Генетические алгоритмы в задаче поиска часто встречающихся комбинаций[1]

Н. Н. Кузюрин , С. А. Мартишин , М. В. Храпченко

Аннотация. Рассматривается задача поиска часто встречающихся комбинаций, связанная с анализом данных (data mining). Рассмотрены некоторые теоретические аспекты, связанные с алгоритмической сложностью задачи и существованием эффективных приближенных алгоритмов. Предложен генетический алгоритм для решения этой задачи и проведено исследование его эффективности на случайных данных.

1. Введение

Термин Data Mining, который часто переводится как «добыча» или «раскопка данных» характеризует систему поиска закономерностей в данных и, возможно, прогнозирования тенденций их проявления. Чаще всего используется определение, данное одним из основоположников этого направления Г. Пятецким-Шапиро: «Data mining — это процесс обнаружения в сырых данных ранее неизвестных нетривиальных практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности».

Подобные задачи стали актуальными в связи с огромным количеством информации самого разного вида, накопленной в базах данных. Часто можно встретить и синоним Data Mining — «обнаружение знаний в базах данных» (knowledge discovery in databases), акцентирующий внимание на необходимости не только извлечения, но и анализа информации, содержащейся в базах данных. Поскольку процесс извлечения и анализа информации является достаточно трудоемким, для такой работы требуется использование больших вычислительных мощностей.

В принципе, сфера применения Data Mining не ограничена. Это анализ покупательской корзины и построение прогнозирующих моделей потребления, страхование и банковское дело (разработка тарифных планов и выявление мошенничества), медицина (для выявления закономерностей при постановке диагноза и назначении лечения), демография и многие другие.

Одним из классических примеров применения Data Mining традиционно считается задача анализа рыночной корзины (market basket analysis). Рыночная корзина — это набор товаров, приобретенных покупателем в рамках

[1]Работа выполнена при поддержке РФФИ, проект 02-01-00713.

одной отдельной транзакции. Любой традиционный поход в магазин (будь то супермаркет, ближайшая булочная или Internet-магазин) имеет практически одинаковые характеристики. Торговые компании (особенно достаточно крупные) регистрируют все свои операции. Во многих компаниях эти данные накапливаются за достаточно длительный период их деятельности. Естественно, что одной из самых распространенных задач, решаемых при проведении анализа подобных баз данных, состоит в поиске товаров или наборов товаров (itemset), которые одновременно встречаются во многих транзакциях. Каждый из шаблонов (pattern) поведения покупателей, выявленных благодаря такому анализу, описывается перечнем входящих в набор товаров и числа транзакций, содержащих эти наборы. Торговые компании могут использовать эти шаблоны для того, чтобы более правильно разместить товары в магазинах и более адекватно прогнозировать спрос.

Набор, состоящий из i товаров, называется i-элементным набором (i-itemset). Принято считать, что для того чтобы набор представлял интерес, количество транзакций, содержащих данный набор, должно быть выше определенного пользователем минимума. Такие наборы называют часто встречающимися (frequent).

При этом поиск часто встречающихся наборов представляет собой достаточно сложную задачу. Это связано с тем, что базы данных, отражающие транзакции, очень велики и, очевидно, не могут быть размещены в оперативной памяти. Кроме того, потенциальное число часто встречающихся наборов быстро растет с ростом количества товаров в наборе, хотя реальное число часто встречающихся наборов может оказаться намного меньше.

В следующем разделе мы даем точную формулировку рассматриваемой задачи и рассматриваем ее связи с другими комбинаторными задачами.

2. Задача поиска часто встречающихся комбинаций и комбинаторика

Рассмотрим класс задач, связанных с анализом информации о рыночной корзине (market basket analysis), и дадим их комбинаторную формулировку. Рыночная корзина может быть описана матрицей, строки которой соответствуют покупкам, а столбцы — товарам.

Для $(0,1)$-матрицы $A = (a_{ij})$ размера $m \times n$ каждому подмножеству S столбцов сопоставляется подмножество $R(S)$ всех строк r, для которых $a_{rc} = 1$ при всех $c \in S$. Для заданного натурального k задача заключается в нахождении максимума $|R(S)|$ по всем подмножествам S, состоящим из k столбцов, т.е. $F(A,k) = \max_{S \subseteq C:\ |S|=k} |R(S)|$.

Самые часто встречающимися k-множествами S называются те, для которых $|R(S)| = F(A,k)$, а t-частыми называются те, для которых выполнено неравенство $|R(S)| \geq t$.

Проблема А. Для заданной $(0,1)$-матрицы A и натурального k найти все самые часто встречающиеся k-множества.

Нахождение самых часто встречающихся множеств для данной матрицы A является важной задачей анализа данных (knowledge discovery и data mining) [13, 9, 10]. В частности, в упомянутом выше приложении (market basket), столбцы матрицы могут соответствовать типам товаров, а строки — покупкам. Тогда самые часто встречающиеся множества указывают наиболее покупаемые комбинации товаров.

Рассмотрим связь этой задачи с задачами комбинаторной оптимизации. Для этого сформулируем сходную задачу поиска одного самого часто встречающегося множества и рассмотрим несколько других комбинаторных проблем.

Проблема 1. Для заданной $(0,1)$-матрицы A и натурального k найти самое часто встречающееся k-множество (хотя бы одно).

Проблема 2. Для данного семейства $F = \{S_j : j \in J\}$ подмножеств множества $I = \{1, 2, \ldots, m\}$ и натурального k, найти подмножество $X \subseteq J$ с $|X| = k$ максимизирующее число элементов в I, не входящих в $\cup_{j\in X} S_j$ (непокрытых $\cup_{j\in X} S_j$).

Проблема 3. Для заданного двудольного графа $G = (V_1, V_2, E)$ и натурального k найти в V_1 k-элементное подмножество, имеющее минимальную тень в V_2. Как обычно, под тенью множества S понимается множество вершин графа, смежных с одной из вершин из S.

Проблема 4. Для заданного двудольного графа $G = (V_1, V_2, E)$ и натурального k найти максимальную двудольную клику, имеющую в V_1 ровно k вершин. Напомним, что двудольной кликой называется полный двудольный подграф исходного графа.

Утверждение 1. Проблемы 1-4 полиномиально сводятся друг к другу.

$1 \Rightarrow 4$. Поставим матрице A в соответствие двудольный граф G такой, что V_1 соответствует столбцам, V_2 соответствует строкам и имеется ребро (i, j) тогда и только тогда, когда $a_{ji} = 1$. Тогда подматрица с k столбцами, состоящая только из единиц, соответствует двудольной клике в графе G.

$4 \Rightarrow 3$. Для данного двудольного графа G рассмотрим дополнительный двудольный граф $\overline{G}$ такой, что в нем имеется ребро (i, j), (где $i \in V_1$, $j \in V_2$) тогда и только тогда, когда в G отсутствует ребро (i, j). Тогда двудольной клике (U, V) в графе G соответствует пара $(U, V_2 \setminus V)$ такая, что в графе $\overline{G}$ множество $V_2 \setminus V$ является тенью множества U. Таким образом, максимизация клики соответствует минимизации тени.

$3 \Rightarrow 2$. Поставим семейство $F = \{S_j : j \in J\}$ подмножеств множества $I = \{1, 2, \ldots, m\}$ в соответствие двудольному графу G так, что V_1 соответствует J, V_2 соответствует $I = \{1, 2, \ldots, m\}$ и S_j состоит из всех i, для которых имеется ребро (i, j) в графе G. Тогда подмножество из k элементов, максимизирующее число непокрытых элементов, соответствует двудольной

клике в двудольном дополнении к графу G.

$2 \Rightarrow 1$. Сопоставим семейству подмножеств $(0,1)$-матрицу такую, что ее столбцы соответствуют подмножествам семейства F, строки — элементам базового множества I, элемент $a_{ij} = 1$ тогда и только тогда, когда элемент i не принадлежит j-му подмножеству семейства. Тогда минимизация числа покрытых k-множеством элементов соответствует максимизации числа непокрытых и, тем самым, максимизации числа вхождений строк, целиком состоящих из единиц.

Вместо оптимизационных задач 1-4 можно рассматривать аналогичные им задачи разрешения. Например, для проблемы 1 таким аналогом является

Проблема 1′. Для заданной $(0,1)$-матрицы A и натуральных k и t существует ли t-частое k-множество.

Утверждение 2. Задачи разрешения, соответствующие проблемам 1–4, являются NP-полными.

Это следует, например, из [7], где доказано, что таковой является задача проверки, является ли данный двудольный граф расширителем. В частности, в упомянутой статье показано, что задача проверки существования в двудольном графе множества вершин S, состоящего из не более, чем половины вершин первой доли и имеющего тень менее $|S|$, является NP-полной. Полагая в Проблеме 3 $k \leq |V_1|/2$, $t = k$ получаем требуемую сводимость.

Упомянем теперь некоторые известные алгоритмы для решения задачи А. Одним из известных алгоритмов решения этой задачи является алгоритм Apriori [8]. Этот алгоритм находит часто встречающиеся наборы за несколько этапов. На i-ом этапе определяются все часто встречающиеся i-элементные наборы. Каждый этап состоит из двух шагов: формирование кандидатов (candidate generation) и подсчет кандидатов (candidate counting).

В работе [10] был предложен алгоритм динамического подсчета самых частых наборов Dynamic Itemset Counting (DIC). Алгоритм основан на построении структуры данных в виде дерева, корневыми вершинами которого являются все 1-элементные наборы.

В [9] проанализированы сложностные аспекты нахождения максимальных по включению так называемых t-частых и t-редких подмножеств и для первой задачи доказана ее NP-трудность, а для второй — предложен квазиполиномиальный алгоритм. Другие алгоритмы нахождения часто встречающихся наборов можно найти в [11, 12].

Все эти алгоритмы имеют экспоненциальную в худшем случае трудоемкость. Интерес, как обычно, представляет построение приближенных алгоритмов с гарантированным качеством аппроксимации. С теоретической точки зрения эти вопросы исследованы явно недостаточно, хотя в последнее время получен ряд интересных результатов. Наилучшая мультиплика-

тивная точность приближения, которую удается эффективно получить для задачи о двудольной клике в графе с n вершинами в каждой доле — это $O\left(n/(\log n)^c\right)$ для некоторой константы c. Поскольку тривиально достижима точность n, то продвижение не слишком велико. Более того, имеются результаты, свидетельствующие о трудности аппроксимации максимальной двудольной клики с точностью n^δ [14].

Нам представляется интересным изучение возможностей применения генетических алгоритмов для этой задачи. Вообще говоря, Data mining не является основной сферой применения генетических алгоритмов. С другой стороны, поскольку генетически алгоритмы достаточно хорошо решают задачи комбинаторной оптимизации, применение их в задачах типа анализа информации о рыночной корзине может оказаться полезным.

3. Кратко о генетических алгоритмах

Пусть дана некоторая целевая функция, в общем случае зависящая от нескольких переменных, и требуется найти такие значения переменных, при которых значение функции максимально. Генетический алгоритм — это простая модель эволюции в природе, реализованная в виде алгоритма. В нем используются как аналог механизма генетического наследования, так и аналог естественного отбора. При этом используется биологическая терминология. Мы имеем дело с особью (индивидуумом). Особь — это некоторое решение задачи. Будем считать особь тем более приспособленной, чем лучше соответствующее решение (чем больше значение целевой функции это решение дает). Тогда задача максимизации целевой функции сводится к поиску более приспособленной особи. То есть выбирая наиболее приспособленную особь в текущем поколении, можно получить не абсолютно точный, но близкий к оптимальному ответ. Особи наделяются хромосомами. Для моделирования наследования в генетических алгоритмах используются следующие термины и понятия:

Хромосома — вектор (последовательность), содержащий набор значений. Каждая позиция в хромосоме называется **геном** .

Набор хромосом — вариант решения задачи.

Целевая функция оценки приспособленности (fitness) решений.

Основные операторы генетического алгоритма:

Отбор — селекция (reproduction, selection) осуществляет отбор хромосом в соответствии со значениями их функции приспособленности.

Скрещивание (crossover) — операция, при которой две хромосомы обмениваются своими частями.

Мутация — случайное изменение одной или нескольких позиций в хромосоме.

Генетический алгоритм состоит из последовательности шагов.

Шаг 0. *Формирование случайной популяции.*

Далее итеративно повторяются шаги 1-2.

Шаг 1. *Селекция и скрещивание.*

Шаг 2. *Мутация.*

Процесс работы генетического алгоритма продолжается до выполнения некоторого критерия останова (как правило, количества итераций). В каждом следующем поколении возникнут новые решения задачи. Среди них будут как плохие, так и хорошие, но благодаря отбору, число приемлемых решений будет возрастать.

Поясним подробнее функционирование основных операторов.

Отбор — это процесс формирования новой популяции из старой на основании определенных правил, после чего к новой популяции опять применяются операции кроссовера и мутации, затем опять происходит отбор, и так далее. В традиционном генетическом алгоритме реализуется отбор пропорционально приспособленности (так называемая селекция), одноточечный кроссовер и мутация. Вообще говоря, после того как выбраны основные операторы генетического алгоритма (отбор, кроссовер и мутация), сам алгоритм определяется однозначно. При этом существуют различные методы выбора операторов, которые могут дать различные результаты реализации. Для оператора отбора (селекции) наиболее распространенными являются два метода — метод рулетки (roulette-wheel selection) и турнирного отбора (tournament selection). Метод рулетки заключается в том, что особи отбираются с помощью n «запусков» рулетки. В этом случае колесо рулетки содержит по одному сектору для каждого члена популяции. Размер i-ого сектора пропорционален соответствующей величине $P(i)$ вычисляемой по формуле:

$$P(i) = \frac{f(i)}{\sum_{i=1}^{n} f(i)}.$$

При таком отборе члены популяции с более высокой приспособленностью с большей вероятностью будут чаще выбираться, чем особи с низкой приспособленностью. Второй метод — турнирного отбора — реализует n турниров, чтобы выбрать n особей. Каждый турнир построен на выборке m элементов из популяции, и выбора лучшей особи среди них. Наиболее распространен турнирный отбор с $m = 2$. При этом решение, какие именно особи переходят в следующую популяцию, также может быть различным. Например, мы можем всегда оставлять в популяции потомков, даже если оценка их приспособленности хуже, чем у родителей. Мы можем сравнивать потомков с родителями и оставлять в популяции лучших. Элитные методы отбора гарантируют, что при отборе обязательно будут выживать лучшие члены популяции. Наиболее распространена процедура обязательного сохранения только одной лучшей особи, если она не прошла как другие через процесс отбора, кроссовера и мутации. Однако здесь следует иметь в виду, что критерий отбора хромосом не дает нам гарантию нахождения

наилучшего решения, поскольку эволюция может пойти по пути неоптимального отбора или из популяции будут исключены неперспективные родители, потомки которых могут оказаться эффективными.

После отбора n избранных особей случайным образом разбиваются на $n/2$ пар. Для каждой пары с вероятностью p может применяться скрещивание. Соответственно, с вероятностью $1-p$ скрещивание не происходит и неизмененные особи переходят на стадию мутации.

Скрещивание (crossover) — осуществляет обмен частями хромосом, как правило, между двумя хромосомами в популяции. Может быть одноточечным или многоточечным. Одноточечное скрещивание работает следующим образом. Сначала, случайным образом выбирается одна из $l-1$ точек разрыва. Точка разрыва — участок между соседними генами. Обе родительские структуры разрываются на два сегмента по этой точке. Затем, соответствующие сегменты различных родителей склеиваются и получаются два генотипа потомков.

Также применяются двухточечное скрещивание, при котором выбираются две точки разрыва, и родительские хромосомы обмениваются сегментом, находящимся между этими точками, и равномерное скрещивание, в котором каждый бит первого родителя наследуется первым потомком с заданной вероятностью, в противном случае этот бит передается второму потомку и наоборот. Достаточно часто хорошо работает метод, при котором одинаковый участок генов для хромосом остается неизменным (и является некоторым шаблоном), и осуществляется обмен генов, не входящих в данный участок.

Мутация — стохастическое изменение части хромосом. В хромосоме, которая подвергается мутации, каждый ген с вероятностью P_{mut} (обычно очень маленькой) меняется на некоторый другой из заданного диапазона значений.

4. Генетический алгоритм для задачи поиска часто встречающихся комбинаций

Опишем применение общей схемы генетического алгоритма к задаче поиска наиболее часто встречающихся сочетаний товаров в покупках. Задачу можно рассматривать как нахождение самых частых множеств для $m \times n$ (0,1)-матрицы $A = (a_{ij})$. При этом столбцы матрицы соответствуют наименованию товаров, а строки — конкретной покупке. Если конкретный товар был выбран при совершении данной покупки, то в соответствующей строке на месте ее пересечения с данным столбцом ставится единица.

Очевидно, что исходная матрица сильно разрежена. Поэтому будем предполагать, что данные некоторым образом отсортированы. То есть на вход генетического алгоритма подается матрица, упакованная в некоторый исходный файл вида:


```
<число товаров в ассортименте>\\
<число покупателей>\\
<число товаров у 1-го покупателя>\\
<номер 1-го товара 1-го покупателя><номер 2-го товара 1-го покупателя>....\\
<число товаров у 2-го покупателя>\\
<номер 1-го товара 2-го покупателя><номер 2-го товара 2-го покупателя>....
```

Таким образом, номера столбцов конкретного покупателя, в которых содержатся единицы в исходной матрице заменяются их номерами. Номера упорядочены по возрастанию. Это более естественный способ хранения данных и с точки зрения возможности получения сведений о реальных покупках, поскольку содержит информацию из какого количества каких наименований товаров состоит конкретная покупка. Таким образом, мы работаем с модифицированной матрицей.

Генетический алгоритм заключается в последовательном выполнении следующих шагов:

Шаг 0. Формирование популяции. Выбираем некоторое число *Npop* особей в популяции (их конкретное число рассчитывается как процент от числа покупок и может быть изменен в файле инициализации). Из строк, содержащих номера товаров случайным образом формируем выбираем *Npop* строк, которые будут основой популяции. Все строки различны. Далее случайным образом из каждой строки формируем особей популяции. Для этого случайным образом выбираем k различных генов (номеров товаров). Если этого сделать нельзя (строка содержит меньшее число товаров), то строка отбрасывается и заменяется другой. Таким образом мы получаем популяцию хромосом, состоящих из генов размерности k. Следующий шаг состоит в том, что каждой хромосоме в последний $(k+1)$-й элемент записывается значение фитнесс-функции. Она вычисляется как число вхождений данного сочетания товаров в исходную матрицу.

Теперь сформированы необходимые исходные данные для выполнения основных операторов генетического алгоритма. В качестве оператора селекции выбирается турнирный отбор двух особей, поскольку для данной задачи, регулируя число турнирных отборов, можно получить дополнительные сведения не только о самом частом сочетании товаров, но и о достаточно частых сочетаниях. Выбрано одноточечное скрещивание в случайно выбранной точке. При этом, если у двух хромосом некоторые гены совпадают, то обмен осуществляется только различными генами, а совпадающие гены в скрещивании не участвуют.

После формирования популяции мы переходим к итеративному выполнению операций генетического алгоритма (селекции, скрещивания и мутации) для ее модификации.

Шаг 1. Селекция и скрещивание. Случайным образом выбираем две особи. Лучшая (по значению фитнесс-функции — обеспечению набора)

остается. Снова случайным образом выбираем две особи. Лучшая из них (также по значению фитнесс-функции) также остается. Две оставшиеся особи составляют пару для скрещивания. Они подвергаются одноточечному скрещиванию в случайно выбранной точке. Сравниваем получившихся потомков (сопоставляя по количеству вхождений) с родителями, лучших из получившихся особей оставляем в популяции. Реализован также другой вариант алгоритма, при котором лучшие особи-дети заменяют родителей в популяции при условии, что они не совпадают ни с одной из особей, уже присутствующих в популяции.

Шаг 2. Мутация. С очень маленькой вероятностью (для нашей задачи было выбрано значение 0.01) изменяем ген некоторых случайным образом выбранных хромосом на некоторое случайным образом выбранное число из диапазона количества товаров, при этом следим, чтобы в хромосоме не появились одинаковые гены. То есть каждый член популяции с вероятностью 0.01 подвергается мутации. Если условие выполнено и процесс мутации запущен, то произвольным образом выбирается ген, который подвергается мутации. Формирование новой популяции закончено. Итерационный процесс продолжается определенное число итераций, число которых также может быть задано в файле инициализации.

Ниже следует описание реализации генетического алгоритма, состоящего из следующих этапов:

1. Формирование популяции: Выбираем случайным образом строку матрицы (члена популяции), получив строку матрицы мы должны проверить:

1. Не короче ли она длины выборки;
2. Не встречалась ли она ранее (проверка производится с использованием битового массива, в который заносится единица, если строка уже была отобрана для популяции).

Таким образом отбирается необходимое число строк матрицы, равное числу членов популяции. Далее делаем выборку длины k для каждой отобранной строки, то есть случайным образом из числа товаров конкретного покупателя выбираются k штук. При этом необходимо проверить, не встречался ли данный товар в выборке ранее. Проверка производится с использованием битового массива, в который заносится единица, если строка уже была отобрана для популяции).

Популяция сформирована.

Для дальнейшей работы алгоритма необходимо определить обеспечение каждого набора, входящего в популяцию, иными словами вычислить его фитнесс-функцию. Определение фитнесс-функции каждой особи в популяции (числа вхождений в исходную матрицу), который будет хранится в последнем элементе строки выполняется при помощи функции вычисления обеспечения.

2. Итерации генетического алгоритма:

```
for i=1 to i=ЧислоПоколений do
begin
.   for j=1 to j=ЧислоТурнирныхОтборов do
.   begin
.     Турнирный отбор
.   end
.   Мутация
end
```

Процедура **Турнирный отбор** реализована для m=2 и состоит из следующих шагов:

1. Случайным образом выбираются две особи популяции и сравниваются по значению фитнесс-функции (числу вхождений). Оставляем лучшую.
2. Аналогичным образом выбираем еще две особи, следя за тем, чтобы они не совпали с ранее выбранными.Сравниваем их таким же способом и оставляем лучшую. Теперь у нас есть две особи для обмена генами в хромосомах.
3. Полученную пару подвергаем скрещиванию. Случайным образом выбираем количество генов, которыми особи будут обмениваться (не менее 1 гена и не более $k-1$). В том случае, если у векторов популяции есть совпадающие гены, то они остаются на месте, а обмен производится теми компонентами, которые не совпадают. В результате получаем две новых особи популяции. Подсчитываем для них значение фитнесс-функции (числа вхождений) и сравниваем со значением родительских значений.

Процедура **Мутация** состоит из следующих шагов:

1. Для каждой строки проводится испытание, чтобы определить, подвергается ли она мутации.
2. Если нет, то переходим на шаг 1, если да, то переходим на шаг 3.
3. Случайным образом выбираем ген, который подвергнется мутации и выбираем произвольное число из допустимого диапазона, в данном случае это число различных товаров (столбцов матрицы). Проверяем, что такого элемента у нас не было в выборке, заменяем выбранный ген на полученный произвольный номер столбца.

Алгоритм реализован на C++. Некоторые результаты вычислительных экспериментов приводятся в следующем разделе.

5. Результаты реализации

Для отладки и интерпретации результатов были рассмотрены два типа случайных матриц. Первая представляет собой модель, в которой вероятность выбора каждого элемента одинакова и описывается биномиальным распределением. Второй тип основывается на более сложной вероятностной модели, описанной ниже. Она основана на реальных данных маркетинговых агентств, анализирующих группы покупателей по частоте их посещений супермаркетов и соответствующих количествах товаров в их транзакциях.

Будем предполагать, что в магазине есть в наличии N=4000 наименований товаров. Рассмотрим покупки 10000 случайным образом выбранных покупателей. Каждая покупка может быть представлена как набор из N чисел $(a_1, \ldots, a_N)$, где каждый коэффициент a_i $(i = 1, \ldots, N)$, равен либо 1, либо 0 в зависимости от того, куплен ли товар i-го наименования или нет (различные номера i соответствуют различным наименованиям товаров). Пусть n обозначает количество различных наименований товаров в произвольной покупке. Тогда $n = \Sigma_{i=1}^{N} a_i$.

Первая вероятностная модель 1. Пусть для каждого покупателя вероятность выбора любого конкретного наименования товара равна p. Элементарным событием в данной модели является набор из N чисел $(a_1, \ldots, a_N)$, описывающий покупку. Пространством элементарных событий является множество всех таких наборов. Согласно нашим предположениям, для каждого покупателя вероятность конкретной покупки есть

$$P\{a_i, \ldots, a_N\} = (\delta_i, \ldots, \delta_N) = p^m q^{N-m},$$

где m обозначает количество единиц в наборе $(\delta_i, \ldots, \delta_N)$, то есть количество наименований товаров в покупке.

Заметим, что при наших предположениях, для каждого покупателя количество наименований товаров в покупке имеет биномиальное распределение: пусть K обозначает количество наименований товаров в покупке конкретного покупателя (K — случайная величина), тогда

$$P\{K = m\} = C_N^m p^m q^{N-m}.$$

При реализации данного алгоритма с p=0.05 (соответственно, q=0.95) генерируется разреженная матрица размера 10000×4000 (если строки соответствуют покупателям, а столбцы — наименованиям товаров), у которой среднее количество единиц по строкам равно $N_p = 4000 \times 0.05 = 20$.

Вторая вероятностная модель. Так же, как и раньше, предполагаем, что 10000 покупателей делают покупки в магазине, выбирая из N=4000 наименований товаров. В отличие от предыдущего случая, предполагаем, что множество покупателей неоднородно, поэтому покупки могут сильно

различаться по количеству выбранных наименований товаров, так и по их ассортименту. В связи с этим (и для большей наглядности модели) мы разбиваем все множество покупателей на четыре группы следующим образом:

а) Группа M1: ежедневное или разовое посещение магазина для приобретения товаров первой необходимости.

б) Группа M2: посещение магазина раз в 2–3 дня.

в) Группа M3: посещение магазина раз в неделю.

г) Группа M4: разовое посещение магазина в связи с праздником, переездом, ремонтом и т.п.

Опишем вторую вероятностное пространство, которое будем использовать для моделирования процесса покупки. Элементарным событием является набор из $N + 1$ чисел $(g, a_1, \ldots, a_N)$. В этом наборе первый элемент g является номером группы, к которой принадлежит покупатель (g может принимать значения из множества {1,2,3,4}), а остальные числа этого набора определяют саму покупку, как это описано выше. Вероятность на множестве элементарных событий зададим следующим образом:

1. Вероятность того, что выбранный случайным образом покупатель принадлежит к конкретной группе, задается таблицей:

$P(M1) = P\{g = 1\}$	$P(M2) = P\{g = 2\}$	$P(M3) = P\{g = 3\}$	$P(M4) = P\{g = 4\}$
0.47	0.4	0.1	0.03

2. Для каждой группы покупателей задаем условные вероятности выбора определенных наименований товаров:

а) Группа M1: каждое из 7 первых наименований товаров выбирается с вероятностью 0.7, следующие 10 наименований — с вероятностью 0.15, остальные наименования — с вероятностью 0. В среднем покупка содержит 6-7 наименований товаров.

Замечание. Первые 7 наименований товаров — это предметы первой необходимости, следующие 10 наименований — товары «ежедневного спроса».

б) Группа M2: каждое из 7 наименований товаров (тех же, что и в предыдущем случае) выбираются с вероятностью 0.7, следующие 10 наименований (те же, что и в предыдущем случае) — с вероятностью 0.25, следующие 20 наименований — с вероятностью 0.1, остальные наименования — с вероятностью 0. В среднем покупка содержит 9-10 наименований товаров.

в) Группа M3: каждое из 7 наименований товаров (тех же, что и в предыдущем случае) выбираются с вероятностью 0.7, следующие 10 наименований (те же, что предыдущем случае) — с вероятностью 0.25, следующие 20 наименований (те же, что предыдущем случае) — с вероятностью 0.15, следующие 20 наименований — с вероятностью 0.15, остальные наименования выбираются с вероятностью 0.005. В среднем покупка состоит из 33 наименований товаров.

г) Группа М4: каждое из 7 наименований товаров (тех же, что и в предыдущем случае) выбираются с вероятностью 0.7, следующие 10 наименований (те же, что предыдущем случае) — с вероятностью 0.25, следующие 20 наименований (те же, что предыдущем случае) — с вероятностью 0.15, следующие 20 наименований — с вероятностью 0.15, следующие 100 наименований — с вероятностью 0.08, остальные наименования — с вероятностью 0.02. В среднем покупка состоит из 98 наименований товаров.

При этом мы предполагаем, что покупатель (из определенной группы) выбирает наименования товаров независимо.

До начала генерации матрицы необходимо сгенерировать 4 типа векторов, соответствующих различным типам покупательской активности. Для каждой строки матрицы проводится испытание — запускается генератор псевдослучайных чисел. В соответствии с полученным значением выбирается вектор и заполняются все элементы строки, вероятности для каждого из которых находятся в соответствующем векторе. Аналогично проводится испытание для каждого элемента строки и, в зависимости от значения генератора псевдослучайных чисел товар выбирается или нет. При этом исходные данные легко преобразовать к требуемому виду.

Помимо исходных данных по покупкам для работы генетического алгоритма необходимы следующие данные: размер популяции (он вычисляется как процент от числа покупок), число k — размер множества, вхождения которого мы ищем, количество итераций (смены поколений) и число турнирных отборов. Если задача заключается в нахождении не только одного решения (наибольшего вхождения), но и некоторого следующих по частоте вхождений, то вводится параметр количества следующих по частоте вхождений $Ndiv$.

Рассмотрим сначала результаты для второй модели. Эксперименты проводились для множеств размера k= 3, 5, 6 и 7.

Остальные параметры (размер популяции, число итераций и турнирных отборов) можно модифицировать для решения других задач. Так, например, размер популяции (для матрицы, сгенерированной первым способом) достаточный для получения точного решения равен 1% от числа покупателей. При некоторых исходных данных это число может быть еще меньше.

Для получения единственного решения число турнирных отборов и смен поколений необходимо увеличить. Экспериментальные данные показывают, что, например, уменьшив размер популяции до 0.5%, необходимо увеличить в 10 раз число турнирных отборов. При этом следует учитывать, что рост числа турнирных отборов увеличивает время выполнения алгоритма прямо пропорционально (см. таблицу ниже). Кроме того, ясно, что существует предельно малое значение популяции, при котором решение задачи найдено не будет. Это зависит от конкретных исходных данных и для нашего способа генерации матрицы близко к 0.1% . также совершенно оче-

Размер популяции 1%				
Число итераций/ число отборов	k=3	k=5	k=6	k=7
10/10	2c	2c	2c	2c
10/100	24c	24c	24c	25c
100/10	24c	24c	24c	25c
100/100	245c	245c	245c	247c
Размер популяции 10%				
Число итераций/ число отборов	k=3	k=5	k=6	k=7
10/10	2c	2c	2c	2c
10/100	24c	24c	24c	25c
100/10	24c	24c	24c	25c
100/100	240c	245c	245c	247c

Таблица 1: Производительность алгоритма для второй модели.

видно, что малый (1%) размер популяции не позволит получить сведения о следующих по частоте вхождениях. Для этого размер популяции следует сделать значительным — не менее репрезентативной выборки от числа покупок (репрезентативная выборка для данной задачи составляет примерно 10% от числа членов популяции). Количество турнирных отборов и итераций следует значительно уменьшить: число итераций от 0.1% до 1% от числа членов популяции, а число турнирных отборов — можно сделать равным числу итераций или немного больше.

Увеличение требуемой памяти в при увеличения числа членов популяции также не является критическим для выполнения алгоритма на современных ПК, поскольку даже при размере множества вхождения 10 и размере популяции 100 000 нам требуется для хранения 1 000 000 целых чисел, которые (как тип integer) требуют для хранения 4 байта. То есть всего нам необходимо около 4Мб (в реальных задачах даже меньше, поскольку для больших матриц репрезентативная выборка меньше 10% от числа покупок и число k меньше 10). То есть объем требуемой памяти не столь велик.

Все вычислительные эксперименты проводились на ПК Athlon 1600XP и время выполнения проверено на достаточно большом числе матриц и отличается для различных случаев не более чем на 5%.

Из таблиц (см. Таб.1) видно, что производительность зависит в большой степени от числа итераций и количества турнирных отборов и практически не зависит от размера популяции.

Для большей наглядности приведем еще одну таблицу (см. Таб.2), показывающую, что длительность прямо пропорциональна числу итераций и турнирных отборов.

Наиболее важным является вопрос, насколько хорошо алгоритм ищет решение задачи. Для этого был проведен сравнительный анализ результатов данного алгоритма с алгоритмом полного перебора. Осуществлялся поиск 15 лучших вхождений. Следует заметить, что время выполнения ал-

Размер популяции 1%		
Число итераций	Число отборов	k=5
100	10	24c
100	50	120c
1000	10	249
1000	20	507c

Таблица 2: Время выполнения в зависимости от числа итераций.

Генетический алгоритм		Алгоритм полного перебора	
Число вхождений	Номера столбцов	Число вхождений	Номера столбцов
3534	2501 3725 2468	3534	2468 2501 3725
3504	3479 3725 2468	3504	2468 3479 3725
3495	2468 3479 2501	3495	2468 2501 3479
3494	3170 2468 2501	3494	2468 2501 3170
3482	3479 2501 3725	3482	2501 3479 3725
3480	2468 3479 42	3480	42 2468 3479
3474	2468 3725 42	3474	42 2468 3725
3465	2501 3170 3479	3465	2501 3170 3479
3463	42 2468 2501	3463	42 2468 2501
3463	3479 2501 42	3463	42 2501 3479
3459	3479 2468 3170	3459	2335 2468 3479
3459	3479 2468 2335	3459	2468 3170 3479
3447	3170 3479 42	3447	42 3170 3479
3447	3170 2501 3725	3447	2335 2468 2501
3447	2335 2468 2501	3447	2501 3170 3725
3445	3170 3725 2468	3445	2468 3170 3725
3439	42 2501 3725	3439	42 2501 3725
3437	2468 42 3170	3437	42 2468 3170
3435	2468 3170 2335	3435	2335 2468 3170

Таблица 3: Сравнение результатов, полученных при помощи генетического алгоритма с с алгоритмом полного перебора.

горитма полного перебора для такой матрицы составляет около 100 часов при размере выборки k=3 (для большего размера выборки время увеличивается). Для генетического алгоритма были выбраны следующие параметры: размер популяции 10%, число итераций и турнирных отборов 10. Результаты полностью совпали, а время работы генетического алгоритма составило 2 секунды (см. Таб. 3).

Теперь рассмотрим результаты поиска 15 наиболее частых вхождений для $k = 5$ при размере популяции 10% и 1% в зависимости от числа итераций и турнирных отборов (см. Таб. 4).

Из таблицы хорошо видно, что чем больше число турниров, тем сложнее получить информацию о следующих по частоте вхождениях. При этом если уменьшить размер популяции до 1%, то сходимость к единственно-

Размер популяции 10%		Размер популяции 1%	
Число итераций/ число отборов		Число итераций/ число отборов	
10/10	100/100	10/10	100/100
1727	1727	1727	1727
1708	1702	1702	152
1702	521	1689	1
1701	516	1688	1
1689	515	1688	
1688	496	1684	
1688	480	1680	
1684	179	1678	
1681	164	1671	
1680	151	1632	
1678	150	1629	
1676	148	535	
1671	72	527	
1668	68	515	
1667	30		
1661			

Таблица 4: 15 наиболее частых вхождений.

му решению становится еще нагляднее, а точность нахождения следующих по частоте вхождений падает (в данном случае не найдены вхождения с частотой 1708 раз, 1701 раз и далее).

Была рассмотрена также модификация алгоритма, в которой при реализации турнирного отбора особь-потомок заменяет особь-родителя не только в том случае, когда она не хуже, но и если такой особи в популяции не существует. Процесс сходимости к единственному решению идет медленнее, но наилучшие вхождения могут быть найдены при помощи данной модификации алгоритма.

Данный алгоритм был также протестирован на первой модели. Эксперименты были проведены для различных вероятностей p=0.0025, 0.005, 0.0125, то есть сильно разреженной матрице. Были получены следующие результаты: число наилучших вхождений при k=3 было равно 2 во всех случаях. При сравнении результатов, полученных при работе генетического алгоритма с алгоритмом полного перебора, оказалось, что генетический алгоритм ищет не все такие вхождения. Увеличения числа турнирных отборов и числа итераций, а также попытка изменить механизм мутаций, эффекта не дает. Это связано с тем, что при данном способе генерации популяции в случае разреженной матрицы очень мала вероятность выбора k элементов для множества размера k таким образом, чтобы частота вхождения была наибольшей. Также очень мала вероятность того, что подверженный мутации элемент, будет заменен элементом, вхождение которого в множество, сделает это множество часто встречающимся. Это происходит и в том случае, когда в популяцию отбираются только различные гены.

Алгоритм	% матриц для которого решение совпадает с алгоритмом полного перебора	max отклонение (в %) от алгоритма полного перебора
Генетический алгоритм	40	25
Генетический алгоритм с различными особями	85	15

Таблица 5: Сравнение результатов, полученных при помощи генетического алгоритма с алгоритмом полного перебора на разреженной матрице.

Если матрица не сильно разрежена, то есть p=0.1, то необходимо выбирать достаточно большой размер популяции, не менее 50% особей от числа членов популяции, в противном случае вероятность того, что в популяцию попадут часто встречающиеся комбинации значительно уменьшается. Были проведены вычисления для генетического алгоритма и модификации генетического алгоритма с различными особями, в которой при реализации турнирного отбора особь-потомок заменяет особь-родителя не только в том случае, когда она не хуже, но и если такой особи в популяции не существует. Алгоритмы были протестированы на 50 различных разреженных матрицах размера 400 на 1000 (матрицы были сгенерированы при помощи генератора псевдослучайных чисел). Размер популяции — 1000, число итераций 1000, число турнирных отборов — 100. Результаты сведены в таблицу (см. Таб. 5).

Если стоит задача поиска наилучшего решения при фиксации одного товара, то ситуация зависит от того, насколько часто данный товар входит в матрицу. При высокой частоте вхождения алгоритм находит не только наибольшее вхождение, но и следующие по частоте вхождения. При этом популяцию для работы данной модификации алгоритма следует брать достаточно большую. В том случае, если мы фиксируем два товара, то оба они должны быть часто встречающимися, иначе нахождение наилучшего решения не гарантируется, поскольку работа алгоритма становится аналогичной его работе с разреженной матрицей и с помощью данного алгоритма можно найти только некоторые вхождения.

Литература

[1] М. Гэри, Д. Джонсон, Вычислительные машины и труднорешаемые задачи: Пер. с англ.- М.: Мир, 1982.

[2] S. Cook, The complexity of theorem proving procedures, Proc. Third Annual ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 1971, pp. 151-158.

[3] D.E. Goldberg, *Genetic algorithms in Search, Optimization and Machine Learning*, 1989, 412 pp.

[4] Al-Sultan K., Hussain M., Nizamio M., A genetic algorithm for the set covering problem, *J. Oper. Res. Soc.*, 1996, v. 47, pp. 702–709.

[5] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, New York, 1996.

[6] Melanie Mitchell. *An Introduction to Genetic Algorithms.* The MIT Press, Cambridge, MA, 1996.

[7] M. Blum, R.M. Karp, C.H. Papadimitriou, O. Vornberger, M. Yannakakis, The complexity of testing whether a graph is a superconcentrator, Inform. Process. Lett. **13** (1981) 164-167.

[8] R. Agrawal, T. Imilienski, A. Swami, Mining associasion rules between sets of items in large databases, In: Proc. 1993 ACM-SIGMOD Conference on Management of Data, pp. 205-216.

[9] E. Boros, V. Gurvich, L. Khachiyan, K. Makino, On maximal frequent and minimal infriquent sets in binary matrices, Proc. STACS-2002. LNCS **2285** (2002).

[10] S. Brin, R. Motwani, J. Ullman, S. Tsur, Dynamic itemset counting and implication rules for market basket data, In: Proc. 1997 ACM-SIGMOD Conference on Management of Data, pp. 255-264.

[11] H. Mannila, H. Toivonen, A.I. Verkamo, Discovery of frequent episodes in event sequences, Data Mining and Knowledge Discovery, **1** (1997) 259-289.

[12] R.H. Sloan, K. Takata, G. Turan, On frequent sets of Boolean matrices, Annals of Mathematics and Artificial Intelligence, **24** (1998) 1-4.

[13] Venkatesh Ganti, Johannes Gehrke, Raghu Ramakrisnan. Mining Very Large Databases. IEEE Computer, August 1999, pp. 38-45.

[14] U. Feige, Relations between average case complexity and approximation complexity, Proc. Annual ACM Symp. STOC'02, May 9–11, Montreal, Canada.