такие, что $(x)_\infty = kP_\infty$, $(y)_\infty = mP_\infty$ и $(z)_\infty = lP_\infty$, где $m > l > k$. Пусть $D = \sum\limits_{P \in J} P$, где $J \subseteq \mathbb{P}_C^{(1)} \setminus P_\infty$. Если $n > \max\{3r, 2\,(l + (k-1)/\mu)\,, 2\,(m + (k-1)/\eta)\}$, причём $\mu = \min\{k-1, \zeta : z^\zeta \in \mathcal{L}(rP_\infty)\}$, $\eta = \min\{k-1, \xi : y^\xi \in \mathcal{L}(rP_\infty)\}$, то

$$\mathrm{Aut}(C_\mathcal{L}(D, rP_\infty)) \cong \mathrm{Aut}_{D,rP_\infty}(C/\mathbb{F}_q).$$

ЛИТЕРАТУРА

1. *Stichtenoth H.* On automorphisms of geometric Goppa codes // J. Algebra. 1990. V. 130. Iss. 1. P. 113–121.
2. *Xing C.* Automorphism group of elliptic codes // Communication in Algebra. 1995. No. 23(11). P. 4061–4072.
3. *Xing C.* On automorphism groups of the Hermitian codes // IEEE Trans. Inform. Theory. 1995. No. 41(6). P. 1629–1635.
4. *Lauter K.* Geometric methods for improving the upper bounds on the number of rational points on algebraic curves over finite fields. With an appendix by J.-P. Serre // Algebraic Geometry. 2001. No. 10(1). P. 19–36.
5. *Milne J. S.* Abelian Varieties. 2008. `www.jmilne.org/math/`
6. *Alekseenko E. and Zaytsew A.* Explicit equations of optimal curves of genus 3 over certain fields with three parametrs // Contemporary Math. 2015. No. 637. P. 245–256.
7. *Alekseenko E., Aleshnikov S., Markin N., and Zaytsew A.* Optimal curves over finite fields with discriminant -19 // Finite Fields and Their Applications. 2011. No. 17(4). P. 350–358.
8. *Stichtenoth H.* Algebraic Function Fields and Codes. Springer, 2009.

УДК 519.7 DOI 10.17223/2226308X/11/37

ПРИМЕНЕНИЕ КОНЕЧНЫХ АВТОМАТОВ ДЛЯ НЕЧЁТКОГО БИНАРНОГО ПОИСКА

И. В. Панкратов

Рассматривается задача нечёткого поиска булевых векторов в потоке данных. Под нечётким вхождением искомого вектора понимается вхождение вектора, близкого к искомому в смысле расстояния Хемминга. Предлагается метод построения конечного автомата для решения данной задачи по заданному набору искомых шаблонов в виде булевых векторов (возможно, частично определённых) и допустимого отклонения для каждого шаблона. Возможно построение автомата, принимающего на вход отдельные биты данных, и автомата, принимающего сразу группы битов. Приводятся оценки размеров таблиц переходов и выходов автомата. Представлены экспериментальные данные производительности поисковых автоматов, принимающих на вход отдельные биты данных, четвёрки битов и восьмёрки битов, а также производительность классического подхода к задаче нечёткого поиска, основанного на регистре сдвига.

Ключевые слова: *поисковые автоматы, нечёткий поиск, бинарный поиск, синхропосылка, поиск подстроки, КМП-поиск, алгоритм Ахо — Корасик.*

1. Задача бинарного поиска в потоке данных и классический подход к ней

Рассматриваемую задачу можно сформулировать так. Имеется двоичная последовательность (*поток данных*) и набор из n булевых векторов (слов) различной длины, далее называемых *шаблонами*. Необходимо найти в последовательности вхождения

всех шаблонов. Эту задачу можно назвать *чётким поиском*. В случае, когда нас интересуют вхождения не только самих шаблонов, но и векторов, похожих на них в смысле расстояния Хемминга, можно говорить о *нечётком поиске*. Тогда помимо самих шаблонов необходимо задать для каждого из них *предельное количество ошибок*, то есть максимально допустимое расстояние Хемминга до вхождений.

Рассмотрим стандартный подход к этой задаче. Он основан на *регистре сдвига*, в который последовательно подаются биты из потока. После подачи каждого бита содержимое регистра сравнивается со всеми шаблонами. Если содержимое регистра и шаблонов помещается в машинное слово на целевом оборудовании, требуется ровно n целочисленных сравнений при чётком поиске и n вычислений расстояния Хемминга — при нечётком поиске. Если длина регистра превышает длину шаблона, нужно обнулять старшие биты регистра перед сравнением. Дополнительные неудобства возникают, когда искомые шаблоны имеют различные длины. Если же регистр не помещается в машинное слово, эта схема ещё усложняется.

2. Применение поисковых автоматов для чёткого поиска

Более эффективным способом поиска бинарных последовательностей является использование *поискового конечного автомата* [1, 2].

Сначала рассмотрим *битовый автомат*, который получает на вход данные побитово и выдаёт на выход сведения о найденных шаблонах. Для использования автомата смысл состояний не важен, поэтому можно считать, что состояния являются абстрактными элементами множества состояний автомата. На этапе построения автомата необходимо для каждого состояния знать, какие префиксы искомых шаблонов в этом состоянии уже найдены. Эта информация сохраняется в *таблице состояний*, которая удаляется после построения автомата. Под *условной подачей* бита на вход автомата будем понимать вычисление того, как изменится набор найденных префиксов и будут ли найдены какие-либо шаблоны после обработки поданного бита.

Поисковый автомат строится индуктивно — изначально задаётся нулевое состояние, в котором не найдено ничего, затем в этом состоянии условно подаётся нулевое значение бита и получается новое состояние. Аналогично получается новое состояние при условной подаче единичного бита. Каждое новое состояние ищется в таблице состояний, и если не находится, то добавляется в неё. Параллельно строятся таблицы переходов и выходов. Алгоритм продолжает работу, пока не будут обработаны все состояния в таблице состояний. Для компактного хранения таблицы выходов все возможные выходные символы сохраняются в *таблице выходных сигналов*, а в таблицу выходов записываются их индексы.

В [2] показана связь поискового автомата для чёткого поиска с алгоритмами КМП-поиска [3] и Ахо — Корасик [4]. Для алгоритма Ахо — Корасик обнаружена полная аналогия с поисковыми автоматами, для КМП-поиска — аналогия с автоматом для поиска одного шаблона.

Далее вводится *байтовый автомат*, который получает на вход сразу группы по q битов. Эффективность такого автомата выше в q раз, однако объём требуемой памяти растёт экспоненциально, поскольку в таблице будет не два столбца, а 2^q. Кроме этого, размер таблицы выходных сигналов увеличивается, как минимум, в q раз. Множество состояний байтового автомата совпадает с множеством состояний битового автомата. Строится байтовый автомат на основе битового условной подачей каждой возможной группы битов в каждом состоянии.

Более подробно алгоритмы построения поисковых автоматов описаны в [2].

Удобно использовать байтовые автоматы с группами битов того же размера, что и минимальная адресуемая ячейка памяти целевого оборудования, как правило, это восемь битов. В случае, когда необходимо уменьшить объём требуемой памяти, можно воспользоваться промежуточным вариантом — использовать группы по 4 бита. Это менее удобно, зато объём требуемой памяти уменьшается в 16 раз.

3. Применение поисковых автоматов для нечёткого поиска

Далее рассмотрим, как можно реализовать нечёткий поиск при помощи поисковых автоматов. Расстояние Хемминга между шаблоном и найденным вектором будем называть количеством *ошибок* в найденном вхождении.

Входные алфавиты поисковых автоматов для нечёткого поиска остаются неизменными, выходные алфавиты требуется расширить таким образом, чтобы была возможность сигнализировать не только о том, в какой позиции какой шаблон был найден, но и о том, какое количество ошибок он содержит. Состояния автомата здесь также должны содержать информацию о количестве ошибок. Алгоритмы построения таблиц автоматов для нечёткого поиска аналогичны алгоритмам построения обычных поисковых автоматов.

Верхняя оценка количества состояний поискового автомата для чёткого поиска приводится в [2], для нечёткого поиска верхняя оценка получилась следующей:

$$\sum_{l=0}^{k_{\max}-1} \min\left(2^l, \sum_{i:k_i>l} \sum_{e=0}^{e_i} \binom{l}{e}\right),$$

где $k_{\max}$ — максимальная длина шаблона; k_i и e_i — соответственно длина и число допустимых ошибок i-го шаблона; $\binom{l}{e}$ — число сочетаний из l по e. Отметим, что формула верна только для полностью определённых шаблонов. Сравнение оценки с экспериментальными данными показало, что для нечёткого поиска её значение сильно завышено, а для чёткого — достаточно точно.

4. Минимизация таблиц поисковых автоматов

Алгоритмы построения автоматов реализованы, проведено исследование количества состояний автоматов для поиска 48-битовых шаблонов, выбранных случайным образом. Изменялось количество искомых шаблонов — от 1 до 6 — и количество допустимых ошибок — от 0 до 5. Отметим, что при нулевом количестве допустимых ошибок фактически строится автомат для чёткого поиска. В табл. 1 представлены полученные результаты. Построить поисковый автомат для поиска трёх и более искомых шаблонов с пятью допустимыми ошибками в 32-битном адресном пространстве не удалось. Поскольку количество состояний существенным образом зависит от вида шаблонов, значения в табл. 1 являются ориентировочными.

По табл. 1 можно заметить, что количество состояний поисковых автоматов для нечёткого поиска существенно превышает количество состояний поискового автомата для чёткого поиска тех же шаблонов и очень быстро увеличивается с ростом допустимого количества ошибок. Кроме того, замечено следующее: в автоматах для чёткого поиска при увеличении количества шаблонов количество состояний растёт медленнее, чем суммарная длина шаблонов, то есть один автомат для поиска двух шаблонов требует меньше памяти, чем два отдельных автомата, ищущих по одному шаблону. С поисковыми автоматами для нечёткого поиска ситуация обратная.

Таблица 1

Количество состояний поисковых автоматов для нечёткого поиска

Допустимое количество ошибок	Количество искомых шаблонов					
	1	2	3	4	5	6
0	48	93	140	183	222	263
1	355	896	1479	2116	2657	3265
2	2335	6566	12 370	18 242	24 097	30 334
3	14 891	44 502	91 703	140 209	192 659	246 747
4	88 979	278 629	621 499	991 030	1 410 670	1 828 102
5	501 340	1 653 861	X	X	X	X

Есть два способа уменьшить количество состояний автомата. Для описания первого метода определим несущественные состояния поискового автомата по аналогии с цепями Маркова.

Несущественным назовём такое состояние i, из которого автомат может перейти в некоторое другое состояние j, такое, что из него автомат никогда не сможет вернуться в состояние i. Содержательно это означает, что автомат может перейти в такой режим работы, в котором никогда не сможет попасть ни в какое несущественное состояние. Для поискового автомата доказано, что он гарантированно войдёт в такой режим после подачи на вход $k_{\max}$ битов данных, где $k_{\max}$ — максимальная длина шаблона.

Первый метод уменьшения количества состояний поисковых автоматов заключается в поиске и устранении несущественных состояний. В этом случае автомат будет начинать работу не из нулевого состояния, а из некоторого другого, как правило, первого по порядку существенного состояния, поэтому в начале работы автомата возможны «ложные» срабатывания в силу того, что все состояния, кроме нулевого, предполагают, что уже найдены какие-то префиксы шаблонов. Для отсечения таких «ложных» срабатываний достаточно контролировать длину найденных шаблонов и сравнивать её с длиной данных, поданных на вход автомата.

Другой метод заключается в построении минимальной формы автомата, то есть устранении эквивалентных состояний.

Первый способ был применён ко всем автоматам из табл. 1. Результаты приведены в табл. 2.

Можно видеть, что удаление несущественных состояний позволило уменьшить общее количество состояний автомата не более чем на 2,857 %.

5. Экспериментальные данные

Проведено тестирование производительности различных поисковых автоматов и обычного метода поиска, основанного на регистре сдвига. Для измерения скорости работы использовался компьютер на базе процессора Intel Core 2 Duo E8400 с тактовой частотой 2,66 ГГц под управлением 64-битной операционной системы Windows 7 и компилятор MinGW версии 5.3.0. Измерение времени производилось с помощью функции Windows API GetThreadTimes(), вычисляющей время работы процессора над указанной задачей без учёта времени, потраченного на другие задачи. В качестве источника данных использован стандартный линейный конгруэнтный генератор, замеры скорости работы проводились на отрезках по 100 МБ. Здесь и далее под 1 МБ понимается 2^{30} байтов.

Бинарный поиск, основанный на регистре сдвига, показал скорость работы от 0,48 до 4,0 МБ/с. Замечено, что производительность поиска, основанного на регистре сдви-

Таблица 2

Количество несущественных состояний поисковых автоматов

Допустимое количество ошибок	Количество искомых шаблонов					
	1	2	3	4	5	6
0	0 / 48 (0,00 %)	0 / 93 (0,00 %)	4 / 140 (2,86 %)	4 / 183 (2,19 %)	4 / 222 (1,80 %)	4 / 263 (1,52 %)
1	3 / 355 (0,85 %)	7 / 896 (0,78 %)	17 / 1479 (1,15 %)	37 / 2116 (1,75 %)	40 / 2657 (1,51 %)	44 / 3265 (1,35 %)
2	23 / 2335 (0,99 %)	43 / 6566 (0,65 %)	98 / 12 370 (0,79 %)	222 / 18 242 (1,22 %)	270 / 24 097 (1,12 %)	323 / 30 334 (1,06 %)
3	229 / 14 891 (1,54 %)	402 / 44 502 (0,90 %)	772 / 91 703 (0,84 %)	1396 / 140 209 (1,00 %)	1820 / 192 659 (0,94 %)	2319 / 246 747 (0,94 %)
4	1453 / 88 979 (1,63 %)	2473 / 278 629 (0,89 %)	4837 / 621 499 (0,78 %)	9275 / 991 030 (0,94 %)	12 039 / 1 410 670 (0,85 %)	16 397 / 1 828 102 (0,90 %)
5	7584 / 501 340 (1,51 %)	13 825 / 1 653 861 (0,84 %)	X	X	X	X

га, не зависит от вида искомых шаблонов и одинакова как для полностью определённых, так и для частично определённых шаблонов одинаковой длины. Производительность уменьшается при увеличении количества шаблонов и их длины.

Производительность битовых поисковых автоматов, напротив, не зависит напрямую ни от количества искомых шаблонов, ни от их вида и длины, но существенно зависит от количества состояний поискового автомата. Под *объёмом таблиц* будем понимать объём памяти, необходимой для хранения всех таблиц автомата. При объёме таблиц, не превышающем 1 МБ, производительность битовых поисковых автоматов достигает 32 МБ/с. С увеличением объёма таблиц производительность падает до 12 МБ/с. В некоторых случаях наблюдается падение скорости обработки вплоть до 4,66 МБ/с, но оно связано с большим количеством найденных шаблонов, обработка которых требует дополнительного времени. Во всех случаях скорость работы битового поискового автомата многократно выше скорости работы регистрового поиска и в среднем составляет 24,13 МБ/с.

Помимо битовых, испытывались полубайтовые и байтовые автоматы. Полубайтовые автоматы показали производительность от 5,77 до 109 МБ/с, работая быстрее битовых автоматов как минимум в 2 раза. Байтовые автоматы достигают производительности 286 МБ/с, однако из-за большого объёма таблиц их производительность довольно быстро снижается с ростом количества состояний, в связи с чем примерно в половине случаев байтовые автоматы уступают по производительности полубайтовым, а в ряде случаев — и битовым автоматам.

На рис. 1 показаны графики производительности регистрового поиска, битовых, полубайтовых и байтовых автоматов. График производительности регистрового поиска обозначен пунктирной линией, битового автомата — двойной линией, полубайтового — точками, байтового — сплошной серой линией. В правой части графика линия производительности байтового автомата прерывается в связи с тем, что для соответствующего количества состояний построить байтовый автомат вообще не удалось из-за нехватки памяти.

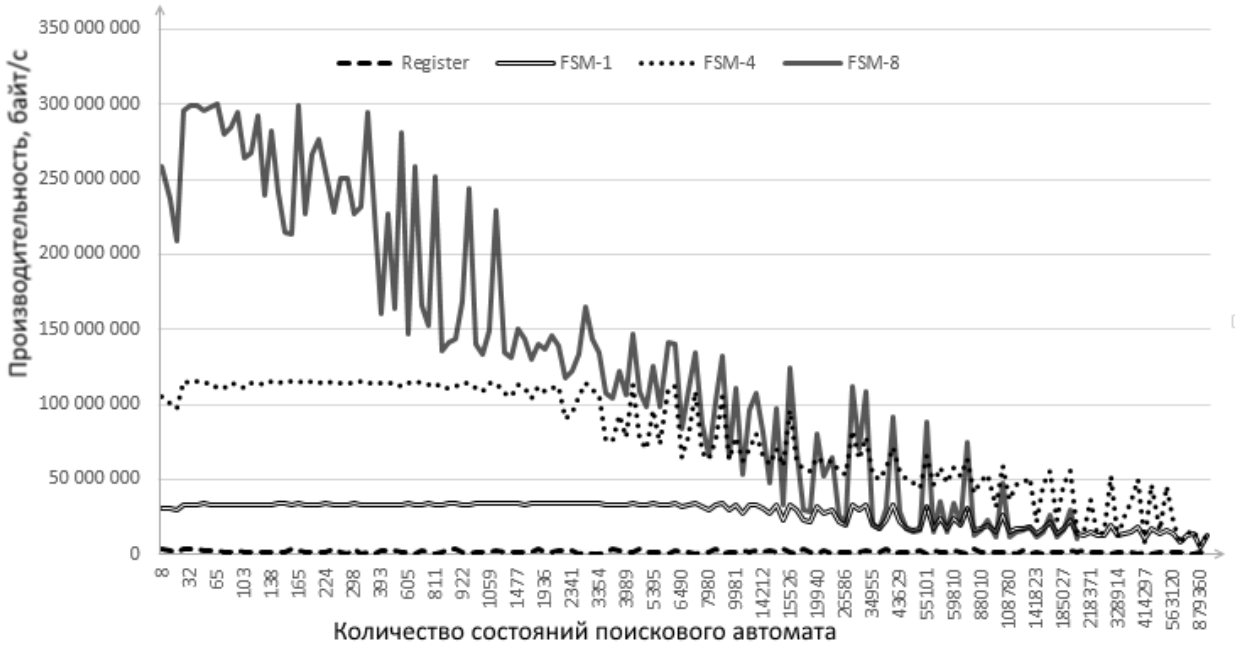

Рис. 1. Производительность поисковых систем

По графикам видно, что битовые автоматы всегда работают быстрее регистрового поиска, полубайтовые автоматы всегда быстрее битовых, а байтовые достигают очень высокой производительности при небольших объёмах таблиц, однако с ростом числа состояний их производительность падает настолько быстро, что они начинают проигрывать полубайтовым, а затем и битовым автоматам.

Можно заключить, что регистровый поиск следует использовать только в том случае, когда доступное количество памяти недостаточно даже для хранения таблиц битового автомата, битовый автомат — только если недостаточно памяти для полубайтового автомата. Выбор между байтовым и полубайтовым автоматами следует делать, исходя из измерений производительности обоих автоматов на целевом оборудовании.

ЛИТЕРАТУРА

1. *Агибалов Г. П., Оранов А. М.* Лекции по теории конечных автоматов. Томск: Изд-во Том. ун-та, 1984. 185 с.
2. *Панкратов И. В.* Одновременный поиск нескольких двоичных шаблонов в потоке с помощью конечного автомата // Прикладная дискретная математика. 2014. № 2. С. 119–125.
3. *Knuth D. E., Morris J. H. Jr., and Pratt V. R.* Fast pattern matching in strings // SIAM J. Comput. 1977. No. 6(2). P. 323–350.
4. *Aho A. V. and Corasick M. J.* Efficient string matching: An aid to bibliographic search // Commun. ACM. 1975. No. 18(6). P. 333–340.

УДК 519.7 DOI 10.17223/2226308X/11/38

АЛГОРИТМ ОПТИМАЛЬНОЙ МАРШРУТИЗАЦИИ В МУЛЬТИСЕРВИСНЫХ ТЕЛЕКОММУНИКАЦИОННЫХ СЕТЯХ

А. А. Солдатенко

Рассматривается NP-трудная задача поиска ресурсоограниченного кратчайшего пути (RCSP) в графе $G = (V, E)$. Задача RCSP является расширением извест-