§5 ПРОГРАММНЫЕ КОПЛЕКСЫ КОМПЬЮТЕРНОЙ РЕАЛИЗАЦИИ ЧИСЛЕННЫХ МЕТОДОВ И МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ НА ИХ ОСНОВЕ

Е.Н. Потехин, А.Н. Леухин

МЕТОДЫ ОПТИМИЗАЦИИ ЗАДАЧИ ПОЛНОГО ПОИСКА БИНАРНЫХ АПЕРИОДИЧЕСКИХ ОПТИМАЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Аннотация. *Ставится проблема поиска бинарных оптимальных апериодических последовательностей для задач обнаружения целей. Приводится алгоритм полного поиска «brunch and bound». Вводится понятие эквивалентных преобразований, описывается их применение для сокращения вычислительной сложности алгоритма. Описываются методы оптимизации алгоритма и сокращения его вычислительной сложности благодаря использованию современных процессорных команд, вычислительных графических кластеров, методов распараллеливания алгоритма, пакетных режимов поиска.*

Ключевые слова: *бинарные последовательности, апериодические последовательности, оптимальные последовательности, импульсная автокорреляционная функция, методы оптимизации, brunch and bound, NVidia CUDA, эквивалентные последовательности, коды Баркера.*

При разработке радиолокационных и телекоммуникационных систем большое влияние уделяется не только качеству и возможностям аппаратной платформы, но и определенным свойствам сигналов, который служат основным носителем информативных признаков. В частности, в системах обнаружения объектов методом радиолокации одной из важнейших характеристик является апериодическая (импульсная) автокорреляционная функция (ИАКФ) сигнала. Чем меньше ее уровень, тем более устойчивым к шумам является излучаемый сигнал. В идеале, самыми лучшими для этих целей сигналами являются фазоманипулированные последовательности Баркера [1]**.** Однако, бинарные последовательности Баркера заканчиваются на длине $N = 13$, а многофазные последовательности Баркера предъявляют слишком высокие требования к точности генерации иррациональных фаз, минимальная погрешность в которых ведет к значительному повышению уровней ИАКФ последовательности, что может вести к ложному обнаружению целей. К сожалению, на сегодняшний день не существует аналитического решения задачи построения идеальных с точки зрения ИАКФ последовательностей. Для удешевления аппаратной составляющей имеет смысл максимально сократить количество фаз сигнала до $N(\varphi) = 2$, т.е. использовать бинарный сигнал, где фазы максимально разнесены друг от друга и имеют алфавит из 2 фаз: $\varphi_1 = 0, \varphi_2 = \pi$. Такие последовательности хоть и не имеют идеальной ИАКФ, однако обладают оптимальными (минимально возможными) значениями ИАКФ. С учетом того, что при излучении таких сигналов не происходит искажение фаз, точность обнаружения целей может быть сопоставима с системами, где используются многофазные последовательности Баркера. В работах [2-7] приводятся методы построения бинарных последовательностей с хорошим уровнем ИАКФ, однако

DOI: 10.7256/2305-6061.2013.2.7600

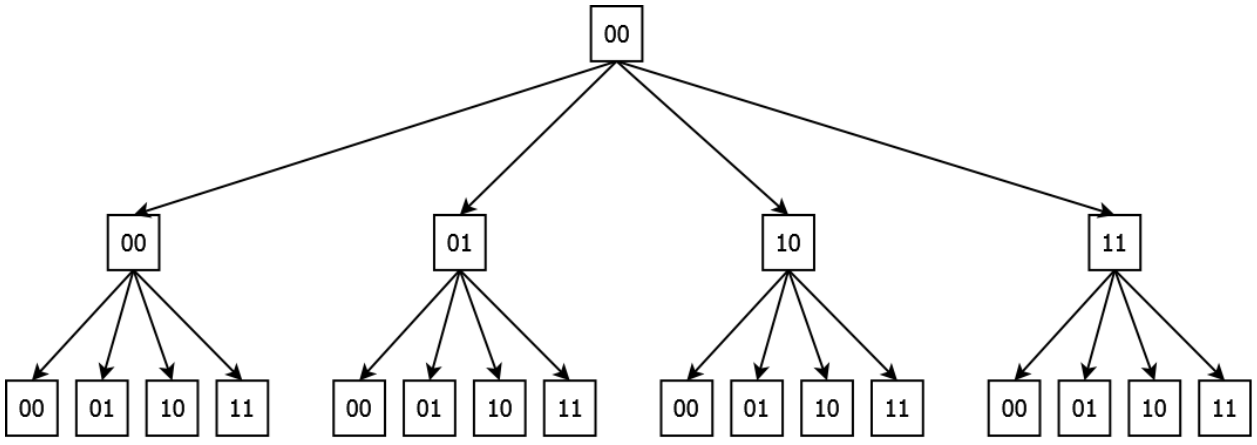
эти уровни далеки от оптимальных. Тем не менее, при сегодняшнем уровне развития вычислительной техники имеется возможность получить полный класс последовательностей для выбранной длины N. Целью данной статьи является описание метода полного поиска бинарных оптимальных последовательностей, а также его модификация с учетом современной элементной базы вычислительной техники, которые позволили осуществить полный поиск таких последовательностей до длины $N = 70$.

Алгоритм полного поиска «brunch and bound»

Для программной реализации в качестве базового используется алгоритм, описанный в работе [8] исследователями Грегом Коксонон и Джоном Руссо. Данный метод носит название brunch and bound, который заключается в последовательном "наращивании" последовательности, начиная с ее краев. Бинарная последовательность представляется в виде двух частей кода одной длины $N/2$ — левого и правого полукодов. На начальном этапе поиска $N = 2$, т.е. устанавливаются крайний левый бит левого полукода x_0 и крайний правый бит правого полукода x_{N-1}. Они образуют своеобразную пару бит $x_0 x_{N-1}$. Эта пара может принимать четыре возможных варианта значений: «00», «01», «10» и «11». На первом этапе последовательность выглядит как: $a = 0, x_1, x_2, \ldots x_{N-3} x_{N-2}, 0$. Затем находится первый боковой лепесток импульсной автокорреляционной функции последовательности. Если $|r_l| \le k$, где k — заданный максимально возможный уровень бокового лепестка, тогда можно продолжить построение последовательности и добавить второй слева бит левого полукода x_1 и второй справа бит правого полугода x_{N-2}. Тогда последовательность примет вид: $a = 00, x_2, \ldots x_{N-3} 00$. Если $|r_l| > k$, тогда необходимо паре $x_0 y_0$ присвоить следующее возможное значение бит: «01». В этому случае последовательность примет вид: $a = 00, x_2, \ldots x_{N-3} 10$. Подобный алгоритм легко представить в виде графа, где вершинами являются все возможные значения пар. В таком случае поиск последовательностей осуществляется обходом графа в глубину (см. рис. 1).

Рис. 1

Граф поиска



Проверка бокового лепестка при каждом изменении значения какой-либо пары значительно сокращает время поиска. Это достигается потому что, во-первых, на каждом шаге вычисляются не все боковые лепестки последовательности, а только один, поскольку предыдущие значения уже вычислены, во-вторых, если боковой лепесток не удовлетворяет условию $|r_l| \le k$, когда мы находимся в одной из вершин графа, то не имеет смысла обходить все дочерние ветки данной вершины. Аналогичный подход применяется для исключения веток графа, которые порождают эквивалентные решения, получаемые при помощи эквивалентных преобразований.

DOI: 10.7256/2305-6061.2013.2.7600

Эквивалентные преобразования — это такие преобразования последовательности, которые не меняют ее апериодическую автокорреляционную функцию. На сегодняшний день известны три таких преобразования: инверсия, поворот и фазовый набег. Если количество преобразований равно N, то количество эквивалентных последовательностей, которое можно получить из одной последовательности равно 2^N. Рассмотрим эквивалентные преобразования для импульсных последовательностей более подробно.

1. Инверсия. Пусть $a = \{a_n\}$ — импульсная последовательность длины N, где $a_n \in \{1,-1\}$, $n = 0,1,..., N-1$. Тогда эквивалентная последовательность длины , полученная при помощи преобразования инверсии, будет равна $b = \{b_n\}$, где $b_n = -1\, x_n$, $n = 0,1,..., N-1$.
2. Поворот. Пусть $a = \{a_n\}$ — импульсная последовательность длины N, где $a_n \in \{1,-1\}$, $n = 0,1,..., N-1$. Тогда эквивалентная последовательность длины , полученная при помощи преобразования поворота, будет равна $b = \{b_n\}$, где $b_n = x_{N-1-n}$, $n = 0,1,..., N-1$.
3. Фазовый набег. Пусть $a = \{a_n\}$ — импульсная последовательность длины N, где $a_n \in \{1,-1\}$, $n = 0,1,..., N-1$. Тогда эквивалентная последовательность длины N, полученная при помощи преобразования фазового набега, будет равна $b = \{b_n\}$, где $b_n = x_n + i\,\varphi$, $n = 0,1,..., N-1$, φ — угол набега фазы. Но поскольку для бинарных последовательностей $\varphi = 180^\circ$, то $b_n = (-1)^n x_n$.

Например, пусть на начальном этапе поиска четной последовательности крайние биты последовательности равны $x_0 = 0$, $x_{N-1} = 0$. Тогда исходная последовательность примет вид: $a0 = 0, x_1, ..., x_{N-2}, 0$. Построим эквивалентные решения для исходной последовательности. Преобразование инверсии даст последовательность: $a1 = 1, -x_1, ..., -x_{N-2}, 1$. Преобразование реверса от каждой последовательности приведет в двум решениям: $a2 = 0, x_{N-2}, ..., x_1, 0$ и $a3 = 1, -x_{N-2}, ..., -x_1, 1$. И преобразование фазового набега породит еще 4 решения: $a4 = 0, -x_1, ..., x_{N-2}, 1$, $a5 = 1, x_1, ..., -x_{N-2}, 0$, $a6 = 0, -x_{N-2}, ..., x_1, 1$ и $a7 = 1, x_{N-2}, ..., -x_1, 0$. В итоге одна неэквивалентная последовательность порождает еще 7 эквивалентных:

$$a0 = 0, x_1, ..., x_{N-2}, 0$$
$$a1 = 1, -x_1, ..., -x_{N-2}, 1$$
$$a2 = 0, x_{N-2}, ..., x_1, 0$$
$$a3 = 1, -x_{N-2}, ..., -x_1, 1$$
$$a4 = 0, -x_1, ..., x_{N-2}, 1$$
$$a5 = 1, x_1, ..., -x_{N-2}, 0$$
$$a6 = 0, -x_{N-2}, ..., x_1, 1$$
$$a7 = 1, x_{N-2}, ..., -x_1, 0$$

Легко заметить, что начальная последовательность длины $N = 2$ и значениями $x_0 = 0$, $x_{N-1} = 0$, порождает все остальные возможные значения пар $x_0\, x_{N-2}$. Таким образом, из четырех возможных веток графа для обхода остается лишь одна. Аналогичным образом, на следующем уровне при длине последовательности $N = 4$ останется лишь 3 ветки для обхода и т.д.:

$$a0 = 0,0, x_2, ..., x_{N-3}, 0,0$$
$$a1 = 0,0, x_2, ..., x_{N-3}, 1,0$$
$$a2 = 0,1, x_2, ..., x_{N-3}, 1,0$$

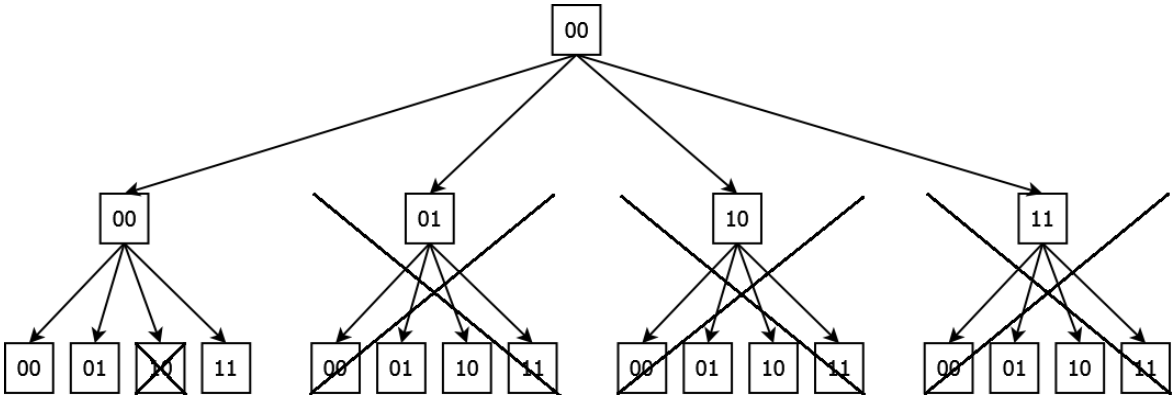
Исключения наглядно демонстрирует граф обхода (см. рис. 2):

Разбиение всего множества поиска на непересекающиеся подмножества позволяет использовать параллельные вычисления в многоядерных системах, где каждое вычислительное ядро работает с отдельным подмножеством.

DOI: 10.7256/2305-6061.2013.2.7600

Рис. 2

Исключения веток, приводящих к эквивалентным решениям



Особенности вычислительной реализации алгоритма

В программной реализации алгоритма полного поиска множества бинарных импульсных последовательностей, оптимальных по минимаксному критерию есть свои особенности и методы оптимизации, характерные для различных платформ и операционных систем.

Наиболее элегантным и простым в реализации решением задачи полного поиска, которая реализуется методом обхода графа в глубину, является рекурсивных обход. Он позволяет сделать программу поиска более компактной и понятной. Практика показала, что замена рекурсивных вычислений циклическими функциями визуально усложняет восприятие программы и практически не дает увеличения производительности.

Оптимизация вычисления боковых лепестков апериодической автокорреляционной функции

Для адаптации алгоритма вычисления бокового лепестка апериодической автокорреляционной функции к современным вычислительным способностям Коксон и Руссо в работе [8] предложили использовать для этого операцию XOR, которую также называют сложением по модулю 2. Боковой лепесток при этом будет равен разности между количеством 0 и количеством 1 в результате работы операции XOR. Например, пусть имеется апериодическая последовательность вида $a = 1, 1, 0, 1$. Данная последовательность будет иметь один главный боковой лепесток $R_0 = 4$, который всегда равен длине последовательности N. Вычислим оставшиеся корреляции. Для этого выполним операцию XOR для элементов, участвующих в вычислении уровня бокового лепестка (Таблица 1).

Таблица 1

Вычисление бокового лепестка операцией XOR

1	1	1	0		1	1	0	1			1	1	0	1					
		⊕					⊕						⊕						
			1	1	0	1			1	1	0	1		1	1	0	1		
		=					=						=						
			0					1	0					0	1	1			

DOI: 10.7256/2305-6061.2013.2.7600

Затем найдем разность между количеством 0 и 1, что будет являться значением уровня бокового лепестка с точностью до знака. Поскольку для нахождения оптимальных последовательностей достаточно знать именно модули боковых лепестков, то не имеет смысла, что из чего вычитать. Поэтому оставшиеся боковые лепестки будут равны соответственно: $R_1 = 1 - 0 = 1$, $R_2 = 1 - 1 = 0$, $R_3 = 1 - 2 = -1$.

Для максимальной производительности программа поиска максимально использует бинарную арифметику. Основная вычислительная сложность приходится именно на вычисление боковых лепестков. Операция XOR является аппаратно-реализованной во всех современных процессорах, неэффективным оставался лишь подсчет разности между количеством 0 и 1.

Благодаря развитию технологии изготовления, новые процессоры, которые поддерживают набор команд микроархитектуры Intel Core версии SSE4.2, позволяют использовать операции подсчета количества единичных бит в переменной на аппаратном уровне. Компилятор C/C++ от Microsoft использует для этого функцию __popcnt64 библиотеки intrin. GNU компилятор GCC и G++ используют функцию _mm_popcnt_u64 библиотеки smmintrin. Использование этой функции позволило увеличить скорость работы программы в 4 раза, по сравнению с программным побитовым подсчетом единичных бит при помощи циклических сдвигов и операций логического И.

Оптимизация вычисления реверсной функции

Другой неоптимальной задачей является отсеивание эквивалентных решений и веток при обходе графа, в частности, исключение реверсных кодов. Коксон и Руссо в работе [8] показали, что код и все его дочерние элементы в дереве обхода являются эквивалентными по реверсному преобразованию, если для левой и правой частей кода выполняется условие

$$l > \mathrm{REV}\,(\mathrm{r}),$$

где $\mathrm{REV}\,(\mathrm{r})$ — реверсный код кода r.

Для ускорения работы программы функция реверса была реализована не для отдельных бит, а сразу для двухбайтных переменных. Т.е. если необходимо произвести реверс 64 бит кода, то следует переставлять не каждый из 64 бит, а 4 пар по 16 бит. Все возможные реверсы 16-битных чисел хранятся в статическом массиве, содержащем 65536 различных вариаций бит. Данное условие позволило увеличить скорость поиска еще в 2 раза.

Параллельный режим вычислений

Как было отмечено выше, при обходе графа в глубину можно на каждом новом уровне можно выделять непересекающиеся поддеревья графа, которые можно обрабатывать отдельно. Например, для кодов четной длины N с уровнем боковых лепестков $\mathrm{PSL} \le 4$, в зависимости от количества начальных бит в левом и правом полукодах выделяются следующее количество непересекающихся поддеревьев (Таблица 2).

Табл. 2

Количество непересекающихся поддеревьев при различном фиксированном количестве начальных бит

Количество начальных фиксированных бит в каждом из полукодов	Количество непересекающихся поддеревьев
1	1
2	3
3	10

DOI: 10.7256/2305-6061.2013.2.7600

4	36
5	126
6	474
7	1638

Это позволяет использовать однопоточную программу для параллельных вычислений, где каждая новая версия программы, запущенная со своими начальными значениями, использует для вычислений свой процессорный поток.

Параллельный режим вычислений на графических процессорах NVidia CUDA

Подобная схема идеально подходит для реализации многопоточных вычислений с использованием графических процессоров серии NVidia CUDA. Принцип многопоточных вычислений основан на том, что каждая графическая карта содержит в себе большое количество графических процессоров, каждый из которых может выполнять свою вычислительную задачу.

Для целей полного поиска бинарных апериодических последовательностей, оптимальных по минимаксному критерию, также было разработано программное обеспечение, адаптированное для работы на графических процессорах. В результате проведенных экспериментов при использовании графических процессоров с 300 процессорами удалось получить прирост производительности в 8 раз, по сравнению с аналогичной реализацией алгоритма для процессора Intel Xeon.

CUDA SDK также имеет аппаратно-реализованную функцию __popcll() для подсчета количества единичных бит в переменной.

Пакетный режим поиска последовательностей

Как было отмечено выше, поиск последовательностей длины N осуществляется путем обхода дерева возрастающих полукодов в глубину. Так, чтобы найти последовательность длины N, необходимо «спуститься» по дереву на глубину $N/2$. При относительно больших значениях N, доход до самых крайних листьев дерева на уровне $N/2$ занимает довольно много времени. Но ведь для того, чтобы найти все оптимальные бинарные последовательности длины $N+2$, необходимо пройти то же самое дерево полукодов, что и при поиске последовательностей длины N, опустившись лишь на один уровень глубже. Так зачем тогда проходить весь путь заново, когда можно при поиске последовательностей длины N одновременно проверять предыдущие уровни для поиска последовательностей длин $N-2$, $N+2$ и т.д. В этом случае львиная часть обхода графа выполняется один раз для всех заданных для поиска длин.

Подобный алгоритм описан для последовательностей четных длин, однако, он с тем же успехом может применяться как для последовательностей нечетных длин, так и для последовательностей любых N.

Естественно, что время поиска последовательностей в пакетном режиме больше, чем поиск последовательностей самой большой длины N, однако если рассматривать общие временные затраты на поиск последовательностей всех длин, участвующих в пакетном режиме, отдельно, то очевидно, что пакетный режим также приводит к сокращению времени поиска.

Исключение невалидных веток дерева обхода

Ясно, что при поиске последовательностей с заданными корреляционными характеристиками, алгоритм осуществляет проход не до всех листьев графа, расположенных на самом нижнем уровне. Это связано с тем, что какой-либо корень, расположенный на более высоком уровне, в котором была заполнена лишь часть полукодов, был исключен из обхода из-за большого значения бокового лепестка ИАКФ. В этом случае вся дальнейшая его ветвь была отброшена.

DOI: 10.7256/2305-6061.2013.2.7600

С ростом длины N разыскиваемых последовательностей, их общий объем уменьшается, поэтому логично предположить, что количество листьев дерева обхода, до которых не дошел алгоритм, увеличивается. А поскольку при поиске последовательностей длины $N+2$ алгоритм обходит то же самое дерево поиска, кроме дополнительного нижнего уровня, что и при поиске последовательностей длины N, то множество необойденных корней дерева совпадают. Это позволяет заранее исключать из будущего поиска те узлы дерева, в которых алгоритм не дошел ни до одного листа на самом нижнем уровне.

Таким образом, модифицировав алгоритм полного поиска с учетом современной элементной базы удалось получить полный класс последовательностей на рекордных длинах до $N = 70$. Дополнительную информацию по периодическим и апериодическим последовательностям с идеальными и оптимальными корреляционными характеристиками можно найти на тематическом сайте [9].

Список литературы:

1. Barker R.H. Group synchronizing of binary digital systems, Communication Theory (W. Jackson, ed.), Academic Press, New York, 1953. — Pp. 273–287.
2. Свердлик М.Б.Оптимальные дискретные сигналы // Сов. Радио, 1975. — 200 с.
3. Свердлик М.Б. Расчет ФМ сигналов с хорошими корреляционными свойствами // Известия вузов. Радиоэлектроника, 1971. — Т. 14. — №12.
4. Titsworth R.C. Optimal and minimax Sequences // Proc. in International Telemetry Conferece, 1963.
5. Binary pulse compression codes // IEEE Trans, 1967. v. IT-13, no. 2.
6. Пелехатый М.И. О некоторых блок-конструкциях, порождающих последовательности с хорошими корреляционными свойствами // Радиотехника и электроника, 1970. — Т. 15. — №7.
7. Пелехатый М.И. Дополнение к статье «О некоторых блок-конструкциях, порождающих последовательности с хорошими корреляционными свойствами» // Радиотехника и электроника, 1971. — Т. 16. — №7.
8. Coxson G.E. Efficient exhaustive search for optimal-peak-sidelobe binary codes // IEEE Trans. Aerospace and Electron. Systems, 2005, V. 41. — Pp. 302–308 / G.E. Coxson, J. Russo.
9. Тематический сайт, посвященный синтезу сигналов и их применению [Электронный ресурс]. Дата обновления: 15.02.2013 // URL: http://signalslab.marstu.net (дата обращения: 15.02.2013).

References (transliteration):

1. Barker R.H. Group synchronizing of binary digital systems, Communication Theory (W. Jackson, ed.), Academic Press, New York, 1953. — Pp. 273-287.
2. Sverdlik M.B.Optimal'nye diskretnye signaly // Sov. Radio, 1975. — 200 s.
3. Sverdlik M.B. Raschet FM signalov s horoshimi korrelyacionnymi svoistvami // Izvestiya vuzov. Radioelektronika, 1971. — T. 14. — №12.
4. Titsworth R.C. Optimal and minimax Sequences // Proc. in International Telemetry Conferece, 1963.
5. Binary pulse compression codes // IEEE Trans, 1967. v. IT-13, no. 2.
6. Pelehatyi M.I. O nekotoryh blok-konstrukciyah, porozhdayushih posledovatel'nosti s horoshimi korrelyacionnymi svoistvami // Radiotehnika i elektronika, 1970. — T. 15. — №7.
7. Pelehatyi M.I. Dopolnenie k stat'e «O nekotoryh blok-konstrukciyah, porozhdayushih posledovatel'nosti s horoshimi korrelyacionnymi svoistvami» // Radiotehnika i elektronika, 1971. — T. 16. — №7.
8. Coxson G.E. Efficient exhaustive search for optimal-peak-sidelobe binary codes // IEEE Trans. Aerospace and Electron. Systems, 2005, V. 41. — Pp. 302-308 / G.E. Coxson, J. Russo.
9. Tematicheskii sait, posvyashennyi sintezu signalov i ih primeneniyu [Elektronnyi resurs]. Data obnovleniya: 15.02.2013 // URL: http://signalslab.marstu.net (data obrasheniya: 15.02.2013).

DOI: 10.7256/2305-6061.2013.2.7600