ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

УДК 519.7

А. Е. Устюжанин

МЕТОД ПОСТРОЕНИЯ СИСТЕМЫ ПАМЯТИ ДЛЯ ХРАНЕНИЯ И ПОИСКА МНОГОМЕРНЫХ ПРОСТРАНСТВЕННО-ВРЕМЕННЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Рассмотрена возможность использования хаотических процессоров как систем хранения и поиска информации, представленной в виде пространственно-временных последовательностей. Основными проблемами построения систем памяти являются значительные объемы информации и нечеткость критериев сравнения двух последовательностей. Рассмотрена функция расстояния для сравнения пространственно-временных последовательностей и способ организации информации в виде хаотического процессора на основе структур пространственных индексов. На основе реальных данных, используемых в качестве стандартного набора при разработке алгоритмов и структур данных многомерной индексации, оценена применимость и эффективность данного подхода.

Во многих задачах, связанных с хранением и анализом данных, основным способом представления информации является кодирование данных в виде пространственно-временных последовательностей. Развитие таких технологий, как Global Positioning System (GPS), мобильных систем и датчиков пространственных координат объектов дает доступ к огромным объемам данных и, как следствие, возрастает потребность в эффективных алгоритмах их анализа и хранения. Например, рассмотрим задачу, когда мобильный объект наблюдения оснащен сенсором, регистрирующим положение объекта в пространстве с определенной временной частотой. Данные о траектории движения объекта сохраняются для дальнейшей обработки в базе пространственно-временных данных. Запросы к такой базе данных могут состоять в поиске траекторий, похожих друг на друга, в поиске возможных продолжений по известному началу, поиску предысторий по известному отрезку траекторий и т.д. На сегодняшний день универсальных способов работы с такими данными нет, и каждый подход имеет свои достоинства и недостатки.

В настоящей работе рассмотрена проблема построения хранилища пространственно-временных последовательностей (траекторий), которое позволяет находить траектории, содержащие подпоследовательности, похожие на поисковый запрос. Именно возможность ответа

на запросы такого типа является ключевой при реализации информационных систем, связанных с временными последовательностями, поскольку она позволяет не только найти похожие траектории, но и предсказать возможные дальнейшие сценарии развития систем. Описание метода хранения информации изложено на примере данных, взятых из современного источника, который приведен далее.

Иллюстративный пример. Работа с пространственно-временными последовательностями востребована во многих областях современных технологий, как-то: анализ финансовых показателей, мультимедийная информация, сложное человеко-компьютерное взаимодействие (использование трехмерных манипуляторов, таких как перчатки или шлемы виртуальной реальности). В каждой из этих областей можно выделить набор параметров, характеризующих интересующий нас процесс в произвольный момент времени. Для простоты описания положим, что число параметров в таком наборе параметров не зависит от времени и численные значения принадлежат интервалу $[0, 1]$. В качестве иллюстративного примера и примера для демонстрации эффективности подхода воспользуемся данными из набора слов австралийского языка глухонемых [1]. Использование этих данных для сравнения алгоритмов индексации и поиска встречается в работах [2–4]. Эта база данных создавалась сканированием пространственных параметров траектории движения руки носителя языка. Длительность последовательностей варьируется от 30 до 100 элементов. Каждая точка последовательности характеризуется девятью параметрами:

- координаты руки человека (x, y, z);
- угол поворота ладони;
- угол сгиба каждого пальца руки.

Частота сбора данных составляла примерно 23 Гц. Пример зависимости изменения координат (X, Z) от времени для слов “Да”, записанных двумя носителями, и словом “Нет”, записанным одним носителем, приведен на рис. 1. Траектории, отмеченные звездочками и плюсами, — слово “Нет”, кружками — слово “Да”. На данном примере мы можем убедиться в том, что разные слова выглядят по-разному и одинаковые слова схожи друг с другом, но могут отличаться всплесками, длиной траекторий и сдвигами одинаковых участков траектории по времени. Аналогично данным работ [2–5], за основу для экспериментов выбрана база последовательностей, содержащая несколько различных классов объектов (слов), где каждому классу принадлежат от 5 до 10 элементов.

Критерий близости пространственно-временных последовательностей. Поиск ключа в ассоциативной памяти по неполному и неточному запросу опирается на сравнение двух последовательностей

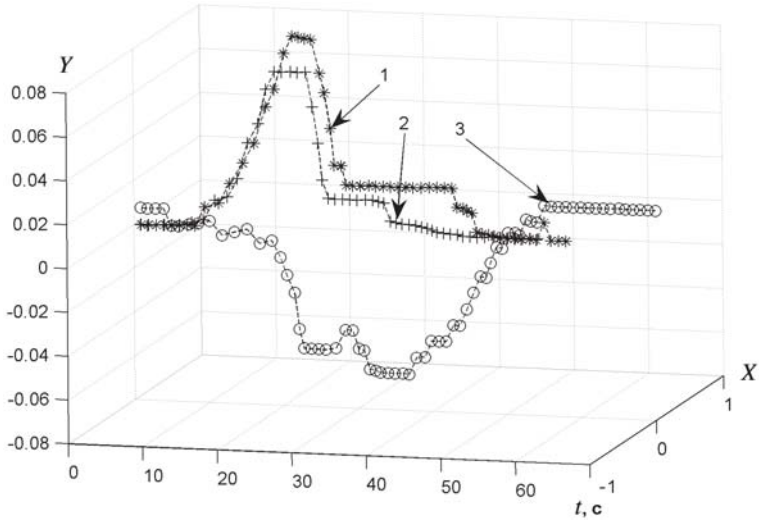

Рис. 1. Пространственно-временные последовательности координат (X, Y) манипулятора, соответствующие словам "Да" (*3*) и "Нет" (*1, 2*) на языке глухонемых

и вычисление количественной меры близости этих последовательностей. Выбор такой меры близости во многом определяет применимость и эффективность подхода в целом. Опишем особенности последовательностей, которые необходимо учесть при выборе критерия близости.

Как правило, сигнал на выходе блока сенсоров появляется с определенной частотой — частотой дискретизации. Выбор этой частоты относительно произволен, поэтому одинаковые условия окружающей среды могут быть представлены по-разному при выборе разных частот дискретизации. Если управляемая система способна перемещаться в окружающей среде (например, мобильный робот), то восприятие одного и того же объекта этой среды при движении с разными скоростями тоже будет выглядеть по-разному. Чтобы не зависеть от скорости движения и частоты дискретизации, выбранная функция должна предусматривать изменения масштаба сравниваемых сигналов.

При сравнении последовательностей n-мерных векторов S_1 и S_2 одинаковой длины k евклидова мера близости этих последовательностей определяется следующим образом:

$$D(S_1, S_2) = \frac{1}{k} \sum_{1 \leqslant i \leqslant k} d(S_1[i], S_2[i]), \tag{1}$$

где d — расстояние между n-мерными точками $S_i[i]$ и $S_2[i]$:

$$d(S_1[i], S_2[i]) = \Big(\sum_{1 \leqslant j \leqslant n} |S_1[i, j] - S_2[i, j]|^2 \Big)^{1/2}. \tag{2}$$

В случае, если длины S_1 и S_2 не равны, выбираем между урезанием длинной последовательности (вычисление минимума евклидовых мер среди возможных подпоследовательностей длинной последовательности с короткой последовательностью) и добавлением нулей к короткой последовательности. Зачастую ни один из этих вариантов не является приемлемым. Наша метрика должна быть приспособлена для сравнения последовательностей вне зависимости от их длины, опираясь лишь на похожести их элементов.

Значительные различия в сравниваемых последовательностях, возникающие на небольших участках, могут быть вызваны самыми разными причинами, например ошибкой сенсоров или ошибкой человека, вводящего данные. Использование евклидовой метрики в этом случае даст значительное расхождение. Искомая метрика должна уметь пропускать такие всплески (промахи) и давать небольшую разницу последовательностей, при условии, что в преобладающем множестве точек последовательности совпадают.

Функция вычисления критерия должна быть одновременно достаточно выразительна (т.е. степень различия значений меры должна соответствовать степени различий последовательностей) и в то же время проста в вычислении, чтобы ее реализация не требовала значительных вычислительных ресурсов.

Такие метрики, как Dynamic Time Warping (DTW) [6], Longest Common Subsequence (LCSS) [2], подразумевают, что при сравнении двух последовательностей каждая из них является полноценным экземпляром, который можно рассматривать как нечто целое. В нашем случае необходимо добиться возможности работы с фрагментарными последовательностями – сравнивать части с целым и получать адекватные оценки похожести рассматриваемых траекторий. Такое свойство метрики позволит извлекать сохраненные элементы, пользуясь лишь частично определенными запросами.

Применяемые на сегодняшний день метрики не удовлетворяют всем поставленным условиям. Например, евклидова метрика оказывается непригодной для сравнения последовательностей, полученных при различных частотах дискретизации. Функция расстояния, основанная на методе DTW, позволяет сопоставить две последовательности разной длины, но она излишне чувствительно к шумам. Функция расстояния, по которой вычисляют наибольшую общую последовательность, наиболее устойчива к шумам и различным аберрациям запроса, но не позволяет сравнивать фрагмент с целым, размеры которого значительно больше размеров фрагмента.

Рассмотрим алгоритм вычисления меры близости двух последовательностей: последовательности-запроса Q и некоторой известной последовательности S. Для этого введем ряд определений.

Определение 1. *Минимальным ограничивающим параллелепипедом*, или *минимальным ограничивающим регионом* (МОП или MBR – minimum bounding rectangle) для последовательности $\{S_i\}$, все элементы которой принадлежат $\Re^n$ ($S_i \in \Re^n$), будем называть n-мерный параллелепипед минимального объема со сторонами, параллельными осям координат, который целиком содержит точки этой последовательности.

Отметим, что MBR($\{Si\}$) определяется двумя точками в $\Re^n$: $P_{\min}$ и $P_{\max}$:

$$P_{\min}(x_1, \ldots, x_n)\colon x_i = \min_j(S_{ji});$$

$$P_{\max}(x_1, \ldots, x_n)\colon x_i = \max_j(S_{ij}).$$

Определение 2. *Расширением параллелепипеда* R на величину δ получен параллелепипед R', такой, что центр R' совпадает с центром параллелепипеда R, а координаты вершин R' отличаются от координат соответствующих вершин R на величину δ вдоль каждой из осей в направлении от центра R.

Другими словами, операция расширения параллелепипеда MBR($\{Qi\}$), определяемого точками $P_{\min}(x_{11}, \ldots, x_{1n})$ и $P_{\max}(x_{21}, \ldots$ $\ldots, x_{2n})$, на величину δ образует параллелепипед, определяемый точками $P_{\min}(x_{11} - \delta, \ldots, x_{1n} - \delta)$ и $P_{\max}(x_{21} + \delta, \ldots, x_{2n} + \delta)$.

Операция расширения региона позволяет работать с зашумленными сигналами.

Определение 3. *Разбиением последовательности* $\{S_i\}$ длиной N на K подпоследовательностей ($K < N$) будем называть такой набор из $K - 1$ целых чисел $\{M_i\}\colon 1 < M_i < N$, и $M_i \neq M_j$, если $i \neq j$, который определяет K частей исходной последовательности: $\{X_1, \ldots, X_{m0-1}\}$, $\{X_{m0}, \ldots, X_{m1-1}\}, \ldots, \{X_{mk-1}, X_N\}$. Среднюю длину таких подпоследовательностей λ будем называть *диаметром разбиения*.

Пусть даны последовательность $\{S_i\}$ и запрос $\{Q_i\}$, для которых необходимо вычислить меру близости. Для этого разобьем $\{S_i\}$ на подпоследовательности, и для каждой из подпоследовательностей построим минимальный ограничивающий параллелепипед R_i. Каждый параллелепипед расширим на величину δ. Таким образом, последовательности $\{S_i\}$ ставим в соответствие последовательность $\{R_i\}$. Заметим, что между любыми двумя R_i, R_j $f\{R_i\}$ установлены соотношения *непосредственного следования* – для них можно определить, является ли R_i непосредственным предшественником R_j или нет. Тогда процедура вычисления меры похожести последовательностей $\{Q_i\}$ и $\{S_i\}$ выглядит следующим образом.

1. По последовательности $\{S_i\}$ строим $\{R_i\}$;

2. Для каждого элемента $Q_i \in \{Q_i\}$ проверяем:

а) принадлежит ли Q_i какому-либо $R_i \in \{R_i\}$, если принадлежит, то запоминаем индексы этих регионов (индексы текущего шага); если принадлежность найдена, то увеличиваем значение меры близости на величину WeightHit;

б) сравниваем индексы текущего шага с индексами прошлого шага. Если оказывается, что индексы текущего следуют за индексами прошлого, то меру близости увеличиваем на величину WeightFollow.

В наших численных сравнениях значения величин WeightHit и WeightFollow выбираются равными 2 и 1 соответственно.

Описанный способ сравнения пространственно-временных последовательностей удовлетворяет выдвинутым требованиям. Действительно, ни разница частот дискретизации, ни разница в длинах последовательностей не влияют на сравнение запроса с последовательностью МОП, поскольку критическим для получения высокой оценки является похожесть очертаний форм, а не длина или число точек в исходной последовательности. Всплески, в отличие от евклидовой метрики и метрики DTW, не вносят значительных изменений в полученные веса, поскольку точки, выпавшие за пределы последовательностей МОП, просто не учитываются. В случае, когда запрос является незначительным фрагментом последовательности, предложенная процедура оказывается применимой, поскольку для вычисления веса не имеет значения время начала и время окончания последовательности — важно лишь совпадение пространственных координат и повторение последовательности попадания точек в коридор, определяемой n-параллелепипедами. Если использовать предложенный ранее алгоритм, то эффективность данной функции можно оценить как $O\left(\frac{m_1 \cdot m_2}{\lambda}\right)$, где m_1 — длина $\{S_i\}$, m_2 — длина $\{Q_i\}$, а λ — диаметр разбиения $\{S_i\}$. Такая оценка справедлива при условии, что операции проверки принадлежности региону и проверки следования индексов регионов выполняются за время $O(1)$. Далее покажем, что такую оценку эффективности можно улучшить.

Численное сравнение выразительности критериев близости последовательностей. Оценим численно, насколько предложенная функция близости выразительна для различных последовательностей. Такую оценку мы проведем согласно методике, предложенной Кео и Касетти в работе [7], которая сводится к следующему. Пусть имеется множество D последовательностей $\{S_i\}$, для каждой из которых явно задана принадлежность некоторому классу C_j. Каждому классу принадлежит более одной последовательности. Тогда из множества D

отбирается набор тестовых последовательностей — подмножество T, а остальные используются как набор классифицирующих последовательностей — подмножество L. Затем для каждой $\{S_i\}$ из T вычисляется мера похожести этой последовательности на последовательности подмножества L и выбирается та, значение меры которой оказалось наибольшим. Если классы тестовой последовательности и найденной последовательности совпали, то этот опыт считается удачным, если классы оказались различны, то счетчик ошибок N_{err} увеличивается на единицу. Затем вычисляется доля ошибочных распознаваний согласно следующей формуле:

$$\rho = \frac{N_{err}}{M}, \tag{3}$$

где N_{err} — число ошибочных распознаваний; M — число проведенных экспериментов (мощность множества T). Чем меньше значение ρ, тем более выразительной считается предложенная функция.

При проведении сравнительных оценок были рассмотрены три набора последовательностей:

1) пять различных слов (классов), записанных несколькими (5. . . 7) дикторами;

2) десять различных слов (классов), записанных другими (5. . . 7) дикторами;

3) взяты десять различных слов, для каждого из которых генерируется 4 искаженных последовательности посредством внесения пространственных (шумов) и временных (растяжение/сжатие) аберраций.

Расчеты доли ошибочных распознаваний $\rho(3)$ для указанных наборов данных содержатся в табл. 1.

Таблица 1

Значения доли ошибочных распознаваний ρ для различных методов расчета меры близости и для различных тестовых последовательностей

Методы расчета	Тестовые последовательности		
	ASL (5 классов)	ASL (10 классов)	Данные ASL + шум
Евклидова мера	0,01	0,05	0,38
LCSS ($d = 20$, $e = 0{,}1$)	0,01	0,08	0,32
MBRMeasure $\lambda = 5$, $\delta = 0{,}05$)	0,04	0,10	0,25

При работе с набором последовательностей из 5. . . 10 классов видно, что евклидова мера более выразительна, однако при внесении возмущений, связанных с изменением частоты дискретизации (растяжений/сжатий), ее избирательность хуже избирательности функции

MBRMeasure. Необходимо отметить, что согласно данным работы [7] величина доли ошибочных распознаваний, не превышающая 0,10, считается приемлемой характеристикой меры.

Значительным преимуществом предложенного способа оценки похожести последовательностей является возможность оценки по неполным запросам, т.е. когда на вход функции оценки подается лишь часть сохраненной последовательности. В табл. 2 приведены интервалы изменений значений ρ при распознавании запросов, длина которых составляет 50 % и 25 % общей длины сохраненной последовательности.

Таблица 2

Интервалы значений доли ошибочных распознаваний ρ для различных методов расчета меры близости, различных тестовых последовательностей и для различных длин запросов

Методы расчета	Тестовые последовательности					
	ASL, 5 классов, 50 %	ASL, 5 классов, 25 %	ASL, 10 классов, 50 %	ASL, 10 классов, 25 %	Данные ASL + шум, 50 %	Данные ASL + шум, 25 %
Евклидова мера	0,04–0,24	0,04–0,60	0,03–0,1	0,05–0,52	0,22–0,40	0,54–0,68
LCSS ($d = 20$, $e = 0,1$)	0,04–0,72	0,12–0,68	0,18–0,75	0,08–0,72	0,26–0,86	0,44–0,92
MBRMeasure ($\lambda = 5$, $\delta = 0{,}05$)	0,04–0,36	0,0–0,28	0,08–0,25	0,13–0,33	0,24–0,44	0,20–0,64

Диапазоны ошибочных распознаваний соответствуют разбросу показателей, которые возникают при задании различных начальных точек выбора фрагмента запроса. Начальные элементы различных слов в ASL достаточно похожи друг на друга и поэтому доля ошибочных распознаваний выше. Средние части траекторий, наоборот, значительно отличаются друг от друга и доля ошибок на этих участках падает. В целом видно, что выразительность предложенного критерия не хуже выразительности евклидовой метрики.

Экспериментальные данные, приведенные в табл. 1 и 2, показывают обоснованность выбора критерия близости пространственно-временных последовательностей на основе представления одной из них последовательностью n-мерных параллелепипедов.

Построение подсистемы памяти на основе хаотического процессора. Рассмотрим построение памяти для числовых последовательностей на основе хаотических процессоров, как это описано А.С. Дмитриевым [7]. В работе [7] предлагается построить такое

отображение F, которое для элементов последовательности $\{X_i\}$ реализует следующее соотношение:

$$X_{n+1} = F(X_n, \mathrm{a}), \tag{4}$$

т.е. каждому элементу последовательности в этом отображении соответствует последующий элемент последовательности. Здесь "а" — вектор управляющих параметров, который может быть использован для управления отображением. Таким образом, воспроизведение последовательности становится возможным путем рекурсивного применения отображения F к одному из известных элементов X_k, т.е. зная элемент X_k и отображение F, можно воспроизвести элементы последовательности $X_{k+1}, X_{k+2}, \ldots, X_n$, для этого искусственно добавляется переход $X_n \to X_0$, т.е. последовательность закольцовывается, что дает возможность многократного итеративного применения F к любому элементу последовательности и воспроизведения последовательности целиком.

Использование описанного подхода для хранения/воспроизведения требует предварительной обработки исходных элементов последовательности на предмет исключения повторяющихся элементов путем расширения используемого алфавита.

Схема такого преобразования приведена в работах [7–10]. Благодаря предварительной обработке и способу построения, преобразование F позволяет работать с неточными исходными данными: если известный элемент последовательности отличается от сохраненного элемента на некоторую величину $X'_i = X_i + dX$ (где dX — допустимый шум сигнала), то при незначительных величинах этого шума ($dX < dX_{\max}$, где $dX_{\max}$ — максимальная величина шума, заданная изначально при построении F) отображение восстановит исходный сигнал X_i. Другими словами, в отображении F реализуется аттрактор соответствующей сохраненной последовательности и бассейн притяжения этого аттрактора, точки которого сходятся к аттрактору в результате итеративного применения преобразования F. Для запоминания нескольких последовательностей в отображении F реализуется несколько аттракторов. Из-за использования процедур фильтрации повторяющихся элементов и замыкания последнего элемента на первый, алгоритм построения нескольких аттракторов принципиально не отличается от алгоритма построения одного аттрактора.

На данный момент существует подход, реализующий хаотический процессор, на базе коннекционистских структур [11], при использовании которого сложность реализации значительно возрастает при увеличении размерности элементов последовательности.

Для снятия этого ограничения предлагается схема хранения данных, основанная на основе аппроксимации бассейна аттрактора посредством многомерных геометрических примитивов и хранения этих элементов в многомерной индексной структуре. Остановимся подробней на данном способе построения аттрактора и процедуре поиска последовательности.

Реализация хаотического процессора. Пусть имеется последовательность $\{X_i\}$ и ее идентификатор I, которые необходимо запомнить. Рассмотрим $\{X_i\}$ как аттрактор некоторого отображения F и построим бассейн этого аттрактора. Для этого выполним процедуру разбиения последовательности $\{X_i\}$ на подпоследовательности и ограничим каждую из них соответствующим n-мерным параллелепипедом — МОП. Тогда полученная последовательность параллелепипедов и будет являться бассейном нашего аттрактора.

Каждому МОП поставим в соответствие следующие числа:

- идентификатор всей последовательности $\{X_i\}$;
- идентификатор элемента X_i в исходной последовательности, который является первым элементом в $\{Q_i\}$;
- точку следующей подпоследовательности, наиболее удаленную от границ соответствующего МОП;
- признак конца последовательности.

Получившийся набор параллелепипедов и связанных с ними данных сохраняется в структуре многомерного индекса — в R-дереве с приоритетами [12]. Такой способ организации позволяет сохранять n-мерные параллелепипеды (регионы) и эффективно определять принадлежность произвольной точки одному или нескольким сохраненным регионам.

Таким образом, элементы последовательности $\{X_i\}$ разбиты на ограничивающие параллелепипеды и сохранена их упорядоченная совокупность в многомерном дереве, которое в силу своего построения позволяет эффективно отвечать на поисковые запросы о принадлежности произвольной точки пространства одному или нескольким параллелепипедам. По оценкам, приведенным в работе [12], время на обработку поискового запроса составляет $O((N/B)^{1-1/n} + T/B)$, где N — число параллелепипедов размерности n, сохраненных в дереве, B — размер дискового блока, T — число полученных ответов на запрос. В работе [12] приведены теоретические и эмпирические оценки сложности выполнения операций добавления и удаления элементов дерева, которые превосходят на сегодняшний день показатели аналогичных многомерных индексных структур.

Хаотический процессор реализуется как надстройка над индексной структурой. Способ построения описанной индексной структуры линейно масштабируется для сохранения набора последовательностей.

Если оказывается, что при сохранении двух разных последовательностей набор ограничивающих регионов совпадает, то в этом случае необходимо либо признать, что эти последовательности для задачи хранения являются идентичными, и исключить вторую последовательность из рассмотрения, либо повторно разбить конфликтующую последовательность с меньшим диаметром разбиения. Реализация функции $F(x)$, в случае использования R-дерева как основы хаотического процессора, будет состоять в получении из R-дерева списка регионов, которым принадлежит точка x, и выборе из него того, центр которого находится к x ближе всего.

Сложность операции вычисления меры близости при использовании данного подхода составит $O(m(N/B)^{1-1/n})$, где m — длина запроса Q, N — число параллелепипедов размерности n, сохраненных в дереве, B — размер дискового блока.

Рассмотрим процедуру поиска сохраненной последовательности по поисковому запросу $\{Q_i\}$. Для этого запроса ассоциативная память должна дать один из возможных ответов:

- идентификатор последовательности, которая содержит подпоследовательность, похожую на $\{Q_i\}$, в смысле критерия, описанного ранее;
- "*необходимы дополнительные данные*", в случае, если нельзя однозначно сопоставить $\{Q_i\}$ с сохраненными последовательностями, т.е. $\{S_i\}$ — подпоследовательность более чем одной последовательности, и для того чтобы однозначно ответить на поисковый запрос, необходимо дополнить поисковый запрос, т.е. расширить его дополнительными точками;
- *null* в случае, если невозможно дать ни один из перечисленных ответов, то $\{Q_i\}$ не соответствует ни одной из сохраненных последовательностей.

Процедура поиска наиболее похожей последовательности по запросу Q состоит из двух логических частей: оценка похожести сохраненных последовательностей на запрос Q и выбор одного из перечисленных ответов по результатам оценки.

Оценки для первого шага получаются итеративным образом, аналогичным вычислению меры похожести запроса Q последовательности S_i: последовательно рассматриваются точки запроса, но учитываются все регионы, в которые попала точка запроса. Таким образом, значения мер близости Q и S_i для всех сохраненных последовательностей вычисляются за один проход по точкам запроса. Все пары $\{S_i, v_i\}$ ранжируются по убыванию v_i, и в зависимости от того, сколько элементов содержат максимальное значение $v_{\max} = \mathrm{supp}(v_i)$, делается второй шаг.

Если максимальное значение представлено лишь в одной паре $\{S_i, v_i\}$, то результатом поиска и будет последовательность S_i; если таких значений несколько, то результатом поиска будет ответ "необходимы дополнительные данные". Если таблица содержит только нулевые значения, это означает заметную непохожесть запроса Q на сохраненные последовательности и приведет к ответу "*null*" поисковой процедуры.

Таким образом, описанная система построения хаотического процессора на основе R-деревьев и с использованием описанной метрики позволяет сохранять последовательности и искать похожие последовательности по заданным подпоследовательностям. Теоретические оценки производительности работы предлагаемого хаотического процессора во многом совпадают с характеристиками работы используемой структуры многомерного индексирования. На рис. 2 показан пример проекции на плоскость XY покрытия последовательности двумерных точек набором прямоугольников.

Приведем эмпирические оценки производительности и ресурсоемкости такой поисковой системы.

Оценка ресурсоемкости и производительности хаотического процессора. Для оценки общей эффективности работы хаотического процессора сравним показатели объема занимаемой памяти и времени выполнения операций сохранения и поиска. В качестве примера используемых данных возьмем последовательности австралийского языка глухонемых. Первый набор испытаний показывает зависимость

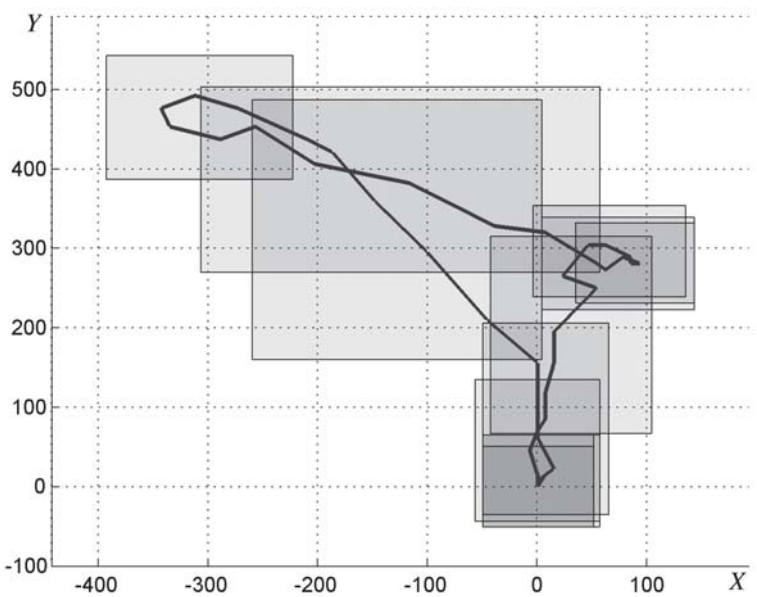

Рис. 2. Пример проекции на плоскость XY покрытия последовательности двумерных точек набором прямоугольников

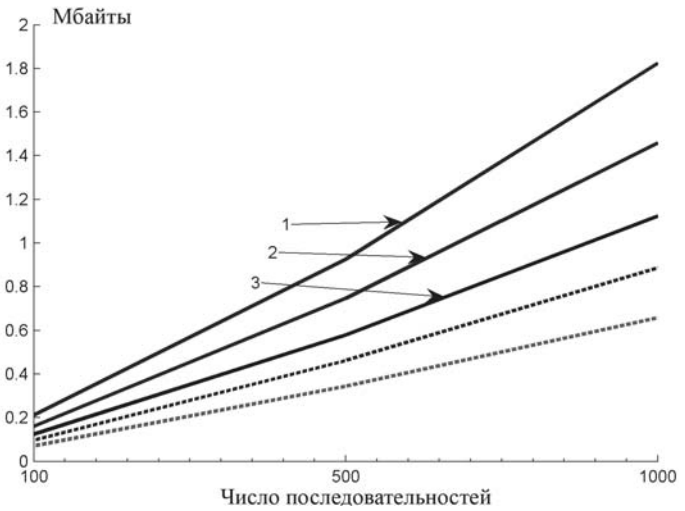

Рис. 3. Зависимость объема памяти от числа сохраненных последовательностей для различных типов памяти и размерности элементов последовательности. Штриховые линии — объем последовательностей в текстовом представлении (нижняя линия — соответствует размерности 3, верхняя — 4), сплошные линии — объем хаотического процессора; *1,2,3* — размерности *5,4,3* соответственно

объема памяти, необходимого для хранения хаотического процессора, от объема хранимых данных (числа последовательностей) и их размерности. Последовательности, используемые в экспериментах, представляют собой цепочки многомерных данных длиной около 100 элементов. Результаты опыта представлены на рис. 3.

Как видно из рис. 3, объем памяти, необходимый для хранения данных хаотического процессора, растет пропорционально размерности (D) хранимых точек и пропорционально объему запоминаемых данных, что находится в соответствии с характеристиками объема памяти, необходимой для хранения приоритетного R-дерева [13].

Временные затраты на создание (bulk loading) хаотического процессора при работе с данными, аналогичными предыдущему эксперименту, отображены на рис. 4.

Полученные результаты характером зависимости и порядком временных величин совпадают с характеристиками приоритетного R-дерева

Следующая серия испытаний представляет наибольший интерес как операция, выполняемая наиболее часто, — операция поиска данных. Проанализируем скорость работы хаотического процессора на зависимость от длины запроса и объема сохраненных данных. Полученные показатели сравним со скоростными показателями при поиске линейным перебором, используя евклидову метрику. Исходные данные для испытаний — пятимерные последовательности, использованные в предыдущих опытах.

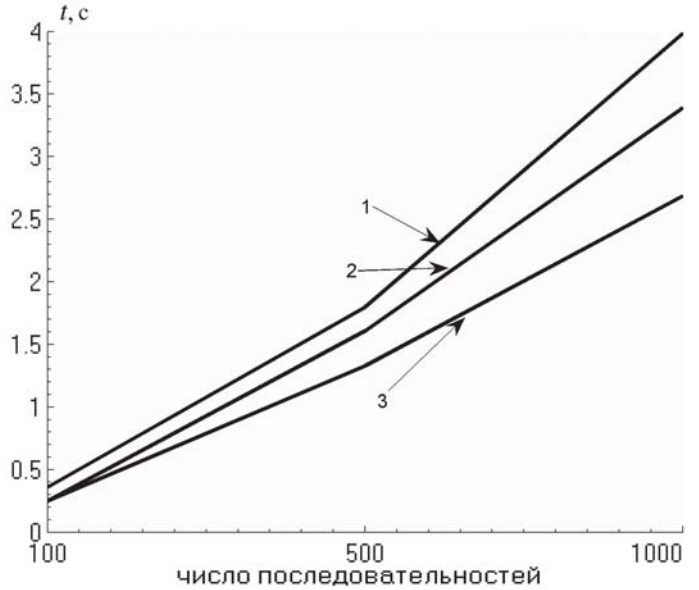

Рис. 4. Длительности загрузки разного числа последовательностей различных размерностей в хаотический процессор:
1,2,3 — размерности *5,4,3* соответственно

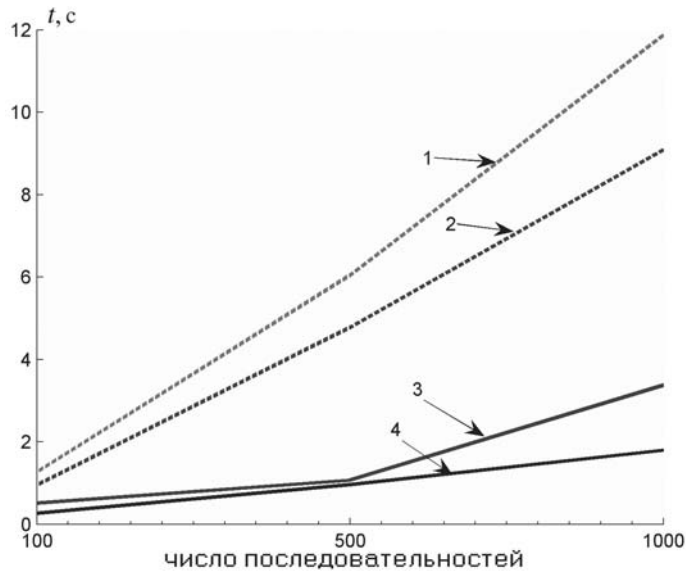

Рис. 5. Зависимость времени поиска запросов различных длин от числа сохраненных последовательностей. Штриховые линии соответствуют алгоритму, основанному на евклидовой мере, сплошные — алгоритму, предложенному в настоящей работе (длина запроса составляет 55 % (*1,3*) и 25 % (*2,4*) от длины сохраненных последовательностей)

Как видно из полученных результатов, скорость работы хаотического процессора превышает стандартный алгоритм в зависимости от начальных условий в 2–6 раз. Наибольший выигрыш достигается в

случаях, когда длина запроса составляет около 50 % от длин сохраненных последовательностей.

Заключение. В данной работе предложен критерий оценки близости пространственно-временных последовательностей, отталкиваясь от которого, становится возможным эффективная конструкция многомерной индексной структуры на основе R-деревьев для построения хаотического процессора. Показан способ использования хаотического процессора для построения ассоциативной памяти. Принципиальной особенностью такой памяти является возможность адресации хранимых элементов по содержанию, которое может быть искажено или заключать в себе неполную информацию об элементе.

Применимость и эффективность предложенного подхода демонстрируется на примерах данных австралийского языка глухонемых, часто используемых как тестовый пример в современных работах по индексации многомерных данных. Полученные результаты показывают увеличение эффективности алгоритмов поиска по сравнению со стандартными алгоритмами в несколько раз.

Дальнейшие пути развития данного подхода включают в себя использование системы памяти на основе хаотических процессоров для реализации практических задач, таких как навигация мобильных роботов. Также представляет интерес поиск алгоритмов и структур данных, альтернативных R-деревьям, использование которых может улучшить временные оценки работы хаотического процессора.

СПИСОК ЛИТЕРАТУРЫ

1. База данных слов австралийского языка глухонемых. http://kdd.ics.uci.edu/databases/auslan/auslan.html
2. Vlachos M., Kollios G., Gunopulos D. Discovering similar multidimensional trajectories. In Proc. of ICDE, 2002.
3. Chen L., Ng R. T. On The Marriage of Lp-norms and Edit Distance, Proc. Intl. Conf. on Very Large Data Bases (VLDB). – P. 792–803, 2004.
4. Chen L., Özsu M., Oria V. Robust and fast similarity search for moving object trajectories, Proc. of ACM SIGMOD international conference on Management of data. – P. 491–502, 2005.
5. Vlachos M., Hadjieleftheriou M., Gunopulos D., Keogh E. Indexing Multi-Dimensional Time-Series with Support for Multiple Distance Measures. In Proc. of 9th SIGKDD, Washington, DC. – P. 216–225, 2003.
6. Berndt D, Clifford J. Using Dynamic Time Warping to Find Patterns in Time Series. In Proc. of KDD Workshop, 1994.
7. Eamonn J. Keogh, Shruti Kasetty On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In International Conference on Knowledge Discovery and Data Mining. – P. 102–111, Edmonton, Canada, July 2002.
8. Андреев Ю. В., Дмитриев А. С., Куминов Д. А. Хаотические процессоры // Успехи современной радиоэлектроники. – 1997. – № 10.

9. Жданов А. А., Устюжанин А. Е. Возможности использования технологии детерминированного хаоса в системах автономного адаптивного управления // Сб. трудов ИСП РАН. – 2001. – С. 141–180.
10. Андреев Ю. В., Бельский Ю. Л., Дмитриев А. С. Запись и восстановление информации с использованием устойчивых циклов двумерных и многомерных отображений // Радиотехника и электроника. – 1994. – Т. 39. – С. 114–123.
11. Andreyev Yu. V. Dmitriev A. S., Kuminov D. A. Pavlov V. V. Information processing in 1-d and 2-d map: recurrent and cellular neural networks implementation, CNNA'96, Seville, Spain, 1996.
12. Guttman A., 'R-trees: A Dynamic Index Structure for Spatial Searching', Proc. ACM SIGMOD Int. Conf. on Data Management, Boston, MA, 1984. – P. 47–57.
13. Lars Arge, Mark de Berg, Herman Haverkort, Ke Yi. The Priority R-Tree: A Practically Efficient and Worst-Case Optimal R-Tree. In Proc. of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD '04), Paris, France, June 2004. – P. 347–358.

Статья поступила в редакцию 26.09.2006

УДК 004.056.53

А. А. Кузнецов

СПОСОБЫ КАМУФЛИРОВАНИЯ ВИРТУАЛЬНЫХ ДИСКОВ

Рассмотрены методы обнаружения шифрованных виртуальных дисков и способы их камуфлирования.

Виртуальный диск представляет собой файл-контейнер, сформированный на жестком диске компьютера, который используется для хранения конфиденциальной информации в зашифрованном виде. Специальная программа при введении установленного пароля может подключать файл-контейнер к операционной системе таким образом, чтобы он был виден для приложений как обычный диск, и обеспечивать шифрование "на лету" всей переносимой на него информации [1].

Таким образом, виртуальный диск, с одной стороны, обеспечивает произвольный доступ к данным, с другой стороны, хранит их в едином защищенном контейнере, что упрощает задачу камуфлирования данных. Сокрытие самого факта наличия секретной информации является одной из основных задач при использовании виртуальных дисков. Кроме того, многие приложения создают на жестком диске временные файлы, которые могут содержать конфиденциальную информацию. Создание же временных файлов на виртуальном диске позволит предотвратить хранение секретных данных в открытом виде.