентов другим подходам к пониманию понятия «вычислимость»;

3) техника ДНК-вычислений является частным случаем параллельных вычислений[18], что позволяет говорить о возможности включения данного аспекта в спецкурс «Параллельное программирование» или «Элементы многопоточного программирования»;

4) идея Эдлмана послужила выразительной демонстрацией возможностей молекулярного подхода к решению некоторого типа задач (миллиарды молекул в капле жидкости ДНК могут решить эту задачу быстрее «обычного» компьютера);

5) стикерная модель молекулярных вычислений позволяет дать представление об организации памяти с произвольным доступом;

6) идея построения ДНК-компьютера позволяет дать толчок к построению компьютеров с принципиально новой архитектурой (данная идея может использоваться в курсе «Архитектура вычислительных систем»);

7) ДНК-вычисления представляют собой физически реализуемую модель вычислений и никакого принципиального улучшения производительности по сравнению с машиной Тьюринга не дает[19].

Представленное в данной статье содержание обучения использовалось в рамках курса «Интеллектуальные информационные системы» для обучения студентов 1, 3 и 4-го курсов факультета информационных технологий РГПУ им. А. И. Герцена в 2005–2007 гг.

ПРИМЕЧАНИЯ

[1] *Воронов В. К., Подоплелов А. В.* Современная физика. М.: КомКнига, 2005.

[2] *Стин Э.* Квантовые вычисления. М.-Ижевск: НИЦ «Регулярная и хаотическая динамика», 2000.

[3] *Китаев А., Шень А., Вялый М.* Классические и квантовые вычисления. М.: МЦНМО, ЧеРо, 1999.

[4] *Люгер Д.Ф.* Искусственный интеллект: стратегии и методы решения сложных проблем. М.: Издательский дом «Вильямс», 2003.

[5] Там же.

[6] *Koza J. R.* Genetic Programming II: Automatic Discovery of Reusable Programs. Cambridge/MA: MIT Press, 1994.

[7] *Люгер Д. Ф.* Указ. соч.

[8] *Гладков Л. А., Курейчик В. В., Курейчик В. М.* Генетические алгоритмы. М.: ФИЗМАТЛИТ, 2006.

[9] *Рутковская Д., Пилиньский М., Рутковский Л.* Нейронные сети, генетические алгоритмы и нечеткие системы. М.: Горячая линия – Телеком, 2006.

[10] *Гасфилд Д.* Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология. СПб.: Невский Диалект; БХВ-Петербург, 2003. С. 590–591.

[11] *Паун Г., Розенберг Г., Саломаа А.* ДНК-компьютер. Новая парадигма вычислений. М.: Мир, 2004.

[12] *Стефанова Т. С., Флегонтов А. В., Швецкий М. В.* Элементы дискретной математики. Ч. I. Учеб. пособие для студентов математического факультета и факультета информационных технологий. СПб.: Интерлайн, 2007.

[13] Биология. Современный курс / Под ред. А.Ф.Никитина. СПб.: СпецЛит, 2005; *Бокуть С. Б., Герасимович Н. В., Милютин А. А.* Молекулярная биология: молекулярные механизмы хранения, воспроизведения и реализации генетической информации. Мн.: Выш. шк., 2005; *Вили Б., Детье Д.* Биология (Биологические процессы и законы). М.: Мир, 1974; *Гоппа В. Д.* Введение в алгебраическую теорию информации. М.: Наука. Физматлит, 1995; *Коничев А. С., Севастьянова Г. А.* Молекулярная биология. М.: Академия, 2003; *Эбелинг В., Энгель А., Файстель Р.* Физика процессов эволюции. М.: УРСС, 2001.

[14] *Паун Г., Розенберг Г., Саломаа А.* Указ. соч. С. 61–62.

[15] Там же. С. 74.

[16] Там же. С. 74.

[17] Там же. С. 76.

[18] *Нильсен Н., Чанг И.* Квантовые вычисления и квантовая информация / Пер. с англ. М.: Мир, 2006. С. 213–214.

[19] *Паун Г., Розенберг Г., Саломаа А.* Указ. соч.