ИСПОЛЬЗОВАНИЕ ТЕРНАРНЫХ ДЕРЕВЬЕВ ДЛЯ ХРАНЕНИЯ ДАННЫХ ВЫЧИСЛИТЕЛЬНОГО ЭКСПЕРИМЕНТА

А.Н. Коварцев, Д.А. Попова-Коварцева, Е.Е. Горшкова

Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет) (СГАУ), Самара, Россия

При реализации численных методов глобальной оптимизации или методов тестирования программных модулей часто возникает необходимость организации хранилища для значительных объёмов «испытаний» исследуемой функции. В связи с чем возникает проблема быстрого поиска нужного элемента. Выход из сложившейся ситуации можно найти за счет организации эффективного хранилища данных, допускающего быструю реализацию операций поиска и вставки новых элементов. В статье рассматриваются варианты хранения данных в бинарных и тернарных деревьях, производится их сравнение, приводятся рекомендации по использованию рассмотренных моделей данных.

Ключевые слова: бинарные деревья, тернарные деревья, структуры данных, глобальная оптимизация.

Введение

При реализации численных методов глобальной оптимизации (ГО) [1] или методов тестирования программных модулей [2] возникает необходимость хранения значительных объёмов «испытаний» функции. Основным требованием к алгоритмам ГО и тестирования вычислительных модулей является скорость работы алгоритма. Предположим, что время вычисления функции $f(x)$ достаточно большое. Такая ситуация характерна при решении задач оптимизации в процессе проектирования технических систем, например, газотурбинных двигателей, летательных аппаратов и т.д. В известных алгоритмах ГО, например, методе половинных делений или его модифицированной версии [3] некоторые испытания повторяются кратно, что является одной из причин избыточности вычислений $f(x)$. Выход из сложившейся ситуации можно найти за счет организации эффективного хранилища данных, допускающего быструю реализацию операций поиска и вставки новых элементов.

1. Недостатки использования традиционных бинарных деревьев

Условимся, что под испытание мы будем понимать вычисление функции $f(x)$ в точке, заданной алгоритмом оптимизации или тестирования функции. Под трудоемкостью методов оптимизации обычно понимают общее число шагов работы алгоритма [4]. Однако, учитывая высокую производительность современной вычислительной техники, под трудоемкостью оптимизационных алгоритмов правильнее понимать общее число обращений к оптимизируемой функции, поскольку шаги алгоритма могут иметь разную сложность, а каждое обращение к функции занимает одинаковое время для любого алгоритма. С другой стороны, число обращений к функции не зависит от производительности ЭВМ, что позволяет сравнивать между собой алгоритмы, реализованные на ЭВМ имеющих разную производительность.

В тоже время, в зависимости от сложности тестируемой функции, каждое обращение к функции может требовать значительного машинного времени.

В данной статье рассматривается проблема организации хранилища данных для алгоритма ГО, основанного на модифицированном методе половинных делений (ММПД) [3]. В алгоритме ММПД стратегию разбиения пространства поиска можно охарактеризовать как неоднородное деление пространства с помощью регулярных структур (см. [1]). Однако при этом возникает необходимость вычисления функции в одних и тех же точках пространства независимых переменных.

Для того чтобы не производить повторных вычислений, целесообразно хранить координаты точек и значения функции в них в отдельном хранилище данных. При этом массив данных может иметь значительные размеры. В этом случае возникает проблема поиска элемента в массиве. Например, средняя сложность последовательного поиска, как в неупорядоченном, так и упорядоченном массиве равна O(n) [5].

На практике, для повышения эффективности алгоритма поиска, обычно используют двоичные (бинарные) деревья (b-деревья) [6, 7]. Двоичный поиск для равновесных бинарных деревьев [8] имеет среднюю сложность пропорциональную $O(\log_2 n)$ и зависит от глубины бинарного дерева.

Под глубиной бинарного дерева обычно понимают [8] число вершин на самом длинном пути от корня дерева к листьям. Однако использование бинарных деревьев для хранения данных в точках испытаний тестируемой или оптимизируемой функции имеет свои особенности. В частности, даже для регулярных сеток построение оптимального равновесного двоичного дерева требует определенного порядка ввода данных или дополнительная сортировка массива данных. Для нерегулярных сеток эксперимента без сортировки данных обойтись невозможно.

На рисунке 2 показано равновесное бинарное дерево, построенное для регулярной сетки (см. рис. 1) одним из рациональных способов построения равновесного бинарного дерева.

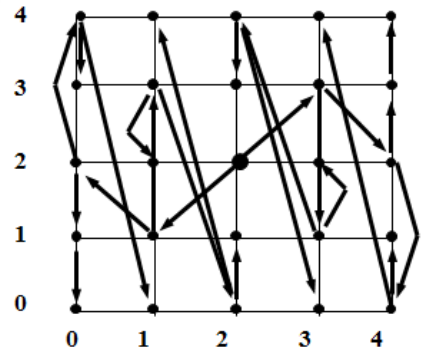

Рис. 1. Стратегия построения равновесного бинарного дерева на регулярной сетке

При этом вычислительный эксперимент должен соответствовать последовательности, представленной на рисунке 1. Вычисления начинаются из точки с координатами (2, 2). Данному плану соответствует бинарное дерево, представленное на рисунке 2.

Однако, для метода ММПД свойственно постепенная детализация области поиска, начиная с большого шага дискретизации с постепенным его уменьшением в отдельных фрагментах области поиска. При этом область поиска «уплотняется» неравномерно. В этом случае добиться равновесности бинарного дерева практически невозможно, что приводит к потере эффективности алгоритмов поиска. На рисунке 3 представлено бинарное дерево, соответствующее эксперименту, представленному на рисунке 4.

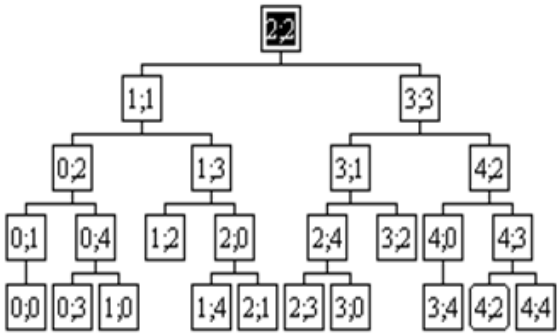

Рис. 2. Равновесное бинарное дерево для сетки эксперимента 5*5

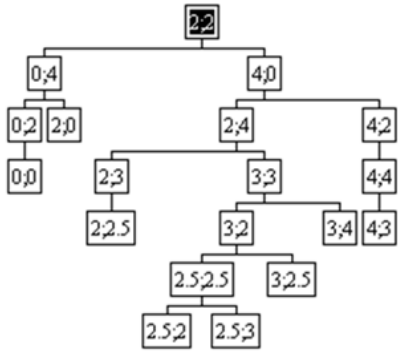

Рис. 3. Бинарное дерево для метода ММПД

Как видно из рисунка для нерегулярной сетки бинарное дерево становится неравновесным.

Одним из способов повышения эффективности алгоритма поиска является использование более сложных структур данных, например, тернарных деревьев (t-деревьев).

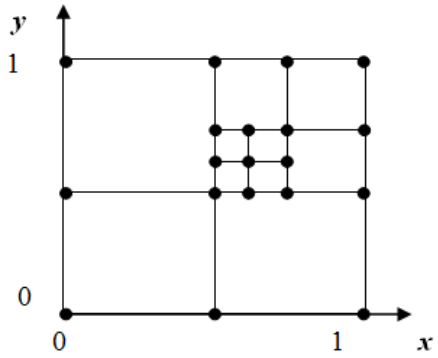

Рис. 4. Построение равновесного бинарного дерева

2. Тернарные деревья

Для построения бинарного дерева используется два отношения порядка (“<”и “>=” или “<=” и “>”), что позволяет реализовать процедуру двоичного деления упорядоченного множества. Для построения тернарного дерева требуется три условия, позволяющее разбить упорядоченное множество на три части. В качестве таковых выберем “<”, “=” и “>”. Точки эксперимента будем упорядочивать по критерию, $z = x + y$ т.е. по сумме координат экспериментальных точек. В общем случае точки с координатами, соответствующие

условию равенства, встречаются не слишком часто и в этом случае тернарное дерево превращается в бинарное. Однако для регулярных сеток таких точек достаточно много. На рисунке 5 для сетки 5*5 на диагоналях ($x+y=const$) расположены точки, удовлетворяющие условию равенства.

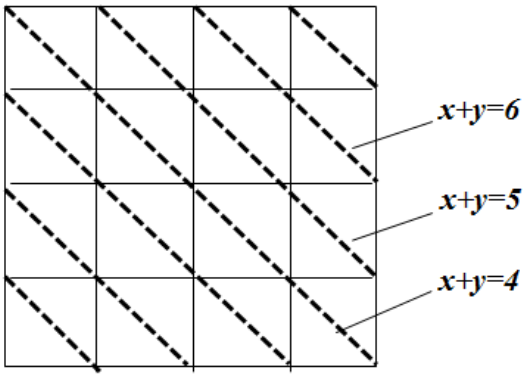

Рис. 5. Построение тернарного дерева

В результате формируется t-дерево, представленное на рисунке 6.

Определенный интерес представляет сравнение между собой средней эффективностей поиска для b и t деревьев для различных регулярных сеток разбиения пространства поиска.

Пусть n – количество узлов регулярной сетки по каждой координате, $N=n^2$ - общее количество узлов сетки, $k_{пол}$ - глубина полного двоичного дерева [8].

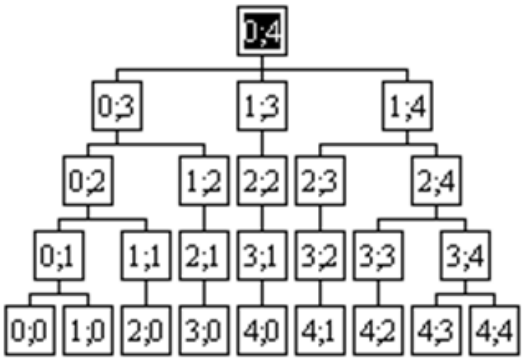

Рис. 6. Тернарное дерево для сетки эксперимента 5*5

Бинарное дерево "полно", если оба поддерева всякой вершины "равновесны", т.е. имеют приблизительно одинаковое количество вершин [8].

Для "полного" (равновесного) b-дерева общее число вершин можно подсчитать по формуле [5] $2^{k_{пол}}-1$.

Предположим, что для каждой регулярной сетки заданного шага строится полное равновесное дерево таким образом, чтобы несколько первых уровней, кроме последнего, были полностью заполнены вершинами. В принципе невозможно построить полностью заполненные "равновесные" деревья для сеток размерности $n\times n$, поскольку у сетки и у полного двоичного дерева не совпадает число элементов.

Для произвольного числа элементов, в нашем случае сетки $n\times n$, глубину "полного" дерева можно подсчитать по формуле:

$$k_{пол}=\begin{cases}\lceil \log_2 N\rceil, & если \quad \lceil \log_2 N\rceil=\log_2 N,\\ \lceil \log_2 N\rceil+1, & иначе.\end{cases}$$

В предположении о равновероятном наполнении двоичного дерева информацией в узлах сетки подсчитаем среднюю сложность поиска на "полном" b-дереве. Для сетки с числом элементов $N = n^2$ элементы "полного" дерева до глубины $k_{пол} - 1$ будут полностью заполнены. На каждом k-м уровне двоичного дерева располагается 2^k элементов. Несложно показать, что при равновероятном распределении узлов регулярной сетки в двоичном дереве средняя сложность поиска может быть подсчитана по формуле

$$B(N) = \frac{1}{n^2}(k_{пол} n^2 + (k_{пол} + 1) - 2^{k_{пол}}).$$

К сожалению "полное", равновесное b-дерево формируется только при определенном порядке включения элементов сетки в бинарное дерево. В общем случае в процессе поиска может сформироваться линейная структура с большей глубиной, чем у "полного" b-дерева.

Кнут [9] показал, что, если все возможные варианты включения (N!) элементов в бинарное дерево считать равновероятными, то средняя глубина двоичного поиска равна $\sqrt{N}$ и потребуется в среднем $1.386 \log_2 N$ сравнений на один поиск. На рисунке 7 показаны графики средних сложностей поиска. Видно, что средняя сложность поиска "полного" двоичного дерева значительно меньше, чем при произвольном способе наполнения b-дерева элементами сетки.

Для тернарного дерева его глубина почти совпадает с размерностью сетки, поскольку из-за операции равенства глубина дерева не может быть меньше количества узлов, лежащих на диагонали сетки. Для тернарного дерева значительно труднее подсчитать среднюю сложность поиска $T(N)$. На рисунке 7 показан график изменения $T(N)$ от числа элементов сетки, полученный на основе вычислительного эксперимента для квазиоптимальных t-деревьев.

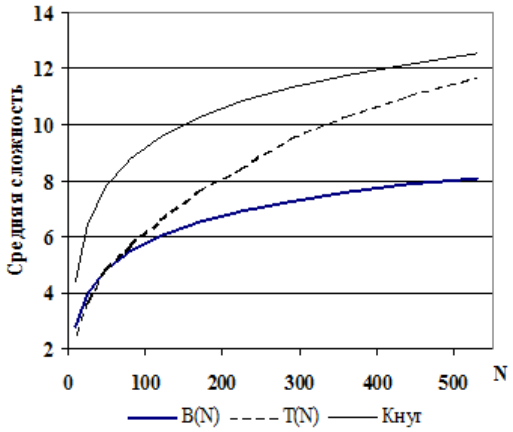

Рис. 7. Сравнение средних сложностей поиска на b и t деревьях

Как видно из рисунка тернарное дерево проигрывает при описании регулярной сетки эксперимента оптимальному бинарному дереву. Однако при нелинейном способе разбиения пространства поиска, при изменении шага дискретизации пространства поиска, большое значение имеет запас свободных веток в дереве, которые могут быть заполнены новыми узлами сетки. В этом смысле бинарное дерево значительно проигрывает t-дереву, поскольку в b-дереве свободные ветки находятся на последнем ярусе дерева.

В t-дереве происходит резервирование свободных мест для новых точек эксперимента, которые достаточно долго не увеличивают глубину поиска. Последнее происходит за счет того, что операция равенства “=” реализует разбиение множества точек сетки на классы эквивалентности. Все классы сравнимы между собой по операциям “<” или “>” и образуют «ветви» эквивалентности.

Уплотнение сетки в отдельных квадратах приводит к появлению новых веток, которые быстро находят свободное место на дереве. Для бинарных деревьев, как правило, верхние уровни к этому моменту времени уже заполнены и дальнейшее развитие возможно только за счет увеличения глубины дерева.

Сортировка b-дерева в целях улучшения его структуры, когда операции включения элемента в структуру дерева и поиск элемента на дереве производятся одновременно, нецелесообразно. Самые быстрые алгоритмы сортировки требуют $N \log_2 N$ операций на улучшение структуры дерева [10]. В этом случае эффективность от дихотомического поиска на равновесном b-дереве будет потеряна за счет выполнения операций сортировки.

Таким образом, в качестве модели данных размещения экспериментальных точек в процессе тестирования программного модуля можно использовать тернарное дерево.

3. Исследование эффективности алгоритма поиска на тернарном дереве

Эффективность алгоритма поиска на тернарном дереве во многом зависит от свойств тестируемой функции. В вычислительных экспериментах использовалось 6 тестовых функций разной топологии и, следовательно, разной вычислительной сложности с точки зрения решения задачи ГО. От сравнительно простых рациональных функций, более сложных овражных функций, многоэкстремальных функций (в том числе и разрывных), до стандартных тестовых функций ROOT предложенных на сайте [11].

На рисунке 8 представлены результаты вычислительных экспериментов при исследовании эффективности алгоритма поиска на тернарных деревьях для рассмотренных тестовых функций. Для наглядности приведены графики зависимостей средней длины поиска оптимального "полного" двоичного дерева и оценки той же величины для равновероятного способа наполнения b-дерева, предложенного Кнутом [9].

Как видно из рисунка, в большинстве случаев эффективность поиска на t-дереве сравнима со средней сложностью поиска на оптимальном двоичном дереве. Последнее достигается за счет большого количества свободных связей тернарного дерева, находящихся на верхних ярусах. В целом средняя длина поиска t-дерева имеет приемлемую величину и для больших объемов хранимой информации. При сравнении следует учесть, что t-дерево рассматривалось в неотсортированном виде, в то время как b-дерево для тех же объемов информации является оптимальным и равновесным.

Последнее говорит о высокой эффективности поиска на t-деревьях.

Если рассмотреть относительную среднюю длину поиска (отношение средней длины поиска к общему числу элементов дерева), то с ростом объема t-дерева эффективность поиска только повышается (см. рис. 9).

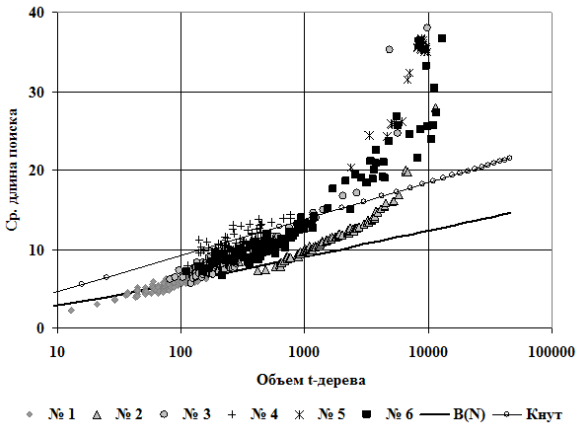

Рис. 8. Эффективность поиска на t-деревьях

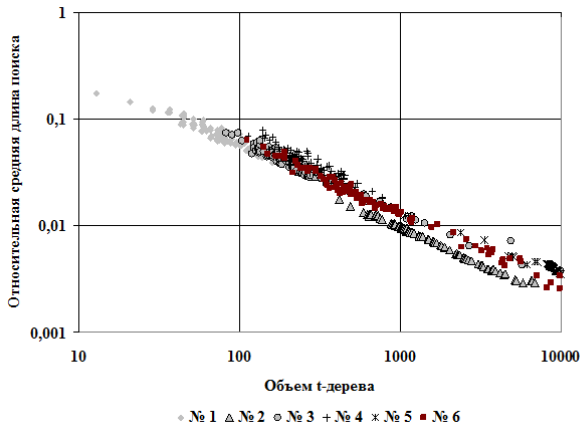

Рис. 9. Относительная эффективность поиска на t-деревьях

Заключение

В данной статье мы рассмотрели задачу организации хранилища данных и алгоритм поиска данных на тернарных деревьях, предназначенный для обслуживания методов глобальной оптимизации и тестирования функций, когда требуется обработка значительных объемов информации. Использование тернарных деревьев даёт ряд преимуществ по сравнению с применением традиционных бинарных деревьев, поскольку для повышения эффективности поиска на тернарных деревьях не требуется выполнять операции сортировки или уравновешивания дерева, а средняя сложность поиска сравнима эффективностью поиска на двоичном оптимальном равновесном дереве.

Работа выполнена при государственной поддержке Министерства образования и науки РФ в рамках реализации мероприятия Программы повышения конкурентоспособности СГАУ среди ведущих мировых научно-образовательных центров на 2013-2020 годы.

Литература

1. Коварцев, А.Н. К вопросу об эффективности параллельных алгоритмов глобальной оптимизации функций многих переменных. / А.Н. Коварцев, Д.А. Попова-Коварцева. // Компьютерная оптика. – 2011. - Т.35, №2. – С.256-261
2. Коварцев, А.Н. Исследование эффективности глобальной параллельной оптимизации функций многих переменных / А.Н. Коварцев, Д.А. Попова-Коварцева., П.В. Аболмасов // Вестник ННГУ.- 2013. - №3 (1). - С. 252-261.
3. Kovartsev, A.N. Software testing based on global search of several variables functions discontinuity / A.N. Kovartsev, D.A. Popova-Kovartseva, E.E. Gorshkova // Proceedings of International Conference Information Technology and Nanotechnology (ITNT-2015). Samara, Russia, June 29 - July 1, 2015 – P. 389-396.
4. Немировский, А.С. Сложность задач и эффективность методов оптимизации / А.С. Немировский, Д.Б. Юдин. – М.: Наука, 1979. 384 с.
5. Макконел, Дж. Анализ алгоритмов. Вводный курс.- М.: Техносфера, 2002.- 304 с.
6. Адаменко, А.Н. Логическое программирование и Visual Prolog / А.Н. Адаменко, А.М. Кучуков– СПб.: БХВ-Петербург, 2003. – 992 с
7. Иванов, Б.Н. Дискретная математика. Алгоритмы и программы: Учеб. Пособие. – М.: Лаборатория Базовых Знаний, 2001. – 288 с
8. Мейер, Б. Методы программирования: В 3-х томах. Т.2./ Б. Мейер, К. Болдуин. – М.: Мир, 1982. – 386 с.
9. Кнут, Д. Искусство программирования для ЭВМ. Сортировка и поиск. Т.3. – М.: Мир, 1978. – 848 с
10. Холл, П. Вычислительные структуры. Введение в нечисленное программирование. – М.: Мир, 1978.- 216 с
11. Алгоритмы глобальной оптимизации [Электронный ресурс]. – URL: http:www.r-tech.narod.ru/gtest/html (дата обращения 20.03.2016).