РЕШЕНИЕ ЗАДАЧИ КОММИВОЯЖЕРА НА МНОГОПРОЦЕССОРНЫХ СИСТЕМАХ С ОБЩЕЙ И РАСПРЕДЕЛЕННОЙ ПАМЯТЬЮ

А.Л. Игнатьев

Работа выполнена при поддержке РФФИ, грант 08-07-00072-а.

В этой статье мы рассмотрим решение широко известной задачи коммивояжера, которая ставится следующим образом. Пусть имеется n+1 город 0, 1, …, n; для каждой пары городов i и j задано «расстояние» $c_{ij} \geq 0$ между ними. Коммивояжер должен, выехав из начального города 0, объехать все города, посетив каждый из них ровно один раз, и вернуться в город 0. Необходимо найти такую последовательность посещения городов, при которой суммарная длина пути минимальна[1]. Асимметричной называется такая задача коммивояжера, где условие $c_{ij} = c_{ji}$ в общем случае не выполняется.

Многопроцессорные системы можно условно разделить на две большие группы: многопроцессорные системы с общей памятью и с распределенной. Параллельные системы с общей памятью характеризуются тем, что все вычислительные устройства (процессоры, ядра) разделяют общее адресное пространство. Это упрощает написание приложений для них — т.к. не надо заботиться о копировании данных между процессорами, с другой стороны появляется проблема синхронизации общего доступа к данным. Кроме того, такие системы плохо масштабируются.

Многопроцессорные системы с распределенной памятью состоят из набора вычислительных модулей, каждый из которых состоит из процессора и оперативной памяти, соединенных между собой высокоскоростной сетью. Такие системы гораздо лучше масштабируются, но написание приложений для них более трудоемко.

Методы решения задачи коммивояжера можно условно разделить на две категории: с доказательством и без доказательства оптимальности. Методы с доказательством оптимальности позволяют получить решение, отличающееся от оптимума не более чем на заданную величину. К этому классу можно отнести метод последовательного анализа вариантов, метод ветвей и границ и многие другие. Основным недостатком методов с доказательством оптимальности является их высокая вычислительная сложность. Методы без доказательства оптимальности часто называются приближенными методами. В приближенных методах решение задачи часто производится в два этапа: построение начального решения и улучшение начального решения. На первом этапе широко используются эвристические алгоритмы – алгоритмы, основанные на правдоподобных, но не обоснованных строго предположениях о свойствах оптимального решения задачи. На втором этапе используются алгоритмы локальной оптимизации, связанные с введением понятия окрестности; при этом можно использовать несколько алгоритмов данного типа, изменяя правила выбора окрестности в зависимости от полученных результатов.

Для решения задачи коммивояжера был выбран метод ветвей и границ, ознакомиться с другими параллельными методами решения задачи коммивояжера можно в [2]. Метод ветвей и границ носит переборный характер и основан на идее разделения множества решений задачи на подмножества (ветвление задачи), которые рано или поздно исключаются из рассмотрения с помощью процедуры оценки нижней границы. Процедура оценки нижней границы позволяет определить, содержится ли в рассматриваемом подмножестве решение заведомо лучшее, чем уже найденное. Такое значение называется рекордом. Подмножества исключается из дальнейшего рассмотрения (отсеиваются), когда нижняя граница подмножества превосходит либо совпадает с рекордом. Если в процессе решения задачи находится решение лучше найденного, оно становится рекордом. Хорошая процедура оценки нижней границы позволяет сократить число рассматриваемых подмножеств за счет их более быстрого отсева, и в ряде случаев приводит к значительному увеличению производительности.

Метод ветвей и границ превосходно поддается распараллеливанию: после ветвления множества решений, каждое полученное подмножество может обрабатываться независимо от остальных.

Известно, что задача коммивояжера принадлежит к классу NP-сложных задач и поэтому требует больших вычислительных ресурсов, при этом, мощностей обычных рабочих станций часто оказывается недостаточно. Поэтому для ее решения представляется целесообразным применение методов параллельных вычислений. Предлагаемый алгоритм был реализован с помощью библиотеки BNB-Solver[3], предназначенной для решения задач дискретной и глобальной оптимизации методом ветвей и границ на параллельных вычислительных системах с общей и распределенной памятью. Данная библиотека имеет гибкую программную инфраструктуру, основанную на шаблонах C++, обеспечивающую удобное подключение новых задач оптимизации. При этом требуется реализовать только проблемно-зависимые компоненты.

Для параллельного выполнения алгоритмов BNB-Solver использует схему «управляющий-рабочий». На начальном этапе генерируется некоторое количество подмножеств, которые затем пересылаются рабочим процессам для дальнейшей обработки. Для равномерной нагрузки процессоров используются централизованные алгоритмы балансировки.

В качестве системы с распределенной памятью использовался вычислительный кластер МВС100К[4], оснащенный 1980 4-ядерными процессорами Intel Xeon 5365 3 ГГц. В качестве системы с общей памятью была выбрана рабочая станция с 4-ядерным процессором Intel Core 2 Quad 9550 2.83 ГГц, 4 Гб оперативной памяти. Далее в случае распределенной памяти мы будем использовать понятие «процессор» для обозначения одного вычислительного устройства. В случае суперкомпьютера МВС100К, использующего многоядерные процессоры, под процессором будет пониматься 1 ядро. Такая терминология вполне допустима и даже желательна – т.к. в случае с распределенной памятью ядро рассматривается как отдельное вычислительное устройство.

Целью первой серии экспериментов было выявление наилучшей процедуры оценки нижних границ. Для проведения эксперимента мы реализовали 2 процедуры оценки нижних границ, основанных на приведении матриц[5] и задаче о назначении[1].

В ходе операции приведения матриц производится вычитание из каждого столбца его минимального элемента, аналогичные действия производится и для каждой строки. Сумма полученных минимальных элементов является нижней оценкой подмножества. В результате операции приведения новая матрица содержит как минимум по одному нулевому элементу в каждой строке и каждом столбце.

Задача о назначении формулируется следующим образом. Пусть имеется n работ и n кандидатов, причем назначение кандидата i на работу j требует затрат $c_{ij} \geq 0$. Требуется так распределить кандидатов по работам, чтобы (i) суммарные затраты были минимальными, при этом (ii)каждый кандидат может быть назначен только на одну работу и на каждую работу назначается только один кандидат[1].

Предположим, что матрица $x = (x_{ij})$ такая, что $x_{ij} = 1$, если работник i назначен на работу j, и $x_{ij} = 0$ в противном случае, является решением задачи о назначении. Из условия (ii) следует, что в каждой строке и в каждом столбце матрицы находится только одна единица. Поставим в соответствие матрице x n-вершинный ориентированный граф. В этом графе существует дуга (i, j), если $x_{ij} = 1$. Получим один или несколько циклов. Случай с одним циклом соответствует задаче коммивояжера. Действительно, задача о назначении является задачей коммивояжера с отсутствующим условием существования только одного цикла (условия посещения каждого города ровно один раз и возврата в начальный город выполняются только в случае одного цикла).

Мы сравнили оба варианта оценки на различных вариантах задачи коммивояжера из библиотеки TSPLIB[6]. Полученные результаты, представленные в Таб. 1, показывают, что процедура оценки, основанная на задаче о назначениях является более предпочтительной. Поэтому процедуру, основанную на задаче о назначениях, мы выбрали для проведения последующих экспериментов. Результаты экспериментов для процедур оценки, основанных на приведении матриц, можно посмотреть в [7].

Таб. 1. Время (сек.) решения задач с использованием различных процедур оценки нижней границы на 64 процессорах.

Задача	Ftv33	Ftv38	Ftv47	Ftv55
Приведение матриц	0.12	0.21	3.61	108.02
Задача о назначении	0.11	0.08	1.64	5.03

Результаты эксперимента для задачи Ftv90 (91 город) представлены в таб. 2. В первой колонке содержится число процессоров, использованное для решения задачи, во второй колонке – время решения задачи T, в третьей колонке – вычислительная сложность S, выражающаяся в числе ветвлений, и в четвертой колонке производительность P = S/T. Результаты эксперимента выявляют одну аномалию: время решения на 8 процессорах больше, чем на 4. Ее можно объяснить влиянием числа ветвлений. Полученные результаты показывают хорошую масштабируемость для реализации с распределенной памятью: производительность выросла в 83 раза на 96 процессорах по сравнению с 1 процессором.

Таб. 2. Результаты эксперимента для распределенной памяти для задачи Ftv90.

Число процессоров	Время (сек), T	Число ветвлений, S	Производительность, p
1	664.83	3299763	4963,30
2	96,90	1331230	13738,65
4	34,49	900641	26116,81
8	42,03	1939826	46153,93
16	26,14	2374063	90818,68
32	21,81	3225846	147886,08
48	8,52	2038415	239296,00
64	4,11	1325692	322588,43
96	2,9	1194724	411675,26

Процесс решения задачи методом ветвей и границ можно представить как обход дерева, в каждой вершине которого с помощью оценки нижней границы решается будет ли вершина ветвится дальше. Очевидно, что число ветвлений (вершин дерева) зависит от того в каком порядке обходится дерево. Можно очень быстро получить оптимальное решение и затем уже отсеять оставшиеся вершины. Поэтому число ветвлений зависит от числа процессоров т.к. число процессоров определяет каким образом будут распределены между ними вершины дерева для ветвления и, следовательно, в каком порядке будет произведен обход дерева. Чтобы нивелировать влияние порядка обхода, перед началом ветвлений мы присваиваем рекорду оптимальное решение. Тогда порядок обхода дерева становится несущественным, т.к. оптимальное решение уже найдено и будет производится только отсев вершин. Полученные результаты, представленные в Таб. 3, показывают монотонное возрастание производительности при увеличении числа процессоров. Потеря производительности в случае с 96 процессорами объясняется увеличившимися потерями в связи с межпроцессорными обменами данными.

Таб. 3. Результаты эксперимента для распределенной памяти для задачи Ftv90, с заданным рекордом равным оптимуму.

Число процессоров	Время (сек), T	Число ветвлений, S	Производительность, p
1	56,53	432703	7654,54
2	28,31	432701	15284,58
4	14,14	432697	30608,72
8	7,83	432689	55268,24
16	4,59	432673	94342,87
32	2,64	432641	164097,71
48	1,72	432609	251570,98
64	1,6	432577	270186,19
96	1,7	432513	254532,90

По результатам двух предыдущих экспериментов легко увидеть, что быстрое получение хорошего рекорда (в нашем случае равного оптимуму) многократно ускоряет решение задачи. Одним из способов улучшить значение существующего рекорда является применение алгоритмов локальной оптимизации. Для проведения эксперимента был реализован алгоритм обмена ближайших соседей. На каждом шаге этого алгоритма происходит обмен мест двух соседних городов в маршруте. Все доступные процессоры делятся на 2 группы: процессоры занимающиеся ветвлением, и эвристические процессоры. В случае, если удается получить новое улучшенное значение рекорда, новый рекорд рассылается процессорам, занятым ветвлением, что позволяет ускорить отсев подмножеств. Результаты проведенного эксперимента представлены в Таб. 4, где можно проследить, что увеличение числа процессоров отведенных под эвристику уменьшает число ветвлений, но в том же время уменьшает число процессоров занятых ветвлением. В рассмотренном примере оптимальным является распределение процессоров под эвристики и под ветвление поровну по 32. Ускорение за счет применения эвристики в этом случае составляет почти 30%.

Таб. 4. Результаты использования эвритических процессоов для задачи Ftv100 на 64 процессорах.

Число эвристических процессоров	Время (сек), T	Число ветвлений, S	Производительность, P
0	51.35	8035502	156475.48
8	58.08	7833920	134892.55
16	50.72	6702488	132136.56
32	36.28	5286109	145705.04
48	46.45	3994667	85996.70

Результаты эксперимента с общей памятью представлены в таб. 5. Реализация с общей памятью показывает худшую масштабируемость, нежели реализация с распределенной памятью. Этот факт демонстрирует проблему синхронизации потоков при использовании общих ресурсов.

Таб. 5. Результаты эксперимента для общей памяти для задачи Ftv47.

Число ядер	1	2	3	4
Вреся (сек)	07.07.09	3.98	2.67	2.23

В таб. 6. показано время решения различных задач реализацией с распределенной памятью на 64 процессорах. В восьми примерах взятых в [3] число городов колеблется от 45 до 444.

Таб. 6. Время решения различных задач

Задача	Число точек	Время (сек.)
Ftv44	45	0.09
Ftv90	91	4.1
Ftv100	101	104
Ftv110	111	888
Rbg323	324	277
Rbg358	359	553
Rbg403	404	1259
Rbg443	444	2523

ЛИТЕРАТУРА:

1. Сигал И.Х., Иванова А.П. Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы: Учеб. пособ. – 2-е изд., испр. и доп. – М.: ФИЗМАТЛИТ, 2007.
2. Игнатьев. А.Л. Сравнение различных методов решения задачи коммивояжера на многопроцессорных вычислительных машинах // Современные информационные технологии и ИТ-образование: III Межд. науч.-практ. конф., Москва, МГУ им. М.В. Ломоносова, 18-21 декабря 2006 г.: Сб. трудов под ред. В.А. Сухомлина, М.: МАКС Пресс. — 2008. — С. 509-513.
3. Посыпкин М.А. Архитектура и программная реализация библиотеки для решения задач оптимизации методом ветвей и границ на многопроцессорных вычислительных комплексах // Проблемы вычислений в распределенной среде: распределенные приложения, коммуникационные системы, математические методы и оптимизация. ИСА РАН. М.: КомКнига, 2006. С. 18-25.
4. Вычислительный кластер МВС100К. http://www.jscc.ru/hard/mvs100k.shtml
5. Литл. Дж., Мурти К., Суини Л., Кэрел К. Алгоритм для решения задачи коммивояжера // Экономика и математические методы. – 1965. – Т.1, В. 1. – С. 94-107.
6. Reinelt G. TSPLIB - A Traveling Salesman Problem Library // ORSA Journal on Computing. – 1991. – N. 3. – P. 376-384.
7. Игнатьев А.Л. Параллельные методы решения задачи коммивояжера // Труды 51-й научной конференции МФТИ «Современные проблемы фундаментальных и прикладных наук»: Часть IX. Инновации и высокие технологии. — М.: МФТИ. — 2008. — С. 4-6.