УДК 004.424.4
DOI: 10.30987/article_5bf3cb4ebb39c5.27869616

В.К. Гулаков, К.В. Гулаков, А.В. Солдатенков

ОЦЕНКА ЭФФЕКТИВНОСТИ ИСПОЛЬЗОВАНИЯ МАТРИЦ РАССТОЯНИЙ В ПРИБЛИЖЁННОМ ПОИСКЕ

Рассмотрена проблема оценки эффективности осуществления быстрого нечеткого поиска на основе матричных алгоритмов (AESA, LAESA, PAESA, iAESA, TLAESA) в текстовых словарях. Приведены результаты практического анализа количества вычисляемых расстояний в момент осуществления поиска.

Ключевые слова: метрические структуры данных, методы поиска, большая размерность, эффективность алгоритмов, приближённый поиск, матрицы расстояний.

V.K. Gulakov, K.V. Gulakov, A.V. Soldatenkov

ASSESSMENT OF EFFECTIVENESS OF DISTANCE MATRICES USE IN APPROXIMATE SEARCH

A problem in the assessment of the rapid fuzzy search effectiveness on the basis of matrix algorithms (AESA, LAESA, PAESA, iAESA, TLAESA) in text dictionaries is considered. There are shown results of the practical analysis of the number of distances under computation at the moment of search carrying out.

Key words: metric structure of data, methods of searches, large dimensionality, algorithm effectiveness, approximate search, distance matrices.

При решении задач, относящихся к различным предметным областям, часто возникает необходимость поиска ближайших объектов. Особенно актуальной она становится в задачах распознавания образов, поиска мультимедийных данных, анализа временных рядов, интеллектуального анализа данных, информационного поиска. К примеру, при анализе временных рядов нужно найти похожие данные в заданных последовательностях. При поиске изображений в базе данных производится поиск по определённому критерию подобия, например по цвету или форме.

Данная задача известна как поиск ближайшего соседа. Одним из способов её решения является использование метрических структур данных для хранения и обработки информации, а также применение методов индексации метрических пространств для ускорения доступа к информации. Как правило, они базируются на метрических структурах данных. Все они последовательно делят множество данных на подмножества, близкие к опорным точкам. Существуют различные подходы: матричный подход, сферическое разбиение, гиперплоскостное разбиение [6].

В метрическом пространстве $M = (X, d)$, определённом для множества объектов X, функция расстояния (метрика) $d : X \times X \rightarrow R$ вычисляет расстояния между двумя произвольными объектами из множества X и является мерой сходства объектов. Функция расстояния должна удовлетворять условию тождества ($d(x,y)=0 \Leftrightarrow x=y$), условию симметрии ($d(x,y)= d(y,x)$) и условию неравенства треугольника ($d(x,z) \leq d(x,y)+ (d(y,z))$. Также функция расстояния является неотрицательной ($\forall\ x,y \in X, d(x,y) \geq 0$).

Вычисление расстояний между объектами считается дорогостоящей операцией [5]. Информация в метрическом пространстве хранится в виде многомерных объектов, и стоимость вычисления расстояния возрастает с увеличением размерности. Основным критерием, по которому оценивают производительность методов поиска в метрических пространствах, является количество вычислений расстояний при выполнении поискового запроса. Существующие методы индексации метрических пространств разработаны таким образом, чтобы минимизировать количество вычислений расстояний.

В данной статье описываются методы индексации, которые используют матрицу расстояний между всеми или некоторыми объектами, принадлежащими к множеству объектов данных.

Введём некоторые обозначения. Пусть $U \subseteq X$ – множество объектов размера n, среди которых производится поиск объектов, похожих на q со степенью сходства r.

Будем осуществлять два типа поисковых запросов:

• диапазонный запрос: $(q,r)_d = \{u \in U \mid (d(u,q) \le r\}$;

• поиск ближайших соседей: $k\mathrm{NN}(q,r)_d = A$: $\forall\, u \in A, v \in \mathrm{U} - A, d(u,q) \le d(v,q), |A|=k$.

AESA. Первым предложенным алгоритмом, использующим матрицы расстояний, является AESA (Approximating and Eliminating Search Algorithm) [9]. Этот алгоритм предварительно рассчитывает все $n(n\text{-}1)/2$ расстояний для каждой пары из n объектов множества U и сохраняет их в матрице. На стадии поиска похожих объектов матрица используется для исключения элементов U из множества кандидатов в ближайшие соседи.

Для поиска ближайшего соседа AESA на очередном шаге выбирает опорную точку $u \in U$ и с её помощью отфильтровывает объекты, которые находятся заведомо далеко от объекта запроса q. Процесс повторяется, пока не будет выполнено сравнение со всеми элементами U либо они не будут заранее отброшены без непосредственного вычисления расстояния от них до q. Для выбора опорной точки на очередном шаге используется минимизирующий критерий

$$D(u) = \sum_{p \in P} |\, d(u,p) - d(p,q) \,|, \quad (1)$$

где P – множество опорных точек, для которых в процессе поиска были вычислены расстояния до объекта q. Расстояния $d(p,q)$ вычисляются во время поиска, а расстояния $d(u,p)$ рассчитаны заранее и помещены в матрицу. Критерий $D(u)$ определяет сумму минимальных границ расстояний между u и q, используя неравенство треугольника.

Ниже представлен алгоритм поиска ближайшего соседа $1NN(q)_d$ [4]:

1. *Инициализация*. Множество P и множество отфильтрованных элементов F изначально пустые: $P \leftarrow \emptyset, F \leftarrow \emptyset$.

$D(u) \leftarrow 0$, $D_{max_u} \leftarrow 0$, $r \leftarrow \infty$.

Шаги 2-5 повторяются, пока U не станет равным $U \cup F$, $P \cup F$.

2. *Выбор опорной точки.* Очередная опорная точка выбирается согласно условию минимизации критериев 1: $P \leftarrow \arg\min_{u \in U-F-P} D(u)$.

3. *Вычисление расстояния.* Между выбранной опорной точкой p и объектом запроса q вычисляется расстояние $d(p,q)$. Объект p добавляется к множеству P.

4. *Обновление кандидата – ближайшего соседа.* Если $d(p,q) < r$, то p становится текущим ближайшим соседом. Затем для каждого объекта в $U - F - P$ согласно формуле (1) обновляется критерий $\mathrm{D}(u)$:

$$D(u) \leftarrow D(u) + |\, d(u,p) - d(p,q) \,|,$$

$$D_{max_u} \leftarrow \max\left(D_{max_u}, |\, d(u,p) - d(p,q) \,|\right).$$

5. *Отсеивание.* Элементы $u \in U - F - P$, для которых $D_{max_u} > r$, исключаются из кандидатов в ближайшие соседи и добавляются к множеству F. Алгоритм возвращается к шагу 2.

AESA является одним из самых быстрых алгоритмов поиска, использующих матрицу расстояний [9]. Однако квадратичная пространственная сложность делает применение AESA непрактичным, если число элементов велико. Например, если множество содержит 10000 объектов, а расстояние между объектами занимает 4 байта, то матрица расстояний занимает 400 МВ. Можно сделать вывод, что метод эффективен только для малых объёмов данных (размер которых не превышает нескольких тысяч элементов). Тем не менее если вычисление расстояний обходится дорого, а высокая стоимость предвари-

тельной обработки позволительна, производительность поиска является гораздо более высокой по сравнению с другими методами.

LAESA. Метод под названием LAESA (Linear AESA) [8] использует уменьшенную матрицу расстояний с целью снижения пространственной сложности. На стадии предварительной обработки формируется множество B⊆*U*, состоящее из *m* элементов – базовых прототипов. В LAESA матрица расстояний содержит расстояния между каждым базовым прототипом и всеми остальными элементами множества *U*. Пространственная сложность и время предварительной подготовки LAESA оценивается как $O(m \cdot n)$.

Алгоритм LAESA практически идентичен AESA. Основное отличие - на шаге 2: выбор опорной точки происходит только среди базовых прототипов, т.е. среди элементов множества *B*. Отсеивание элементов на шаге 5 выполняется, только если они не принадлежат множеству *B*.

Количество вычислений расстояний во время поиска LAESA, как и в случае AESA, стремится к некоторому среднему значению, растущему с увеличением размерности данных и практически не зависящему от размера множества *U*.

Рекомендуемый способ выбора базовых прототипов состоит в том, чтобы формировать множество *B* из элементов, расположенных максимально далеко друг от друга. При этом эксперименты, проведённые авторами LAESA, показывают, что существует оптимальное значение *m*, определяемое опытным путём, при котором количество вычислений расстояний будет минимальным.

Преимущество LAESA перед AESA заключается в том, что LAESA, производя больше вычислений расстояний по время поиска, потребляет значительно меньше памяти для хранения матрицы расстояний. Время предварительной обработки данных при использовании LAESA также гораздо меньше, чем при использовании AESA.

PAESA. Алгоритм PAESA (the Projection-based AESA – проекция на основе AESA) [5] использует матрицу расстояний, хранящую *O* (*N*(*N*+1)/2) расстояний между объектами, которая является очень практичным решением для больших наборов данных. Основным различием между PAESA и AESA является применение нового ограничения расстояния нижней границы для аппроксимации и исключения. В AESA для каждой точки *p*, выбранной из набора данных, неравенства $d_{1D}(q,u|p) = |d(q,p) - d(u,p)| \le d(q,u)$ и $D_{1D}(q,u|p) = |d(q,p) + d(p,u)| \ge d(q,u)$ применяются только один раз для обновления связанной нижней границы $d_{1D}(q,u_i)$. В алгоритме PAESA, когда точка поворота будет добавлена в набор таких точек, могут быть вычислены различные нижние границы для $d(q,u_i)$.

Таким образом, с выбранными *M* точками поворота нижние границы $d_{2D}(q,u_i)$ или $d_{3D}(q,u_i)$ для $d(q,u_i)$ выбираются как максимум от *N*(*N*+1)/2 приближений. Следовательно, нижняя граница, определенная с помощью алгоритма PAESA, как правило, будет гораздо более строгой, чем у AESA, а значит, может быть сохранено больше расчетов расстояний.

Дополнительная нижняя граница от расстояния между двумя объектами определяется тогда, когда известны их расстояния до двух опорных точек.

Если доступно положительное полуопределенное метрическое двухмерное пространство *N* = (*D*, *d*), где *o*, *p*, *u* - неидентичные точки, принадлежащие *D*, *o* = *p*, то для любых *q*, принадлежащих *D*, справедливо неравенство $d_{2D}(p,u|o.p) = \left|\bar{d}(o,p;q) - \bar{d}(o,p;u)\right| \le d(q,u)$.

При использовании метрических пространств общего вида можно расширять их мерность. Принципиально алгоритм не меняется, однако, для вычислений расстояний изменяется подсчет расстояния между точками. Для трехмерного пространства уравнения для границы аналогичные, однако расстояние измеряется в соответствии с формулой вычисления расстояний между точками в трехмерном пространстве:

$$d(q,u) \geq d_{3D}(p,u|o,p) = \sqrt{d^2(u,q|o,p) + (d(q,q') - d(u,u'))^2},$$
$$d(q,u) \leq D_{3D}(p,u|o,p) = \sqrt{d^2(u,q|o,p) + (d(q,q') + d(u,u'))^2}.$$

Исходя из этого размерность не имеет значения, границы будут строиться на основе существующих с добавлением уровня мерности.

iAESA. Метод под названием iAESA (Improved AESA) [4] использует отличный от AESA способ выбора очередной опорной точки на шаге 2. Для этого вводятся следующие понятия:

- *Отношение следования* $\leq_u$: $y,z \in X$: $y \leq_u z \Leftrightarrow d(y,u) \leq d(z,u)$.
- *Перестановка*: $\Pi_u = p_1, p_2, \ldots, p_{|p|}$, где $p_i \leq p_{i+1}$. Иначе говоря, перестановка элемента u – это множество элементов p_i, расположенных в порядке возрастания расстояния до u.

Идея, лежащая в основе iAESA, основывается на следующем утверждении: если два объекта расположены близко, то их перестановки отличаются ненамного. Если, к примеру, объект u расположен близко к q, то Π_u будет мало отличаться от Π_q.

Для сравнения перестановок используется критерий Спирмана:

$$F(u)=F(\Pi_u, \Pi_q) = \sum_{i=1}^{|P|} | \Pi_u^{-1}(p_i) - \Pi_q^{-1}(p_i) |,$$

где $\Pi_u^{-1}(p_i)$ – индекс элемента p_i в перестановке Π_u.

Основные отличия iAESA от AESA:

- На шаге 2 выбирается опорная точка, имеющая наименьшее значение $F(u)$: $P \leftarrow \arg\min_{u \in U-F-P} F(u)$.
- На шаге 3 выбранная опорная точка p вставляется в Π_q.
- На шаге 5 опорная точка p вставляется в перестановки Π_u элементов u, оставшихся после отсеивания. Затем для них обновляются значения $F(u)$.

Временная сложность поиска iAESA в худшем случае оценивается как $O(|P|^2 \cdot n)$, поскольку требуется $O(|P|)$ времени на обновление Π_u и $F(u)$.

TLAESA. Tree-LAESA [7] – развитие метода LAESA. Основное отличие TLAESA от LAESA – сублинейная временная сложность поиска. Она достигается за счет использования бинарного дерева T, хранящего все объекты, среди которых производится поиск. Как и в LAESA, используется матрица расстояний M между опорными точками и остальными объектами. Во время выполнения поиска матрица M используется для исключения тех ветвей, объекты в которых находятся заведомо далеко от объекта запроса. Алгоритм состоит из двух стадий.

Стадия предварительной обработки:

- Формируется подмножество опорных точек $B = \{b_1, b_2, \ldots, b_m\}$. Первая опорная точка выбирается случайным образом, а в качестве остальных выбираются точки, находящиеся на максимальном расстоянии друг от друга: $b_i = argmax_{p \in (U-B_i)} \sum_{k=1}^{i-1} d(p, b_k)$, где $B_i = \{b_1, \ldots, b_{i-1}\}$.
- Формируется бинарное дерево T. Каждый узел $t \in T$ соответствует подмножеству $S_t \subset U$. Если t – листовой узел, то S_t содержит один объект. В противном случае $S_t = S_{t_l} \cup S_{t_r}$, где t_l и t_r – левый и правый потомки t. В каждом узле хранится опорная точка $m_t \in S_t$ и радиус $r_t = \max_{p \in S_t} d(p, m_t)$. Для листового узла $r_t = 0$.

Для корневого узла ρ подмножество $S_p = U$, а m_p выбирается случайным образом. Для правого потомка t_r каждого узла t опорная точка совпадает с опорной точкой самого узла: $m_{t_r} = m_t$. В качестве опорной точки левого узла берется объект, находящийся на максимальном расстоянии от опорной точки узла: $m_{t_l} = \operatorname{argmax}_{p \in S_t} d(p, m_t)$.

В подмножество правого потомка S_{t_r} входят те объекты, которые находятся ближе к m_{t_r}. Остальные объекты помещаются в подмножество левого потомка

$$S_{t_l}:S_{t_r} = \{p \in S_t : d(p, m_{t_r}) < d(p, m_{t_l})\}, S_{t_l} = S_t - S_{t_r}.$$

• Формируется матрица *M*, в которой хранятся расстояния между каждой опорной точкой b_i и всеми остальными элементами *U*.

Стадия поиска:

• Формируется *D* – вектор расстояний от всех опорных точек до объекта запроса: $D[b_i] = d(q,b), i = 1 \ldots m.$ При этом находится наибольшая из нижних границ расстояний от корня дерева до объекта запроса, полученная с помощью опорных точек и неравенства треугольника: $g_p = \max(g_p, |M[b,\rho] - D[b]|)$. Также определяется ближайшая к объекту запроса опорная точка.

• Производится рекурсивный поиск в дереве. Если *t* – листовой узел и нижняя граница расстояния между ним и объектом запроса *q* меньше, чем расстояние до текущего ближайшего соседа d_{min}, то вычисляется расстояние *d*(*t*,*q*). Если оно меньше d_{min}, то *t* объявляется новым ближайшим соседом.

Если *t* – узел с потомками, то производится рекурсивный поиск в левом и правом поддеревьях. Сравниваются нижние границы расстояний от опорной точки узла до объекта запроса, и первым производится поиск в том поддереве, в котором эта граница меньше.

Оценка эффективности рассмотренных методов

В таблице приведены оценки сложности рассматриваемых в статье алгоритмов.

Пространственная сложность и временная сложность предварительной обработки AESA оценивается как $O(n^2)$, поскольку AESA хранит матрицу расстояний между всеми парами элементов *U*. Временная сложность поиска в худшем случае оценивается как $O(|P| * n)$. Экспериментально показано, что в среднем количество вычислений расстояний во время поиска является постоянным и практически не зависит от n [1; 2]. С ростом размерности данных количество вычислений расстояний увеличивается. Таким образом, временную сложность поиска AESA можно оценить как *O*(1).

Пространственная сложность и время предварительной подготовки LAESA оценивается как *O(mn)*. Временная сложность поиска оценивается как *m*+*O*(1) [8].

Для iAESA требования к памяти не отличаются от AESA. Временная сложность поиска в худшем случае оценивается как $O(|P|^2 * n)$, поскольку требуется *O*(|*P*|) времени на обновление $\prod_u$ и *F*(*u*).

Для TLAESA сложность поиска варьируется между *O*(*log n*) и *O*(*mn*).

Оценки пространственной сложности поиска PAESA и LPAESA совпадают с AESA и LAESA соответственно.

Таблица

Оценка сложности алгоритмов

Алгоритм	Пространственная сложность	Временная сложность выполнения поискового запроса
AESA	$O(n^2)$	$O(1)$
LAESA	$O(m \cdot n)$	$m + O(1)$
iAESA	$O(n^2)$	$O(\|P\|^2 \cdot n)$
TLAESA	$O(m \cdot n)$	$O(n^2)$
PAESA	$O(n^2)$	$O(1)$
LPAESA	$O(m \cdot n)$	$m + O(1)$

В данной работе была изучена ситуация, когда LAESA быстрее и дешевле, чем AESA. В предыдущих работах было выявлено, что AESA - наилучший алгоритм, потому что он вычисляет меньше расстояний. Тем не менее стоимость функции несхожести очень важна для конечного результата.

Также был предложен метод Reduced Overhead AESA (ROAESA), отличающийся от AESA и LAESA уменьшенным потреблением памяти.

СПИСОК ЛИТЕРАТУРЫ

1. Гулаков, В.К. Анализ методов поиска в метрических пространствах на основе сферического разбиения множества объектов / В.К. Гулаков, В.Н. Матюшин // Информационные технологии, радиоэлектроника, телекоммуникации (IIRT-2014): IV междунар. заоч. науч.-техн. конф. - Тольятти, 2014. - С. 98-102.
2. Гулаков, В.К. Оценка эффективности использования метрических деревьев в приближенном поиске на основе обобщенного гиперплоскостного разбиения множества объектов / В.К. Гулаков, В.Н. Матюшин // Вестник Брянского государственного технического университета. - 2013. - № 4. - С. 106-112.
3. Chavez, E. Proximity searching in metric spaces / E. Chávez, G. Navarro, R. Baeza-Yates, J.L. Marroquin // ACM Computing Surveys (CSUR). - 2001. - Vol. 33. - Is. 3. - P. 273-321.
4. Figueroa, K. On the Least Cost for Proximity Searching in Metric Spaces / K. Figueroa, E. Chávez, G. Navarro, R. Paredes // 5th International Workshop, WEA 2006. - Cala Galdana, Menorca, Spain, 2006. - P. 279-290.
5. Matthew, A.S. Aspects of Metric Spaces in Computation / A.S. Matthew. - Waterloo, Ontario, Canada, 2008. - P. 258.
6. Samet, H. Foundations of Multidimensional and Metric Data Structures / H. Samet. - Imprint: Morgan Kaufmann, 2006. - 1024 p.
7. Tokoro, K. Improvements of TLAESA Nearest Neighbor Search Algorithm and Extension to Approximation Search / K. Tokoro, K. Yamaguchi, S. Masuda // In Proc. Twenty-Ninth Australasian Computer Science Conference (ACSC 2006). - Australian Computer Society, Inc. Darlinghurst, Australia, 2006. - P. 77-83.
8. Vidal, E. An algorithm for finding nearest neighbors in (approximately) constant average time / E. Vidal // Pattern Recognition Letters. - 1986. - Vol. 4 (3). - P. 145-157.
9. Vidal, E. A new version of the nearest-neighbor approximating and eliminating search algorithm (AESA) with linear preprocessing-time and memory requirements / E. Vidal, L. Mico, J. Oncina // Pattern Recognition Letters. - 1994. - Vol. 15 (1).

1. Gulakov, V.K. Analysis of methods for search in metric areas based on spherical division of objects multitude / V.K. Gulakov, V.N. Matyushin // *Information Technologies, Radio-electronics, Telecommunications (IIRT-2014): The IV-th Inter. Remote Scientific.-Tech. Conf.* – Togliatti, 2014. – pp. 98-102.
2. Gulakov, V.K. Effectiveness assessment of metric trees use in approximate search based on generalized hyper-flat division of objects multitude / V.K. Gulakov, V.N. Matyushin // *Bulletin of Bryansk State Technical University.* – 2013. – No.4. – pp. 106-112.
3. Chavez, E. Proximity searching in metric spaces / E. Chávez, G. Navarro, R. Baeza-Yates, J.L. Marroquin // ACM Computing Surveys (CSUR). - 2001. - Vol. 33. - Is. 3. - P. 273-321.
4. Figueroa, K. On the Least Cost for Proximity Searching in Metric Spaces / K. Figueroa, E. Chávez, G. Navarro, R. Paredes // 5th International Workshop, WEA 2006. - Cala Galdana, Menorca, Spain, 2006. - P. 279-290.
5. Matthew, A.S. Aspects of Metric Spaces in Computation / A.S. Matthew. - Waterloo, Ontario, Canada, 2008. - P. 258.
6. Samet, H. Foundations of Multidimensional and Metric Data Structures / H. Samet. - Imprint: Morgan Kaufmann, 2006. - 1024 p.
7. Tokoro, K. Improvements of TLAESA Nearest Neighbor Search Algorithm and Extension to Approximation Search / K. Tokoro, K. Yamaguchi, S. Masuda // In Proc. Twenty-Ninth Australasian Computer Science Conference (ACSC 2006). - Australian Computer Society, Inc. Darlinghurst, Australia, 2006. - P. 77-83.
8. Vidal, E. An algorithm for finding nearest neighbors in (approximately) constant average time / E. Vidal // Pattern Recognition Letters. - 1986. - Vol. 4 (3). - P. 145-157.
9. Vidal, E. A new version of the nearest-neighbor approximating and eliminating search algorithm (AESA) with linear preprocessing-time and memory requirements / E. Vidal, L. Mico, J. Oncina // Pattern Recognition Letters. - 1994. - Vol. 15 (1).

Статья поступила в редакцию 12.07.18.
Рецензент: к.т.н., доцент Брянского государственного технического университета
Подвесовский А.Г.
Статья принята к публикации 10.10.18.

Сведения об авторах:

Гулаков Василий Константинович, к.т.н., профессор кафедры «Информатика и программное обеспечение» Брянского государственного технического университета, e-mail: gvk10@yandex.ru.
Гулаков Константин Васильевич, к.т.н., доцент кафедры «Информатика и программное обеспечение» Брянского государственного технического университета, e-mail: gulakov32@yandex.ru.
Солдатенков Алексей Владимирович, аспирант кафедры «Информатика и программное обеспечение» Брянского государственного технического университета, e-mail: avsoldatenkov@yandex.ru.

Gulakov Vasily Konstantinovich, Can. Sc. Tech., Prof. of the Dep. “Informatics and Software”, Bryansk State Technical University, e-mail: gvk10@yandex.ru.
Gulakov Konstantin Vasilievich, Can. Sc. Tech., Assistant Prof. of the Dep. “Informatics and Software”, Bryansk State Technical University, e-mail: gulakov32@yandex.ru.
Soldatenkov Alexey Vladimirovich, Post graduate student of the Dep. “Informatics and Software”, Bryansk State Technical University, e-mail: avsoldatenkov@yandex.ru.