УДК 004.02

М.С.Куприянов, Ю.А.Шичкина

Распараллеливание процесса построения дерева решений

АННОТАЦИЯ. Стремительное развитие информационных технологий, в частности, прогресс в методах сбора, хранения и обработки данных позволил многим организациям собирать огромные массивы данных, которые необходимо анализировать. Объемы этих данных настолько велики, что возможностей экспертов уже не хватает, что породило спрос на методы автоматического исследования данных, который с каждым годом постоянно увеличивается. Одним из таких методов автоматического анализа данных является построение дерева решений. В данной статье рассматриваются вопросы построения параллельного алгоритма вывода дерева решений и оценки объема вычислительных ресурсов многопроцессорной системы, необходимых для реализации данного алгоритма. Входными данными при получении оценок являются высота информационного графа и данные о высоте каждого яруса графа. Выходными данными являются границы интервала, покрывающего объем вычислительных ресурсов, и позволяющие принимать решения о дальнейшем применении заданного алгоритма.

Ключевые слова: дерево решений, правила, параллельные вычисления, алгоритм, ID3, список смежности.

Введение

Современные базы данных содержат так много данных, что практически невозможно вручную проанализировать их для извлечения ценной информации, помогающей принимать важные решения. В связи этим спрос на методы автоматического исследования данных с каждым годом постоянно увеличивается.

Деревья решений – один из таких методов автоматического анализа данных. Область применения деревья решений в настоящее время широка, но все задачи, решаемые этим аппаратом, могут быть как известно, объединены в три класса: описание данных, классификация и зависимость целевой переменной от других переменных.

На сегодняшний день существует значительное число алгоритмов,

реализующих деревья решений CART, C4.5, NewId, ITrule, CHAID, CN2 и т.д.

На практике в результате работы этих алгоритмов часто получаются слишком детализированные деревья, которые при их дальнейшем применении дают много ошибок. Это связано с явлением переобучения. Для сокращения деревьев используется отсечение ветвей. Еще один существенный недостаток практического применения алгоритма построения деревьев решений состоит в том то, что оно основано на эвристических алгоритмах, таких как алгоритм «жадности», где единственно оптимальное решение выбирается локально в каждом узле. Такие алгоритмы не могут обеспечить оптимальность всего дерева в целом и являются трудоемкими. И наконец, практически все эти алгоритмы с трудом поддаются распараллеливанию.

В последнее время формальным моделям параллельных вычислений было посвящено много работ отечественных ученых: В.Воеводин и Вл.Воеводин, В. Гурман, О.Омаров, В.Котов, И.Вирбицкайте, А.Барский, В.Бочаров, В.Корнеев, Н.Миренкова и зарубежных исследователей: Я.Фостер, Ч.Хоар, Р.Милнер, К.Петри, Г.Винскель, М.Нильсен, Э.Гобо, М.Беднарчик. Наиболее общими математическими моделями параллельных вычислений являются: граф алгоритма, информационный граф алгоритма, граф зависимости и граф процесса [2]. Все эти модели позволяют исследовать проблемы, связанные с достижимостью некоторого состояния вычислительного процесса, его нетупиковостью, решать проблемы синхронизации или анализировать состояние системы различными методами. Одной из проблем, решаемых на основе формальных моделей, является преобразование вычислительного процесса с сохранением одних параметров и улучшением других. Ручная оптимизация программ является достаточно дорогостоящим процессом. Виду этого особый интерес представляют распараллеливающие системы. К настоящему времени разработан ряд методов решения задачи построения расписания выполнения задач обработки информации на вычислительной системе заданной структуры [1, 3-5]: – «жадные» алгоритмы и сети Хопфилда

для решения задачи построения расписаний; –генетические и эволюционные алгоритмы, настраиваемые на конкретный вариант постановки задачи. К основным недостаткам существующих методов оптимизации параллельных вычислений можно отнести: – отсутствие этапа формального анализа параллельной структуры алгоритма, для реализации которого проектируется вычислительная система — вместо этого предлагается применять различные эвристики; – необходимость на каждом шаге поиска оптимальной структуры вычислительной системы решать задачу построения по возможности оптимального расписания организации вычислительного процесса (являющуюся, в общем случае, NP-полной задачей); – зависимость качества получаемых с помощью эвристик решений от начального приближения и структуры исследуемого алгоритма обработки информации; – отсутствие методик оценивания качества получаемой в результате оптимизации параллельной структуры.

В данной статье приводится решение последней из перечисленных проблем применительно к алгоритмам построения деревьев решений, т.е. способы получения параллельного алгоритма и численного определения степени близости полученного после оптимизации алгоритма к теоретически оптимальному алгоритму.

Применение графов для построения деревьев решения

Пусть обучаемое множество представляют собой набор объектов C = C1, C2, ... Каждый объект Ci = x1, x2, ... является вектором, где x1, x2, ... представляют собой атрибуты объекта.

Обозначим: m – число атрибутов, n – объем выборки, i – номер атрибута, i=1..m, m_i – число значений i-го атрибута, j – номер значения в атрибуте, j=1..m_i, cij – количество значений в выборке, соответствующих j-му значению i-го атрибута. Например, если C1 – первый атрибут и C1={sunny, overcast, rain}, то c13=количество в выборке значений «rain». Очевидно, что

$$\sum_{i=1}^{m}\sum_{j=1}^{m_i} c_{ij} = mn$$

На основе данных об атрибутах объектов и множестве решений S, можно построить граф взаимосвязей значений атрибутов G(V,E), где $V = \{v_1, v_2, \ldots, v_p\}$ и $E = \{e_1, e_2, \ldots, e_q\}$ - соответственно множества вершин и ребер графа. Множество V – совокупность всех значений атрибутов обучаемого множества, $p = \sum_{i=1}^{m} m_i$, E – совокупность всевозможных в обучаемом наборе связей между атрибутами, $q = \sum_{i=1}^{m} m_i \left(\sum_{j=i+1}^{m} mj \right)$. В полученном графе каждая вершина, соответствующая одному значению своего атрибута, будет связана с каждой вершиной, соответствующей отдельному значению другого атрибута. Граф не будет являться полным и, следовательно, $q < \frac{p(p-1)}{2}$. Пример получаемого графа, изображен на рисунке 1.

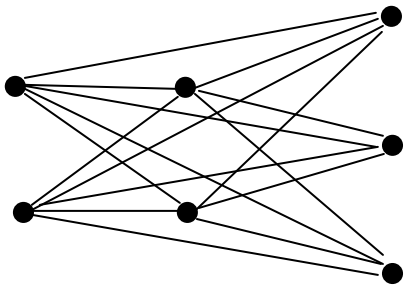

Рисунок 1 – Граф взаимосвязей значений атрибутов

Для полноты графой модели преобразуем граф взаимосвязей значений атрибутов во взвешенный граф, добавим в него в качестве весов ребер количество связей значений i-го и j-го атрибутов.

Граф взаимосвязей значений атрибутов не является ориентированным и по своей сути характеризует многосвязную сложную систему.

Систему взаимосвязанных значений атрибутов можно в соответствии с определением сложной системы разбить на две подсистемы по принципу их принадлежности тому или иному значению из множества решений S.

Собственно на этом разбиении системы значений атрибутов на несколько подсистем и основан метод построения дерева решений, представляющий

собой прямой метод.

Прямой матричный метод построения дерева решений

1. Разбить обучаемое множество на два непересекающихся подмножества S1 и S2, в соответствии с принципом принадлежности отдельного объекта той или иной группе множества решений S.

2. Построить взвешенный граф взаимосвязей значений атрибутов для каждого подмножества.

3. Построить матрицу смежности для каждого графа. Так как матрица будет симметричной относительно главной диагонали, то можно изначально строить верхнее треугольную матрицу смежности.

4. На главной диагонали матрицы смежности разместить значения cij.

5. Рассчитать приоритетность атрибутов в соответствии с формулами энтропии и Gain-индекса:

$$Entropy(S)=-\sum_{i=1}^{2} p_i \log_2 p_i \ ;$$

$$Gain(S,Ci)=Entropy(S)-\sum_{i=1}^{2} \frac{|S_i|}{|S|} Entropy(S_i) \ ;$$

6. Если в матрице смежности Tk существует нулевая строка Ci, то можно сформулировать правило Ci -> S-Sk.

7. Провести поэлементное сравнение матриц множеств S1 и S2. Если T1(i,j)=0, а T2(i,j)>0, то можно сформулировать правило: Ck,Cr->S2, где Ck – атрибут и его значение, соответствующие i-й строке, а Cr - атрибут и его значение, соответствующие j-му столбцу матрицы смежности.

8. Если T2(i,j)=0, а T1(i,j)>0, то можно сформулировать правило: Ck,Cr->S1, где Ck – атрибут и его значение, соответствующие i-й строке, а Cr - атрибут и его значение, соответствующие j-му столбцу матрицы смежности.

9. Если в i-й строке несколько элементов, удовлетворяющих условию 7 или 8, то выбирается атрибут с наибольшим Gain-индексом.

Регулирования глубины дерева

В теории построения деревьев решений существует понятие Регулирования глубины дерева, как техники, которая позволяет уменьшать размер дерева решений, удаляя участки дерева, которые имеют маленький вес или низкую вероятность адекватности решения.

Один из вопросов, который возникает в эвристических алгоритмах дерева решений – это оптимальный размер конечного дерева. Так, небольшое дерево может не охватить ту или иную важную информацию о выборочном пространстве. Тем не менее, трудно сказать, когда алгоритм должен остановиться, потому что невозможно спрогнозировать, добавление какого узла позволит значительно уменьшить ошибку. Эта проблема известна как «эффект горизонта». Тем не менее, общая стратегия ограничения дерева сохраняется, т.е. удаление узлов реализуется в случае, если они не дают дополнительной информации.

В нашем случае для решения задачи глубины дерева можно рассчитать вес каждой ветви по формуле:

$$p_i = \frac{T(i, j)}{Cij} 100\%$$

Следует отметить, что совокупность ветвей с весом, равным 100%, получаемая с помощью описанного прямого матричного метода, полностью соответствует совокупности ветвей, получаемых с помощью эвристического алгоритма ID3.

Совокупность правил, получаемых с помощью матричного метода, является более широкой по сравнению, к примеру, с методом ID3, но может быть отрегулирована с помощью несложных расчетов.

Распараллеливание метода построения дерева решений

Матричный метод предпочтительнее эвристических алгоритмов за счет более низкой трудоемкости. В общем случае, трудоемкость эвристических алгоритмов равна $O(rm^2k^2)$, где: m – число атрибутов, k – наибольшее значение

вариантов значений в атрибуте, r – число итераций по построению генов, трудоемкость матричного метода – $O(m^2k^2)$.

Данный метод легко подается автоматическому анализу и распараллеливанию с помощью информационного графа и списков смежности. Хранить и обрабатывать все элементы матрицы информационного графа, большинство из которых нулевые, является не рациональным.

При построении информационного графа предполагается, что время выполнения любых вычислительных операций является одинаковым и равняется одной условной единице, а передача данных между вычислительными устройствами выполняется мгновенно без каких-либо затрат времени.

Список смежности – совокупность пар смежных вершин, составленных по правилу: если из вершины v_i следует вершина v_j, то пара имеет вид $\langle v_i, v_j \rangle$.

Метод поиска оптимального по ширине и высоте информационного графа состоит из двух частей: разбиение совокупности вершин на группы (построение параллельной формы) и перемещение вершин между группами с целью уменьшения ширины яруса и получения максимальной плотности.

Прямой ход

1. Для заданного информационного графа построить соответствующий ему список смежности $\langle V_1, V_2 \rangle$, где V_1 - начальная вершина направленного ребра, V_2 - конечная вершина.

2. Найти множество $V = V_1 - V_2$. Вершины, вошедшие в это множество, составят группу вершин, принадлежащих одному ярусу.

3. Удалить из списка смежности все пары, начальная вершина которых совпадает с одной из вершин множества V.

4. Если список не пустой, то вернуться на шаг 2. Если в списке не осталось ни одной пары, то первая часть метода – разбиение совокупности вершин на группы – окончена.

При оптимизации информационного графа по ширине, передвигать вершины графа можно только в направлении от входных вершин к выходным, поэтому после окончания второй части метода можно рассчитать уточненную оценку минимальной ширины с учетом числа групп:

$$d' = \max_{1 \le k \le m} \left\lceil \frac{\sum_{i=i_k}^{n} d_i}{s-k+1} \right\rceil \qquad (3.4)$$

где m – число групп, i_k – номер первой вершины в k-й группе.

Формула (3.4) является, более приближенной к практике, оценкой минимальной ширины графа, но и она не является верхней границей. Если выходная вершина одна, то, как минимум, одна группа (последняя) будет состоять из одной вершины. Следовательно, плотность на других ярусах будет выше. Аналогичный вывод можно сделать для входных вершин. Поэтому рассчитывать верхнюю оценку минимальной ширины графа следует с учетом числа выходных вершин.

Обозначим $\Delta_{вх} = d' - n_{вх}$ - разность между шириной графа и числом входных вершин, где $n_{вх}$ - количество входных вершин. Аналогично, $\Delta_{вых} = d' - n_{вых}$, где $n_{вых}$ - количество выходных вершин Рассмотрим возможные варианты:

1) $\Delta_{вх} > 0, \Delta_{вых} > 0$. Плотность остальных групп повышается на величину:

$$\Delta = \left\rceil \frac{\Delta_{вх} + \Delta_{вых}}{g-2} \right\lfloor, \qquad (3.5)$$

где g – число групп, полученных в результате первой части метода. Следовательно: $d' = d' + \Delta$.

2) $\Delta_{вх} > 0, \Delta_{вых} < 0$. Количество выходных вершин больше минимальной ширины группы. Плотность последней группы уменьшить нельзя, поэтому: $d' = n_{вых}$.

3) $\Delta_{вх} < 0, \Delta_{вых} > 0$. Количество входных вершин больше минимальной ширины группы. Специфика информационного графа такова, что в зависимости от информационных связей между первой и последующими группами, в результате второй части метода ширина первой группы может быть и уменьшена и остаться в первоначальном состоянии. Поэтому поведение оценки минимальной ширины в данном случае остается под вопросом.

4) $\Delta_{вх} < 0, \Delta_{вых} < 0$. Данный случай является композицией 2 и 3-го вариантов. Поэтому: $d' = n_{вых}$.

С учетом того, что число выходных вершин уже учтено в формуле (3.5), то от последних трех вариантов можно отказаться, оставив верхнюю оценку минимальной ширины в следующем виде:

$$d' = d' + \Delta, \tag{3.6}$$

где:

$$\Delta = \begin{cases} \dfrac{\Delta_{вх} + \Delta_{вых}}{g - 2}, & \Delta_{вх} > 0 \cap \Delta_{вых} > 0 \\ 0, \textit{ в остальных случаях} \end{cases} \tag{3.7}$$

Если свести полученные результаты на одну числовую ось d, то можно определить границы практической минимальной ширины информационного графа (рис. 3.6):

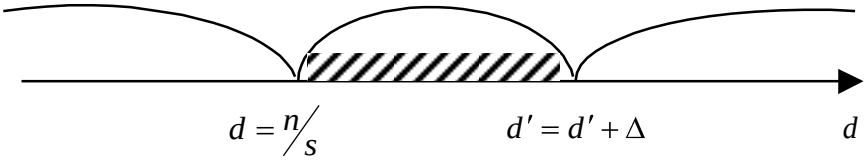

Рисунок 3.6. – Интервал практической минимальной ширины информационного графа.

Таким образом, после применения первой части метода оптимизации параллельного алгоритма по ширине можно определить границы минимальной ширины графа, и уже во время применения второй части метода оптимизации по ширине судить об эффективности его применения. В самом методе во второй части, в зависимости от архитектуры используемого вычислительного кластера, можно применять вместо теоретической оценки минимальной ширины d

практическое значение d', или любое другое значение, диктуемое условиями решения задачи.

Обратный ход

Примечание. Вторая часть – равномерное распределение вершин по группам – начинается с предпоследней группы.

5. Рассчитать значение теоретической минимальной ширины d информационного графа. Выбрать первую с конца группу с числом вершин меньшим теоретической минимальной ширины графа d . Обозначим ее M_i.

6. Составить список смежности для каждой из вершин, входящих в группу $M_{i-1}: V_{i-1j} \langle V_1, V_2 \rangle$, где $j = \overline{1,k}$, k – количество вершин в группе M_{i-1}.

7. Найти пересечения множеств: $V = M_i \cap V_2$. Если для некоторой вершины группы M_{i-1} пересечение пустое, то эту вершину можно перенести в группу M_i.

Примечание. На этом шаге следует переносить вершины из группы M_{i-1} в группу M_i до тех пор, пока число вершин группы M_i не достигнет теоретической минимальной ширины графа d.

8. Если $i > 1$, то перейти к шагу 6, иначе метод окончен.

Результаты

Проведенный анализ алгоритмов построения деревьев решений в совокупности с разработанной программой, реализующей алгоритм построения дерева решений ID3, показал, что наряду с основным достоинством – всегда приводить к результату, рассматриваемые алгоритмы обладают рядом недостатков, основные из которых – небольшой внутренний параллелизм этих алгоритмов и высокая трудоемкость.

Матричный метод построения дерева решений позволяет получать более широкую совокупность правил по сравнению, к примеру, с методом ID3.

Совокупность правил может быть отрегулирована с помощью выведенных формул, позволяющих также оценить вероятность срабатывания правила. Разработанный матричный метод также предпочтительнее эвристических алгоритмов за счет более низкой трудоемкости.

Информационный граф последовательного алгоритма, соответствующего матричному методу вывода дерева решений, можно оптимизировать с помощью метода списков смежности, сократив трудоемкость матричного метода с $O(m^2k^2)$ до $O(mk)$ на mk процессорах.

Все методы были протестированы с помощью программ, написанных на fortran 90 под управлением ОС Linux Slakeware.

Список литературы.

1. Вавинов С.В., Костенко В.А. Параметризованный жадный алгоритм построения статических расписаний. - М.: Изд-во МГУ, 2003.

2. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.:БХВ-Петербург, 2004. – 608с.

3. Гурман В. И., Трушкова Е. А. Приближенные методы оптимизации управляемых процессов. - Программные системы: теория и приложения, № 4(4), 2010, с. 85–104.

4. Корячко В. П., Скворцов С. В. Иерархическая модель глобальной оптимизации у параллельных объектных программ. Наука и образование, 2006, №8.

5. Костенецкий П.С., Лепихов А.В., Соколинский Л.Б. Технологии параллельных систем баз данных для иерархических многопроцессорных сред // Автоматика и телемеханика. -2007. -Том 68, №5. -С. 847-859.

6. Шичкина Ю.А., В.И.Воробьев Оптимизация параллельного алгоритма по числу процессов. Вестник гражданских инженеров.– 2008. - № 2(15). – С. 92-97.

7. Шичкина Ю.А., Комбинированный метод многопараметрической

оптимизации взвешенного информационного графа. Системы. Методы. Технологии. – 2010 - №3(7). - с.76-82.

Об авторах:

Михаил Степанович Куприянов, д-р техн. наук, профессор (Санкт-Петербургский государственный электротехнический университет), Юлия Александровна Шичкина, канд. техн. наук, доцент (Братский государственный университет), E-mail: Strange.y@mail.ru