ний из данных, связанного с развивающейся технологией кооперирующихся интеллектуальных (рациональных) агентов. В многоагентной системе Data Mining отдельные агенты отвечают за извлечение закономерностей и удовлетворение информационных потребностей пользователя. При этом основные затраты связаны именно с поиском закономерностей (основной упрек Data Mining со стороны пользователей состоит в том, что эти средства слишком медлительны [1]). Дополнение интерфейсных агентов средствами синтеза МС позволит не только разгрузить агентов-исследователей за счет снижения размерности задач, но и облегчить их кооперацию за счет автоматического распределения задач в соответствии со связями в схеме БД.

СПИСОК ЛИТЕРАТУРЫ

1. Devlin B. Data warehouse: from architecture to implementation. — Reading: Addison Wesley Longman, 1997. — 432 p.
2. Перевозчикова О. Л., Ющенко Е. Л. Диалоговые системы. — Киев: Наук. думка, 1990. — 184 с.
3. А χ о А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции: В 2 т. — М.: Мир, 1978. — 612 с.
4. Кулиш Е.Н., Тульчинский В.Г. Метод парных грамматик для структурного синтеза и анализа маршрутных схем // Представление знаний в информационных технологиях. — Киев: Ин-т кибернетики НАНУ, 1995. — С. 44-57.
5. Тульчинский В.Г. Алгебро-грамматический подход к проектированию интерфейса // Кибернетика и системный анализ. — 1996. — № 6. — С. 169-183.
6. Кокорева Л. В., Перевозчикова О. Л., Ющенко Е. Л. Диалоговые системы и представление знаний. — Киев: Наук, думка, 1992. — 448 с.
7. Костин А.Е. Принципы моделирования сложных дискретных систем. — М.: МИЭТ, 1984. — 140 с.
8. Котов В.Е. Сети Петри. — М.: Наука, 1984. — 160 с.

Поступила 10.12.2001

УДК 681.3.06

Н.Н. ГЛИБОВЕЦ, С.А. МЕДВИДЬ

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И ИХ ИСПОЛЬЗОВАНИЕ ДЛЯ РЕШЕНИЯ ЗАДАЧИ СОСТАВЛЕНИЯ РАСПИСАНИЯ

Ключевые слова: *генетический алгоритм, эволюционные исчисления, режимы репродукции, мутация хромосом, отбор хромосом, оптимизационная задача.*

Генетические алгоритмы рассматриваются как один из удачных подходов к решению оптимизационных задач. Наблюдается заинтересованность в использовании принципа генетического алгоритма [1-6], предложенного Дж. Холландом (Мичиганский университет) в 1975 г. [7]. Генетические алгоритмы имеют большое число реализаций с исследованными методами оптимизации, улучшения и настройки, объединенными единой идеей совершенствования алгоритма.

Функционирование любого генетического алгоритма основано на принципе эволюции, впервые описанном Ч. Дарвином. Существует математическое доказательство, которое объясняет причину эффективности такого подхода (так называемая теорема схем — Schema Theorem) [8, 9]. Как и в

природной эволюционной модели, поиск оптимального решения ведется параллельно во множестве вариантов. Поэтому в генетических (или эволюционных) алгоритмах терминология в основном аналогична теории эволюции Дарвина.

В отличие от обычных алгоритмов, где поиск оптимального решения ведется линейно, в одном варианте, в эволюционных алгоритмах имеем множество альтернативных решений, которые в процессе работы алгоритма конкурируют между собой и изменяются по определенным правилам. В итоге выбирается наилучшее решение, которое считается результатом работы алгоритма.

Продемонстрируем практическое применение генетических алгоритмов на примере решения *NP*-полной оптимизационной задачи составления расписания занятий в учебном заведении. Необходимость нахождения наиболее удачного решения такой задачи возникает в любом учебном заведении, поскольку требуемое для решения *NP*-полной задачи время возрастает до астрономических величин даже при незначительном увеличении объема и усложнении структуры входящих данных.

ПРИНЦИП РАБОТЫ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Рассмотрим основной принцип работы классического генетического алгоритма, который условно можно разделить на три этапа: инициализация, моделирование эволюционного процесса и окончание работы.

Инициализация. Создается популяция (набор) значительного числа решений-кандидатов. В общем случае при их генерации не накладывается никаких условий — они полностью случайны, а их дальнейшая оптимизация проводится в процессе работы второй части алгоритма. Однако в реальных реализациях генетических алгоритмов часто используют различные эвристики уже во время создания начальной популяции решений. Эти эвристики специфичны для каждой конкретной оптимизационной задачи, но основное их предназначение — увеличить среднее значение оптимальности уже среди начальных решений. Это, в свою очередь, позволяет существенно уменьшить время работы основной части алгоритма.

В классических реализациях генетического алгоритма альтернативные решения задаются в виде набора битовых строк, каждая из которых называется хромосомой, а бит — соответственно геном. Способ представления хромосом (например, в виде битовых строк) не имеет существенного семантического значения, кроме влияния на продуктивность конкретной программной реализации алгоритма.

Моделирование эволюционного процесса. Следующая фаза алгоритма наиболее длительна и важна, поскольку к начальной популяции применяются эволюционные законы. Как известно, основными принципами эволюции есть наследование, мутации и отбор. Именно эти правила итеративно применяются к начальной популяции, которая под их действием начинает постоянно изменяться. Моделировать природные эволюционные процессы оказалось удобно, рассматривая каждое решение как хромосому, которая состоит из неделимых частей — генов. Поэтому на этапе реализации решения представляются в виде хромосом, а в алгоритме законы эволюции реализуются следующим образом.

• К каждому гену всех хромосом первого поколения применяется оператор мутации, который с малой вероятностью может изменить ген на любую его модификацию.

• В хромосомах первого поколения происходит обмен участками. Таким образом формируются хромосомы второго поколения. При этом вероятность участия в формировании следующего поколения тем больше, чем выше оценка хромосомы первого поколения.

• После образования такого же числа хромосом во втором поколении, что и в первом, второе поколение полностью замещает первое и итерация повторяется.

Интуитивно ясно, что можно подобрать такие параметры работы второй фазы генетического алгоритма, при которых популяция решений будет постепенно улучшаться, т. е. становиться оптимальнее. Тип репродукции, используемый в работе генетического алгоритма, определяет способ, по которому осуществляется смена поколений. Различают два основных типа репродукции — генерационный (generation — поколение) и «стойкий» режим (steady state).

Генерационный тип репродукции означает полную замену популяции следующим поколением на каждой итерации алгоритма [7]. Пусть количество хромосом в популяции равно N. Тогда на каждой итерации алгоритма выбирается $N/2$ пар родительских хромосом, с помощью которых будут сгенерированы N хромосом-детей. Полученное таким образом поколение полностью замещает родительское поколение.

При стойком режиме репродукции в отличие от генерационного режима каждая итерация повторяется только для одной пары хромосом, т. е. предыдущее поколение не заменяется новым. Сгенерированные хромосомы-дети возвращаются в исходную популяцию, замещая имеющиеся в ней наихудшие решения [10].

Кросоверы и мутации. Основная функция мутаций и обменов хромосом определенными частями (так называемых кросоверов) — это перебор вариантов решений. Конкретная реализация алгоритма должна обеспечить потенциальную возможность получения любого решения из всего пространства возможных решений только с помощью кросоверов и мутаций. Действительно, кросовер позволяет эффективно перебирать достаточно большое число вариантов (поскольку начальные решения формировались случайным образом). Основная задача мутации — обеспечить возможность перехода к тем решениям, к которым невозможно перейти с помощью кросоверов из начального состояния популяции. Именно поэтому вероятность возникновения мутации — это предмет особого, исследования [11—14]. С одной стороны, мутация не должна вносить хаос в популяцию, разрушая «хорошие» решения (такая ситуация будет наблюдаться при большой вероятности возникновения мутации). Так, ярким примером неприемлемости высокого уровня мутации может служить факт, что при высоком уровне радиации в природе все живое обычно вымирает уже через несколько поколений. С другой стороны, мутация должна обеспечить широкие возможности перебора решений. Недаром в природе мутация играет ключевую роль; без нее эволюция была бы невозможной.

Для генерации следующего поколения используется семейство генетических операторов кросовера, основное предназначение которого — генерация следующего поколения хромосом-детей путем рекомбинации частей хромосом-родителей через обмен сегментами между парами хромосом. Часто в генетических алгоритмах используется схема «два родителя», но в некоторых исследованиях [3] приводятся примеры генетических алгоритмов, когда в генерации хромосом-детей принимают участие больше двух родителей. Наиболее часто используются типы кросовера: одноточечный, двухточечный, N-точечный и однородный. При одноточечном кросовере выбираются две хромосомы и генерируется случайная точка перекрещивания. Исследования одноточечного кросовера продолжаются [8, 15]. Этот процесс изображен на рис. 1.

Двухточечный кросовер (рис. 2) отличается от одноточечного только тем, что случайным образом выбираются две точки перекрещивания и хромосомы обмениваются участками между этими двумя точками.

N-точечный кросовер — это обобщение одно- и двухточечного кросоверов путем определения N точек перекрещивания. Этими точками хромосомы разбиваются на $N+1$ участок и обмениваются только четными (или только нечетными) участками.

Однородный кросовер (рис. 3) впервые описан в [4]. Каждый ген первого из родителей имеет 50-процентный шанс обменяться местами с соответствующим геном второго родителя.

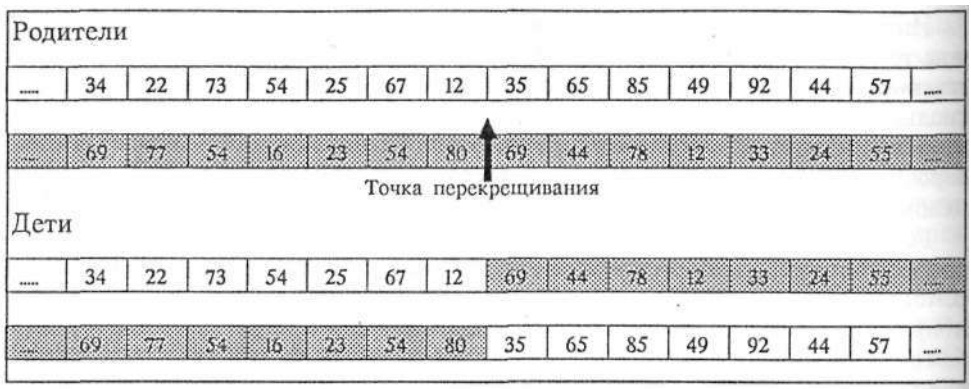

Рис. 1. Одноточечный кросовер

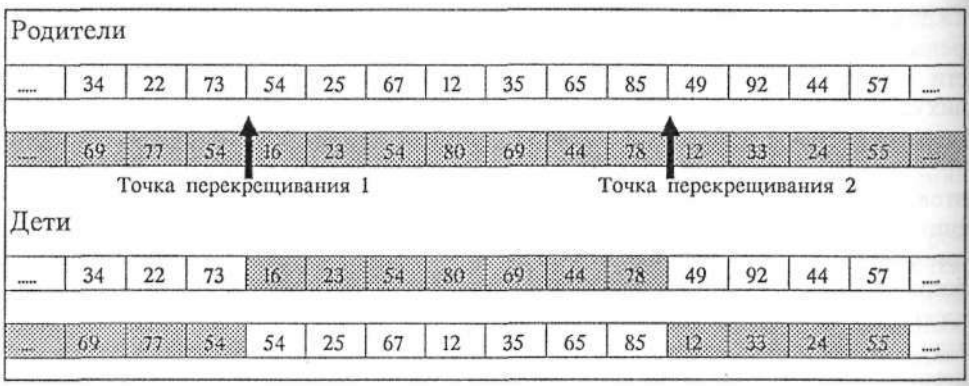

Рис. 2. Двухточечный кросовер

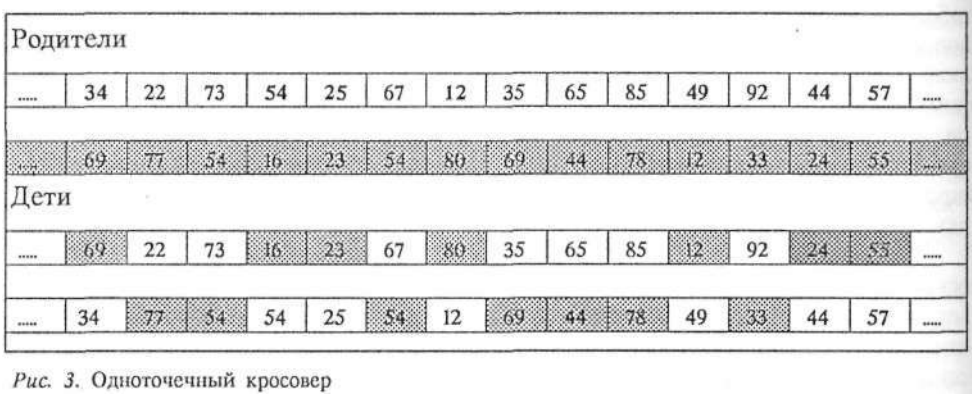

Рис. 3. Одноточечный кросовер

В одной из родительских хромосом жирным шрифтом выделены гены-кандидаты на кросовер. Выбор проводился случайным образом с 50-процентной вероятностью. Хромосомы-дети образовались из хромосом-родителей путем обмена генами, которые были избраны на предыдущем шаге.

Существуют еще два генетических оператора, влияющие только на одну хромосому, т. е. не принимающие участия в генерации следующих поколений решений. Такими операторами являются мутация и инверсия, предназначенные для случайной модификации одного из расписаний. Поскольку начальные расписания генерируются случайным образом, то такая генерация и последующие кросоверы не обязательно будут обеспечивать поиск по всему пространству решений. Операторы мутации и инверсии именно и предназначены для внесения в популяцию случайных решений, к которым сложно или невозможно прийти, применяя только операторы кросовера.

Вероятность возникновения мутации и инверсии должна быть сравнительно малой, но сбалансированной. С одной стороны, она должна генери-

ровать новые решения, а с другой, — не вносить хаос в популяцию. Неудачные решения, порожденные мутацией (уменьшающие значение оценочной функции хромосомы), отсеятся в процессе отбора, а удачные решения будут иметь больше шансов на жизнь благодаря тому же отбору.

Оператор мутации изменяет один из генов хромосомы на случайное значение. Применяется он ко всем генам, но с малой вероятностью срабатывания. После срабатывания мутации должна быть пересчитана функция оценки хромосомы.

Действие оператора инверсии состоит в том, что случайным образом выбираются две позиции на хромосоме и участок между ними инвертируется таким образом, что каждый ген заменяется инверсным. Как видим, такая инверсия не влияет на функцию оценки хромосомы. Однако кросовер, который впоследствии будет применяться к хромосоме, породит совсем других хромосом-детей. Оператор инверсии должен применяться с вероятностью, существенно меньшей единицы.

Отбор. Такой важный принцип теории эволюции как отбор реализуется в алгоритме путем обеспечения участия хромосом предыдущего поколения в формировании следующего поколения с разной вероятностью. Благодаря этому более оптимальное решение (хромосома) имеет больше шансов на продолжение рода. Однако это не означает, что у наименее оптимальных решений вообще нет шансов. По аналогии с подобными процессами в природе неоптимальная хромосома может принять участие в формировании следующего поколения, в то время как самая оптимальная останется без потомков. Но вероятность этого довольно мала, поскольку в таком случае мы осознанно идем на возможность уничтожения наиболее оптимальных решений. Как и в природе, имеет место ситуация, когда наименее оптимальное решение содержит в себе часть оптимального и в последующих поколениях именно из него можно сформировать наиболее оптимальное решение.

Другим осложнением при построении решения является множество ограничений, которые должны соблюдаться в оптимальном результате. Формально вводится два типа ограничений — жесткие и нежесткие.

При нарушении жестких ограничений решение получается неправильным, т. е. невозможным для практического применения. Примером нарушения жесткого ограничения является составление такого расписания, согласно которому один лектор должен одновременно читать три лекции в трех аудиториях.

Жесткие ограничения должны задаваться в конфигурации алгоритма и не влиять на его реализацию. Количество жестких ограничений должно быть небольшим, поскольку их нарушение означает полную невозможность построения решения, т.е. создаются условия, когда ни одно из возможных решений не будет удовлетворять всем условиям.

Исходя из этого вводятся нежесткие ограничения, которые в отличие от жестких определяются не невозможностью конкретного события, а его нежелательностью. Причем, разные нежесткие ограничения задают степень нежелательности. Так, существуют два альтернативных решения, одно из которых нарушает нежесткое ограничение на существование «окон» (свободное время между занятиями) в расписании определенной группы (две группы в расписании имеют по одному «окну» за целую неделю), а во втором решении в трех аудиториях занятия проводятся долго и без перерывов. Необходимо выбрать лучшее из описанных расписаний. Для формализации такого выбора для каждого нежесткого ограничения вводятся весовые коэффициенты, влияющие на оценку определенного варианта расписания. В этом случае лучшим считается расписание, у которого функция оценки имеет большую оценку.

Итак, оператор отбора имеет большое значение для генетических алгоритмов. Различают несколько видов отбора: турнирный, отсекающий (truncation selection), линейный и экспоненциальный по рангу (ranking selection), пропорциональный. Все типы отбора проводятся на основании

сравнения двух хромосом и определения лучшей из них. Такое оценивание осуществляется с помощью функции оценки.

При турнирном отборе случайным образом проводится отбор t хромосом из родительской популяции и среди них выбирается лучшая. Число повторений процедуры соответствует N — количеству хромосом в популяции. Часто турниры проводятся между двумя индивидуумами (бинарные турниры). При t индивидуумах получим общий случай турнирного отбора. Такой алгоритм отбора можно довольно эффективно реализовать, поскольку для его работы не требуется сортировки начальной популяции. Временная оценка алгоритма равна $O(N)$. Анализ такого метода отбора впервые проведен в [16], а математический анализ — в [9].

При использовании отсекающего отбора с порогом T выбираются только T лучших индивидуумов, причем каждый из них имеет одинаковую вероятность быть избранным [1]. Поскольку для такого алгоритма отбора необходимо сортировать всю популяцию, алгоритм отсекающего отбора имеет временную оценку $O(N \cdot \ln N)$.

Пропорциональный метод отбора предложен в первой реализации генетического алгоритма в 1975 г. Джоном Холландом [7]. Вероятность быть избранным для каждого индивидуума прямо пропорциональна значению его оценки. Такой алгоритм можно реализовать с временной оценкой $O(N)$.

Отбор по рангу, впервые предложенный Бейкером (Baker), избавляет от серьезных недостатков пропорционального отбора [2]. В этом случае все хромосомы начальной популяции сортируются по их оценке и наилучшей из них присваивается ранг N, а худшей — ранг 1. Вероятность отбора прямо пропорциональна полученному таким образом рангу хромосомы. Экспоненциальный отбор по рангу отличается от линейного тем, что вероятности экспоненциально зависят от ранга. Основание экспоненциальной функции — параметр $0 < c < 1$, который не изменяется в течение всей работы алгоритма. Итак, эти методы также требуют сортировки популяции, поэтому их временная оценка определяется временной оценкой алгоритма сортировки, как известно, составляющей $O(N \cdot \ln N)$.

Необходимо формализовать оценку хромосом в популяции. Все методы отбора основаны на сравнении хромосом между собой согласно оценки. Функцию оценки вводят таким образом, чтобы большие значения этой функции соответствовали меньшему количеству ограничений, нарушаемых решением.

Завершение работы генетического алгоритма. При моделировании эволюционного процесса решения становятся все более оптимальными и постепенно уменьшается число конфликтов. В отличие, например, от алгоритма полного перебора генетический алгоритм не гарантирует нахождения наиболее оптимального решения. Можно быстро найти хорошее решение, но оно не всегда будет наиболее оптимальным. Однако большинство реальных оптимизационных задач не требуют нахождения самого оптимального решения.

В связи с этой особенностью генетических алгоритмов появляется проблема определения условия остановки основного цикла алгоритма и выдачи окончательного решения на выход. Обычно для этого определяют пороговое значение функции оценки оптимальности решений. Если одна или несколько хромосом из популяции достигли заданного уровня оптимальности, алгоритм останавливается, а на выход подается наилучшее решение.

В процессе нахождения решения генетическим алгоритмом наиболее важной является постоянная поддержка правильности (допустимости) решений в течение работы алгоритма, т.е. поддержание хромосом такими, чтобы они не нарушали жестких ограничений. Это экономит процессорное время, которое в противном случае тратилось бы на расчет и поддержание недопустимых решений. Каждый из описанных генетических операторов (кроме инверсии) может повлиять на хромосому таким образом, чтобы в ней нарушались жесткие ограничения. Поэтому после выполнения такого оператора нужно проверять новообразованную хромосому на наявность нарушений

жестких ограничений и при появлении таковых совершать откат, возвращая хромосому в предыдущее состояние. После такого отката алгоритм может действовать, следуя одной из двух стратегий: либо не повторять использование генетического оператора, либо повторять его определенное число раз и прекратить попытки, убедившись, что ни один из повторов не дал правильного расписания.

ЗАДАЧА СОСТАВЛЕНИЯ РАСПИСАНИЯ ЗАНЯТИЙ

Поскольку задача составления расписания занятий является *NP*-полной [17] даже для такой задачи, как расположение N экзаменов в T временных слотах, число возможных решений равняется T^N. Это число достигает больших значений для реальных условий учебного заведения, насчитывающего небольшое количество учащихся. Но задача составления расписания экзаменов по сравнению с задачей составления расписания занятий является существенным упрощением. Например, каждый экзамен четко привязан к определенной группе студентов и определенному преподавателю, чего нельзя сказать об обычном занятии по какому-либо предмету. Ведь каждый предмет может читать группа преподавателей, а слушает его некоторое количество групп студентов. Таким образом, определение того, какой преподаватель у какой группы ведет занятия, — это тоже часть расписания.

Введем следующие жесткие ограничения: один лектор должен в одно время читать лекцию лишь в одной аудитории; одна группа в любой момент может иметь только одно занятие; одна аудитория в любой момент может использоваться только под одно занятие. Ограничимся этим списком для обеспечения реальности построения расписаний. Например, к разряду жестких можно было бы отнести следующее: после проведения каждого занятия аудитория должна быть свободной в течение двух временных слотов (пар) (скажем, с целью проветривания). Но в этом случае, если количество аудиторий небольшое, а количество занятий существенно, может случиться, что все возможные расписания будут нарушать жесткие ограничения, т.е. задача вообще не будет иметь решения.

Среди нежестких ограничений выделим следующие: как можно меньше «окон» в индивидуальных расписаниях преподавателей и студентов; занятие не может проводиться, если число студентов в группе больше количества мест в аудитории.

Итак, имея на входе набор таких данных, касающихся частей расписания на определенный период времени (семестр, триместр), как списки студентов, преподавателей, предметов, аудиторий, связи между ними, количество учебных недель и часов для каждого предмета, а также наборы жестких и нежестких ограничений, алгоритм должен выдать расписание, вообще не нарушающее жестких ограничений и максимально отвечающее нежестким ограничениям.

Адаптация генетического алгоритма к задаче составления расписаний. При применении генетических алгоритмов к решению реальной задачи составления расписаний необходимо определить прежде всего внутренний формат данных, который, с одной стороны, должен быть удобным для работы алгоритма, а с другой, — учитывать все особенности внешних данных.

Входные данные целесообразно сохранять во внешней базе данных в удобном формате для наполнения и представления. Это позволит подключить модуль составления расписания к более сложной системе управления учебным процессом университета. Однако в работе алгоритма использование такого формата неприемлемо ввиду малой скорости доступа к таким данным и их структуре, не отвечающей внутренним структурам алгоритма. Поэтому перед началом работы алгоритма необходимо трансформировать внешние данные во внутренний формат, а по окончании — выполнить обратную трансформацию.

В рассматриваемом случае альтернативные решения можно представить битовыми строками, построив определенное отображение. Но работать с

таким отображением неудобно, так как оно требует значительных ресурсов (например, при подсчете функции оценки для каждого расписания битовые строки постоянно приходилось бы конвертировать в реальные данные). Поэтому обратимся к более естественной реализации хромосом.

Ген — это неделимая единица, которая входит в состав хромосомы и кодирует одну аминокислоту. Аналогично в нашей презентации хромосомы ген кодирует один кортеж расписания — тройку (группа, аудитория, время). Группа определяет совокупность некоторых других данных: учебный курс, для которого сформирована группа (например, Химия-1, Английский язык-2); тип занятия (лекция, лабораторная работа, практическое занятие). Каждый тип занятия определяет список возможных преподавателей и аудиторий для проведения занятия (эти данные поступают извне): список студентов, принадлежащих группе; преподаватель, читающий в данной группе.

Эти данные можно разделить на две категории:

• статические данные, не изменяющиеся в процессе работы алгоритма, — учебный курс и тип занятия для каждой группы;

• динамические данные, фактически входящие в само расписание, — список студентов в каждой группе, преподаватель этой группы. Список студентов может изменяться (если это не ограничено извне). Например, для четырех практических групп по Химии-1 непринципиально, в какой группе будет тот или иной студент, поэтому в составление расписания также входит генерация списков этих групп. Аналогично не имеет значения, какой преподаватель из некоторого набора будет читать у той или иной группы.

Статические данные будем идентифицировать кодом группы, а динамические — номером ее модификации. Итак, каждая группа представлена в гене статической и динамической частями. Статическую часть составляют предмет, тип занятия, номер группы, а динамическую — текущая модификация группы (лектор, список студентов). В пределах одного расписания для групп с одним кодом (сформированных для одного предмета и типа занятия), но с разными номерами модификация группы будет одинакова.

Рассмотрим пример. Пусть по Химии-1 определено шесть групп для практических занятий. Идентифицируются эти группы следующим образом: Х1П-1, Х1П-2, Х1П-3, Х1П-4, Х1П-5, Х1П-6. Тут Xl означает Химия-1, П — практические занятия, а цифра — порядковый номер группы. Пусть Химию-1 слушает шестьдесят студентов, т.е. в каждой группе будет по десять человек. Практические занятия ведут три преподавателя — каждый в двух группах. В данном случае количество групп, название курса и тип занятия — статическая информация. Конкретный список студентов в каждой группе, а также соответствие преподаватель — группа есть динамическая информация.

Итак, группа будет представляться в гене тройкой: код группы (например, Х1П); номер группы (например, 3); номер модификации — число, определяющее списки студентов в каждой группе, а также какой преподаватель в какой группе читает (строится соответствие между номером модификации и списками студентов в группах, а также между группой и преподавателем).

По формулам комбинаторики количество возможных способов, когда три преподавателя могут вести занятия в шести группах, имея по две группы, равно девяносто. Если считать постоянными списки студентов в группах, то номер модификации можно представить семью двоичными разрядами ($2^7 = 128$). Можно задать однозначное соответствие между этим числом и преподавателем, который читает в каждой из шести групп. При этом не будут нарушены начальные ограничения (каждый преподаватель читает в двух и только в двух группах).

Аналогично можно сопоставить номер модификации со списком студентов. Вариантов будет значительно больше, но и их можно представить сравнительно

небольшим количеством двоичных разрядов. В приведенном примере количество вариантов разных списков групп определяется выражением

$$N = \frac{60!}{(10!)^6} \approx 3,64 \cdot 10^{42} \approx 2^{141,4},$$

т.е. для представления номера модификации списков групп необходимо 142 двоичных разряда, а с учетом вариантов с преподавателями для номера модификации (в приведенном примере) необходимо 149 двоичных разрядов.

Итак, номер модификации группы нельзя представить в машинной реализации как обычное целое число, поскольку оно может принимать большие значения. Поэтому будем представлять его битовым полем, имеющим однозначное отображение в списки студентов и сопоставление преподавателей группам. Это поле может достичь размера нескольких килобайт. Однако оно не должно находиться в самом гене. Номер модификации группы определяется не для каждого занятия в расписании (ген представляет собой именно занятие), а для кода группы (в нашем примере Х1П). Значит, все группы в расписании, начинающиеся на Х1П, должны иметь одинаковый номер модификации.

Поскольку в рассматриваемом примере шесть подобных групп и каждая из них имеет несколько занятий в течение недели, то нецелесообразно сохранять битовые поля модификации группы в самой хромосоме. Во-первых, это занимало бы значительный объем памяти, а во-вторых, необходимо было бы постоянно следить за синхронизацией этих битовых полей, изменяя все остальные при модификации одного из них.

Поэтому код группы лучше сохранять в хромосоме в виде указателя на структуру, которая содержала бы всю информацию о группе (например, принадлежность ее к определенному учебному курсу), а также битовое поле, которое задает номер модификации группы.

Для каждого кода группы создается только одна такая структура, на которую указывают соответствующие указатели из хромосомы. Поэтому при внесении изменений в один из генов (например, в группу Х1П-5), будут автоматически изменены номера модификаций всех остальных подобных групп (от Х1П-1 до Х1П-6).

Представление хромосомы. Каждая хромосома — это набор генов. Поскольку количество занятий известно еще до начала работы алгоритма, хромосомы можно представить в виде динамического массива, каждый элемент которого есть структура, являющаяся геном, как показано на рис. 4.

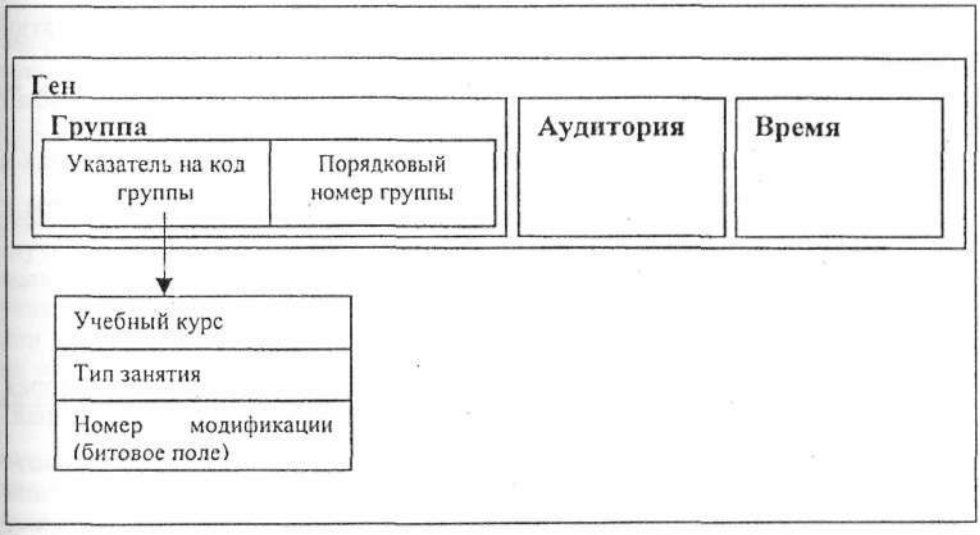

Рис. 4. Представление гена в бинарном виде

Структура, содержащая учебный курс, тип занятия и номер модификации, — одна на несколько групп с одинаковым кодом.

Итак, один ген в хромосоме — это одно занятие в расписании. Сама хромосома имеет следующий вид: ген 1, ген 2, ген 3, ... , ген *N*.

Инициализация. Генетические алгоритмы не требуют выполнения каких-либо правил при построении начального поколения решений. Расписания вначале могут быть конфликтными, с нарушениями жестких ограничений. Естественно, это упрощает процесс генерации начального поколения, определяя для него только один простой в реализации метод — генерацию полностью случайного расписания. Но в дальнейшем, на второй фазе алгоритма, такой способ создания начальной популяции вызывает цепочку проблем.

Во-первых, необходимо определять весовые коэффициенты и для жестких ограничений, причем эти коэффициенты должны быть намного большими, чем те, которые определяются для нежестких ограничений. Но выбрать правильное соотношение сложно, к тому же это требует длительных экспериментов. Во-вторых, значительный ресурс будет тратиться на расчеты, изменение и поддержку заведомо недопустимых расписаний. Существует вероятность того, что эти недопустимые решения станут бесконфликтными в течение работы алгоритма, но она ничтожно мала для полностью случайных начальных решений.

Поэтому будем придерживаться бесконфликтности расписаний на всех этапах работы алгоритма [18]. Это, естественно, усложнит алгоритм, но выигрыш будет несравнимо большим — не будет тратиться лишнее процессорное время, а также не нужно вводить весовые коэффициенты для жестких ограничений, что существенно упростит настройку алгоритма.

На этапе начальной генерации бесконфликтный в отношении жестких ограничений алгоритм построения расписания примет следующий вид.

1. Определение состава групп.

1.1. Из входных данных формируется список студентов для каждого учебного курса.

1.2. Содержимое списков перемешивается.

1.3. Из входных данных определяются наборы групп (лекционные, практические и т.д.).

1.4. Для каждого набора вычисляется количество групп и их размер.

1.5. Списки студентов заполняются в группы последовательно.

2. Назначение преподавательского состава, отвечающего за группы.

2.1. По входным данным определяется, какие курсы и виды занятий может вести преподаватель.

2.2. Случайным образом формируются соответствия преподаватель.↔ группа.

3. Определение времени прохождения занятия.

3.1. Группы расставляются случайным образом во временные слоты с учетом следующих требований:

3.1.1. Нет жестких конфликтов по преподавателям.

3.1.2. Нет жестких конфликтов по студентам.

4. Определение аудитории.

4.1. На каждое полученное занятие определяется случайная аудитория из списка доступных так, чтобы не было конфликта по занятости аудиторий.

Поскольку расписаний, сгенерированных таким образом, должно быть много и все они разные, в алгоритме присутствуют случайные функции (см. пп. 1.2, 2.2, 3.1, 4.1). На основании этого можно получить большое количество неоптимизированных, но бесконфликтных (относительно жестких ограничений) расписаний.

Далее из сгенерированных расписаний формируются хромосомы, составляющие начальную популяцию, которая в дальнейшем будет видоизменяться. Заметим, что согласно предложенной структуре данных информация о списках групп и преподавателей задается в битовых полях. На практике они являются двоичным представлением чисел, означающих номер модификации группы. Этим номерам ставится в однозначное соответствие определенный список всех подгрупп, а также набор соответствий преподаватель ↔ группа.

Основная фаза работы алгоритма. Итак, после инициализации имеем структуру данных, представляющую набор хромосом и удобную для проведения над ней действий основной фазы работы алгоритма — эмуляции эволюционных процессов. В этой фазе алгоритм предполагает итеративное выполнение следующих действий над хромосомами-расписаниями:

- отбор хромосом для репродукции;
- применение операторов изменения к хромосомам-кандидатам;
- репродукция нового поколения;
- проверка условий окончания работы алгоритма.

Эти шаги применяются к популяции решений до тех пор, пока проверка условий окончания работы алгоритма не приведет к позитивному результату. Опишем это подробнее для случая применения к структуре данных, полученной после инициализации алгоритма составления расписаний.

Отбор хромосом. Как было замечено выше, существует несколько методов отбора хромосом для участия в процессе генерации следующего поколения. Все они базируются на том или ином подходе к определению придатности хромосомы к репродукции. Эта придатность определяется с помощью функции оценки и непосредственно влияет на вероятность участия хромосомы в генерации решений-потомков.

Функции оценки и стоимости. Функция оценки определяет степень придатности хромосомы к формированию следующих поколений: $E = \frac{1}{1+C}$, где E — оценка хромосомы, C — стоимость. Здесь функция стоимости $C = \sum w_i * n_i$, где i — номер нежесткого ограничения, w_i — его весовой коэффициент, n_i — число нарушений таких ограничений. Функция оценки принимает значения от 0 до 1, причем она тем больше, чем меньше нарушено нежестких ограничений. Алгоритм направлен на нахождение решения с наибольшим значением этой функции.

Процесс отбора. Реализовано несколько функций отбора. Все они имеют стандартный интерфейс передачи и принятия параметров и отличаются только способом самого отбора. С помощью конфигурации можно выбрать одну из функций отбора — поочередно или с определенной вероятностью. Это открывает дополнительные возможности в процессе настройки алгоритма.

Типы отбора. Определим некоторые особенности методов отбора в конкретной реализации алгоритма составления расписаний.

Турнирный отбор в реализации алгоритма составления расписаний совершается для пар решений. Значит, лучшая хромосома выбирается из пары случайно выбранных хромосом. В зависимости от типа репродукции, используемого при вызове функции, функция турнирного отбора возвращает указатель на одну хромосому (стойкий режим репродукции — см. далее) или массив (генерационный тип репродукции), который содержит указатели на половину хромосом из популяции. Эти хромосомы являются наилучшими кандидатами на генерацию нового поколения.

Отсекающий отбор возвращает T укататателей на хромосомы при стойком режиме репродукции, при этом T — существенно меньше общего количества хромосом в популяции. При генерационном типе репродукции число возвращаемых указателей равно половине хромосом в популяции.

Пропорциональный отбор и его улучшенная модификация — отбор по рангу, как и функция, совершающая отбор по турнирному методу, возвра-

щают указатель на одну хромосому либо количество указателей, равное половине хромосом популяции.

Режимы репродукции. При реализации алгоритма составления расписаний предусматривается его работа в двух режимах репродукции: генерационном и стойком.

Генерационный режим репродукции предвидит полную замену предыдущего поколения новым на каждой итерации. При этом выполняется следующая последовательность действий.

• Дважды вызывается функция отбора, которая всякий раз возвращает массив указателей на хромосомы.

• Попарно для элементов массивов применяется оператор генерации следующего поколения, который формирует две хромосомы, записываемые в новый массив.

• Сформированный массив хромосом заменяет собой предыдущее поколение.

Заметим, что при вызове функции отбора возвращается массив указателей, некоторые из них могут указывать на одну и ту же хромосому. Значит, одна хромосома несколько раз может принимать участие в формировании нового поколения. В то же время это означает, что некоторые хромосомы из родительского поколения исчезнут, не оставив после себя потомков.

Стойкий режим репродукции близок к естественному, поскольку предвидит одновременное существование хромосом, относящихся к нескольким поколениям. Последовательность действий следующая.

• Дважды вызывается функция отбора. В этом случае она возвращает один или небольшое количество (описание типов отбора см. выше) указателей на хромосомы.

• Применяется оператор генерации нового поколения.

Хромосомы-потомки заменяют собой наихудшие хромосомы из популяции. Заметим, что в этом режиме функция отбора возвращает наборы указателей, которые указывают на физическую хромосому только один раз, т.е. в данном случае нет повторений, как при генерационном режиме репродукции.

Генетические операторы. Определим особенности реализации генетических операторов для описанной выше структуры данных, которая используется в алгоритме составления расписаний.

Кросовер в алгоритме составления расписаний. Ген был определен как тройка (группа, аудитория, время). Но это не означает, что точки кросовера в хромосоме выбираются только по границам генов — между компонентой «время» предыдущего гена и компонентой «группа» следующего, не разрывая их. Наоборот, для более полного перебора возможных решений целесообразно, чтобы кросовер также разрывал гены родительских хромосом, когда одна часть гена переходит первому потомку, а другая — второму. Таким образом, имеем гарантированный способ получения новых генов даже с помощью кросовера в добавление к мутации и другим однохромосомным генным операторам (о чем речь пойдет далее).

Рассмотрим особенности представления учебной группы, как части гена. В отличие от времени и аудитории, которые неразделимы, точка кросовера может отделить код группы (например, Х1П-2 — см. предыдущие примеры) от номера ее модификации. Это означает, что в хромосоме-потомке код какой-либо группы может быть унаследован от одного родителя, а номер модификации — от другого. Для обеспечения такой возможности необходимо использовать номера модификаций как битовые поля одной размерности — наибольшей среди всех групп. Для групп, которые имеют меньшую размерность, задается неоднозначное соответствие, когда нескольким значениям номера модификаций группы отвечает одна ее модификация. Такой подход

к кодированию модификаций групп позволит отделять кросовером коды групп от их модификаций.

Оператор мутации изменяет случайным образом одну из трех компонент одного гена в хромосоме. Значит, после мутации у одного из генов хромосомы изменяется группа, аудитория или время занятий.

Чтобы хромосома-решение продолжала иметь смысл, компонента, поддающаяся такому изменению, заменяется не полностью случайным значением, а произвольным значением из определенного набора. Так, под влиянием оператора мутации на группу новое значение может быть заменено на любую другую доступную группу.

Завершение работы алгоритма. Проверка на соответствие условию завершения проводится в конце каждой итерации основной фазы алгоритма составления расписания занятий. Как и в большинстве других генетических алгоритмов, остановка происходит по достижении порогового значения функции оценки, которое задается на этапе конфигурации. На выход подается одно или несколько альтернативных расписаний, которые имеют значение функции оценки, большее или равное заданному.

В заключение следует отметить, что тестирование генетического алгоритма на примере составления расписания для факультета информатики Национального университета «Киево-Могилянская академия», имеющего довольно сложный профиль предпочтений и преференций, продемонстрировало практическую приемлемость алгоритма.

СПИСОК ЛИТЕРАТУРЫ

1. Muhlenbein H. Schlierkamp-Voosen D. Predictive models for the breeder genetic algorithm // Evolutionary Computation. — 1993. — 1, № 1. — P. 25-50.
2. Grefensette J. J., Baker J.E. How genetic algorithms work: A critical look at implicit parallelism // Proceedings of the Third International Conference on Genetic Algorithms. — San Mateo: Morgan Kaufmann, 1989. — P. 20-27.
3. Eiben A.E., Raue P-E., Ruttkay Z. Genetic algorithms with multi-parent recombination // Proceedings of the 3rd Conference on Parallel Problem Solving from Nature. — Berlin: Springer-Verlag, 1994. — P. 78-87.
4. Syswerda G. Uniform crossover in genetic algorithms // Proceedings of the Third International Conference on Genetic Algorithms. — San Mateo: Morgan Kaufmann, 1989. — P. 2-9.
5. Radcliffe N. J. Equivalence class analysis of genetic algorithms // Complex Systems. — 1990. — 5, № 2. — P. 183-205.
6. Летичевский А.А., Капитонова Ю.В. Доказательство теорем в математической информационной среде // Кибернетика и системный анализ. — 1998. — № 4. — С. 3-12.
7. Holland John H. Adaptation in natural and artificial systems. Ann Arbor The University of Michigan Press. — 1975. — 97 p.
8. Poli R. Exact schema- theory for genetic programming and variable-length genetic algorithms with one-point crossover // Genetic Programming and Evolvable Machines. — Las Vegas: Morgan Kaufmann. — 2001. — P. 469-476.
9. Blickle T., Thiele L. A mathematical analysis of tournament selection // Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA'95). — San Mateo: Morgan Kaufmann, 1995. — P. 9-16.
10. Whitley D. The GENITOR algorithm and selection pressure / In J.D. Schaffer, editor // Proceedings of the Third International Conference on Genetic Algorithms. — San Mateo: Morgan Kaufmann, 1989. — P. 116-121.
11. Hesser J., Manner R. Towards an optimal mutation probability for genetic algorithms// Parallel Problem Solving from Nature. — 1991. — 11. — P. 96-119.
12. Strongly typed genetic programming in evolving cooperation strategies / T. Hayncs, R. Wainwright, S. Sen, D. Schoenefeld // Proceedings of the 6th International Conference on Genetic Algorithms. — San Mateo: Morgan Kaufmann. — 1995. — P. 271-278.
13. Jansen T., Wegener I. On the choice of the mutation probability for the (1 + 1) EA // Parallel Problem Solving from Nature. — 2000. — 6. — P. 233-239.

14. Fogarty T.C. Varying the probability of mutation in the genetic algorithm // Proceedings of the Third International Conference on Genetic Algorithms. — La Jolla, CA: Morgan Kaufmann, 1989. — P. 104-109.
15. Poli R., Langdon W.B. Genetic programming with one-point crossover // Second on-line World Conference on Soft Computing in Engineering Design and Manufacturing. — London: Springer-Verlag, 1997. — 77 p.
16. Blickle T., Thiele L. Genetic programming and redundancy // Genetic Algorithms within the Framework of Evolutionary Computation. — Saarbrücken: Max-Plank-Institut fur Informatik. — 1994. — P. 33-38.
17. Evan S., Itai A., Shamir A. On the complexity of timetable and multicom-modify flow problems // SIAM J. of Comp. — 1976. — 5, № 4. — P. 691-703.
18. Erben W., Keppler K. A general algorithm solving a weekly course timetabling problem. — London: Springer-Verlag, Lecture Notes in Computer Science. — 1999. — 1153. — P. 198-211.

Поступила 11.02.2002

УДК 519.716.35

Л.П. ЛИСОВИК, Т.А. КАРНАУХ

О КЛАССЕ ФУНКЦИЙ, ВЫЧИСЛИМЫХ С ПОМОЩЬЮ ИНДЕКСНЫХ ГРАММАТИК

Ключевые слова: *индексная грамматика, вычислимость, линейное множество, полулинейное множество, линейные рекуррентные соотношения, свойства замкнутости.*

Индексные грамматики впервые были рассмотрены А. Ахо [1]. Порождаемые ими индексные языки представляют собой класс языков, занимающий промежуточное положение между классами контекстно-свободных и контекстно-зависимых языков в иерархии Хомского. Известно [2], что класс индексных языков совпадает с классом языков, распознаваемых гнездовыми стековыми автоматами. Позднее в работах [3, 4] было показано, что любой индексный язык распознается детерминированным линейно-ограниченным автоматом.

В данной работе продолжается исследование индексных грамматик в ракурсе их вычислительных возможностей. Начиная с аналога теоремы Парика для индексных языков мы переходим к рассмотрению классов функций *IGF* и JGF_0— функций вычислимых и абсолютно вычислимых с помощью индексных грамматик. Показано, что класс IGF_0 совпадает с классом строго возрастающих функций из N_+ в N_+, задаваемых системами линейных рекуррентных соотношений с натуральными коэффициентами, и замкнут относительно операций сложения, суммирования, умножения, прибавления натуральной константы к аргументу или к результату.

Напомним кратко определение индексной грамматики. Под индексной грамматикой подразумевается пятерка $G = (V, \Sigma, F, P, \sigma)$, где V — конечное множество, Σ — его подмножество, σ ($\sigma \in \Delta$) — начальный символ, $\Delta = V \setminus \Sigma$ — множество нетерминалов, F — конечное множество индексов (указателей), P — конечная система продукций вида $A \to \xi$, где $A \in \Delta$,