обработки запускается для всего изображения, чтобы обработать отрезки на границах между квадратами. Число частей, на которое делится изображение, удобно брать равным квадратному корню из меньшего из его длины и ширины.

На основании разработанных алгоритмов был создан программный комплекс, осуществляющий векторизацию монохромных изображений. Созданный комплекс имеет следующую функциональность:

- работа с изображениями различных графических форматов(JPEG, BMP, TIFF, PNG)
- редактирование растрового изображения (поворот на произвольный угол, масштабирование, удаление частей изображения)
- векторизуются монохромные изображения, поэтому предоставляется функциональность изменение глубины цвета и разрешения
- специальная предобработка растра, корректирующая дефекты сканирования (фильтрация шумов, соединение разъединенных пикселей)
- автоматическая постобработка векторов (удаление мелких объектов, обработка соединений)
- редактирование и ручная корректировка векторных объектов (копирование, перенос, стыковка объектов)
- полученное векторное изображение сохраняется в формат DXF различных версий
- имеется возможность задать сценарий векторизации, описывающий значения параметров алгоритмов и сохранить его в файл.

Использование интерактивных методов позволяет уменьшить долю участия человека в процессе векторизации, увеличить производительность работы и, соответственно, уменьшить затраты на оцифровку документов.

Литература

1. Абламейко С. В., Лагуновский Д. М. Обработка изображений: технология, методы, применения. Учебное пособие. – Мн.: Амалфея, 2000. – 304 с.

2. Dori Dov, Wenyin Liu. "Sparse Pixel Vectorization: An Algorithm and Its Performance Evaluation", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No.3, pp.202-215, 1999.

3. Sklansky J and Gonzalez V. Fast polygonal approximation of digitized curves. Pattern Recognition 1980; 12:327-331.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МЕТОДОВ ПОЛУЧЕНИЯ МНОГОМЕРНОГО ТАЙМИРОВАНИЯ ДЛЯ РАСПАРАЛЛЕЛИВАНИЯ АЛГОРИТМОВ

Е.В. Адуцкевич, А.О. Сикорский
Беларусь, г. Минск

Введение. При любом уровне развития вычислительной техники всегда существуют задачи, решение которых требует максимально возможной произ-

водительности. Как правило, это задачи, связанные с численным моделированием сложных систем, требующие высокой точности и скорости расчетов. Постоянно растущие требования к производительности вычислительных систем привели к использованию параллелизма в архитектуре компьютеров. Самые высокопроизводительные компьютеры в настоящее время являются параллельными. Однако сложность решения задач на таких компьютерах заключается в том, что алгоритмы для них также должны быть параллельными. В связи с этим возникла задача преобразования существующих последовательных алгоритмов таким образом, чтобы их можно было реализовать на компьютере параллельной архитектуры, т.е. задача распараллеливания алгоритмов. Для распараллеливания большого количества алгоритмов, многие из которых обладают сложной информационной структурой, возникла необходимость в разработке специального математического аппарата, т.е. формализованных методов распараллеливания алгоритмов [1–3].

В отделе параллельных вычислительных процессов Института математики НАН Беларуси проводятся исследования по разработке единого формализованного подхода к распараллеливанию алгоритмов. Отличительной особенностью подхода является совместное решение комплекса задач, возникающих при распараллеливании алгоритмов, что позволяет получать эффективные с точки зрения времени решения задачи параллельные алгоритмы. Поскольку разработанные методы являются формализованными и имеют строгое математическое обоснование, то их можно использовать для автоматизации процесса распараллеливания алгоритмов.

Доклад посвящен разработанному программному комплексу, позволяющему автоматизировать в интерактивном режиме некоторые этапы процесса распараллеливания программ.

Теоретические аспекты. Методы, реализуемые программным комплексом, применимы к алгоритмам, заданным аффинными гнездами циклов (т.е. последовательность вложенных циклов, в которых индексные выражения переменных и границы изменения параметров циклов алгоритма являются аффинными функциями от параметров циклов и внешних переменных). В качестве целевой архитектуры выбраны параллельные компьютеры с распределенной памятью. Вычислительные системы такого вида являются наиболее распространенными вследствие их невысокой стоимости и относительной простоты разработки.

Введем ряд обозначений. Пусть в гнезде циклов имеется K операторов S_β и используется L массивов a_l. Простые переменные будем считать массивами размерности 0. Пусть $V_\beta \subset \mathbf{Z}^{n_\beta}$ — область изменения параметров гнезда циклов для оператора S_β, где n_β — число циклов, окружающих оператор S_β; $W_l \subset \mathbf{Z}^{\nu_l}$ — область изменения индексов l-го массива, где ν_l — размерность l-го массива; $J \in V_\beta$ — вектор параметров цикла; $N \in \mathbf{Z}^e$ — вектор внешних переменных алгоритма, где e — число внешних переменных. Реализацию оператора S_β при кон-

кретном значении вектора J будем называть операцией и обозначать $S_\beta(J)$. Выполнение всех операций, зависящих от J, называется J-й итерацией.

В общем случае для распараллеливания алгоритма необходимо распределить его операции и данные между процессорами и установить порядок выполнения операций и обмена данными. При этом желательно добиваться улучшения характеристик получаемого параллельного алгоритма, влияющих на скорость решения задачи.

В качестве математического аппарата для распараллеливания алгоритмов удобно использовать векторные таймирующие функции, которые задают размещение операций алгоритма по процессорам и временной порядок выполнения операций. Функции осуществляют преобразование исходного гнезда циклов таким образом, что значения параметров нескольких внешних циклов можно интерпретировать как координаты процессоров, а оставшиеся внутренние циклы использовать для организации вычислений, выполняемых процессорами. Таким образом, задача распараллеливания алгоритма сводится к отысканию векторных таймирующих функций, удовлетворяющих некоторым ограничениям.

Для алгоритма, заданного аффинным гнездом циклов, можно выразить в формализованном виде следующие функции: функции $\overline{F}_{l,\beta,q}$, задающие выражения индексов элементов массивов, и функции $\overline{\Phi}_{\alpha,\beta}$, задающие зависимость операций алгоритма друг от друга. А именно: индексы элементов l-го массива, встречающегося в операторе S_β и относящегося к q-му входу элементов этого массива в оператор, выражаются функцией $\overline{F}_{l,\beta,q}: V_\beta \to W_l$; если существует зависимость зависимость операции $S_\beta(J)$ от операции $S_\alpha(I)$ (обозначение $S_\alpha(I) \to S_\beta(J)$), то верно выражение $I = \overline{\Phi}_{\alpha,\beta}(J)$, где $\overline{\Phi}_{\alpha,\beta}: V_{\alpha,\beta} \to V_\alpha$, $I \in V_\alpha$, $J \in V_{\alpha,\beta} \subseteq V_\beta$. Тем самым, указанные функции описывают всю необходимую информацию об алгоритме: как используются данные алгоритмом, и как взаимодействуют друг с другом операции. Формализованную таким образом информацию об алгоритме будем использовать при распараллеливании.

Для распараллеливания алгоритмов будем использовать аппарат векторных таймирующих функций, которые определяются следующим образом. Пусть $n = \max\limits_{1 \le \beta \le K} n_\beta$. Функции $\overline{t}^{(\beta)}: V_\beta \to \mathbf{Z}^n$, $1 \le \beta \le K$, ставят в соответствие операции $S_\beta(J)$ вектор $(t_1^{(\beta)}(J),\ldots,t_n^{(\beta)}(J))$ с целочисленными координатами.

$$t_\xi^{(\beta)}(J) = \tau^{(\beta,\xi)} J + b^{(\beta,\xi)} N + a_{\beta,\xi}, \quad 1 \le \beta \le K, 1 \le \xi \le n,$$
$$J \in V_\beta, \tau^{(\beta,\xi)} \in \mathbf{Z}^{n_\beta}, b^{(\beta,\xi)}, N \in \mathbf{Z}^s, a_{\beta,\xi} \in \mathbf{Z}. \qquad (1)$$

Функции $\overline{t}^{(\beta)}$ называются векторными таймирующими, если

$$\operatorname{rang} T^{(\beta)} = n_\beta, \quad 1 \le \beta \le K, \quad (2)$$

$$\overline{t}^{(\beta)}(J) \le_{lex} \overline{t}^{(\alpha)}(I), \quad J \in V_\beta, I \in V_\alpha, \; S_\alpha(I) \to S_\beta(J), \qquad (3)$$

где $T^{(\beta)}$ — матрица, строки которой составлены из векторов $\tau^{(\beta,1)},\ldots,\tau^{(\beta,n)}$, запись $\le_{lex}$ означает "лексикографически больше либо равно".

Выполнение условия (2) гарантирует невырожденность преобразования. Выполнение условия (3) гарантирует сохранение порядка выполнения информационно связанных операций алгоритма, что обеспечивает корректность распараллеливания алгоритма. Условие (3) называется условием сохранения зависимостей алгоритма и является важнейшим условием теории распараллеливания алгоритмов.

Набор векторных функций $\bar{t}^{(1)},...,\bar{t}^{(K)}$ называется многомерным таймированием. Набор функций $t_\xi^{(1)},...,t_\xi^{(K)}$ называется ξ-й координатой многомерного таймирования.

Помимо обязательных для корректного распараллеливания алгоритма условий (2) и (3), при поиске векторных таймирующих функций следует также учитывать желательные ограничения, удовлетворение которых позволяет получать более эффективную реализацию алгоритма. К таким условиям относятся минимизация количества коммуникаций между процессорами, улучшение локальности данных в памяти процессоров, организация эффективного обмена данными между процессорами.

Все необходимые и желательные ограничения на векторные таймирующие функции могут быть записаны в виде векторно-матричных неравенств на координаты многомерного таймирования. При этом вектор неизвестных содержит параметры искомых функций, остальные матрицы и векторы содержат известную информацию об алгоритме и составлены с использованием функций $\overline{F}_{l,\beta,q}$ и $\overline{\Phi}_{o,\beta}$.

Как правило, количество ограничений на искомые векторные таймирующие функции является большим. При этом, многие из ограничений не являются необходимыми и имеют различную значимость для распараллеливания алгоритма. Задачу поиска векторных таймирующих функций можно свести к решению серии из n оптимизационных задач, в результате чего последовательно, начиная с первой, определяются координаты векторных таймирующих функций $t_\xi^{(1)},...,t_\xi^{(K)}, \xi = 1,...,n$. Каждая оптимизационная задача позволяет учитывать большое количество ограничений на искомую координату многомерного таймирования. При этом значимость каждого ограничения может быть задана путем выбора весовых коэффициентов, что позволяет получать альтернативные решения в зависимости от требований решаемой задачи.

Программная реализация. На основании полученных теоретических результатов разработан программный комплекс, позволяющий для заданного алгоритма находить векторные таймирующие функции.

Программный комплекс представляет собой оконное приложение, посредством которого пользователь осуществляет ввод необходимой информации и просмотр результатов. На верхнем уровне приложение состоит из основного меню и панели с тремя закладками: «начальные данные», «промежуточные данные» и «результат». При помощи меню пользователь может либо сохранить уже введенные частично или полностью рассчитанные данные, либо загрузить

ранее сохраненные данные. Перейдем к описанию интерфейса и назначения каждой из трех закладок.

Закладка «Начальные данные». Панель существует для задания исходной информации об алгоритме, необходимой для применения формализованного метода распараллеливания, а именно информации о внешних переменных алгоритма, операциях и информационной зависимости между ними, массивах данных и способах их использования. На рисунке 1 изображен внешний вид приложения с выбранной первой закладкой.

Программный комплекс предоставляет две возможности построения ограничений на векторные таймирующие функции для сохранения зависимостей алгоритма: метод вершин [1, 2], использующий вершины многогранника $V_{\alpha,\beta}$, и общий способ [3], использующий информацию о соотношениях между координатами точек многогранника $V_{\alpha,\beta}$. В зависимости от выбранного способа, следует ввести необходимую информацию для составления условий сохранения зависимостей.

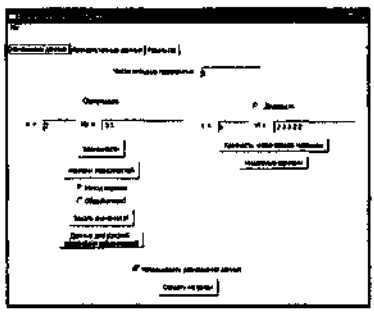

Рис. 1. Первая закладка программного комплекса

Программный комплекс также предоставляет возможность одновременного поиска векторных таймирующих функций и функций размещения данных алгоритма по процессорам.

Закладка «Промежуточные данные». После того, как вся необходимая информация об алгоритме введена, приложение создает матрицы и векторы, определяющие ограничения, которые следует учитывать при поиске векторных таймирующих функций.

Выбрав закладку «промежуточные данные» можно просмотреть все полученные матрицы и векторы, а также ввести дополнительные данные, требуемые для постановки оптимизационной задачи, решением которой является координата многомерного таймирования.

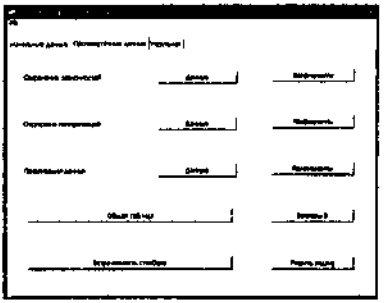

Рис. 2. Вторая закладка программного комплекса

Нажатием кнопки «решить задачу» запускается процесс решения поставленной оптимизационной задачи, результаты которой можно видеть на следующей закладке.

Закладка «Результат». Данная панель существует для отображения параметров очередной координаты многомерного таймирования, а также значения целевой функции. Кроме того, есть возможность просмотра всех значений переменных оптимизационной задачи (оптимизационная задача содержит ряд вспомогательных переменных, которые дают диагностическую информацию о полученном решении).

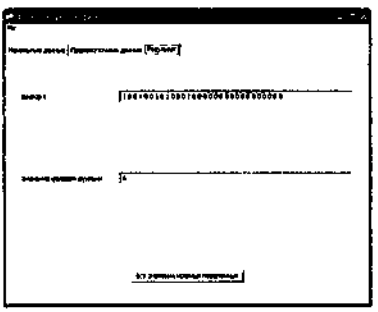

Рис. 3. Третья закладка программного комплекса

Поиск многомерного таймирования осуществляется покоординатно. Оптимизационная задача для поиска каждой координаты многомерного таймирования, начиная со второй, зависит от результатов решения предыдущих задач. Поэтому, работая с программным комплексом необходимо переходить между закладками «промежуточные данные» и «результат», задавая ограничения для оптимизационных задач, учитывая уже полученные результаты.

Таким образом, разработанный программный комплекс дает возможность поиска многомерного таймирования, задающего эффективную параллельную реализацию алгоритма. Методы распараллеливания алгоритмов, реализованные программным комплексом, имеют строгое математическое обоснование, что дает уверенность в полученных результатах. Программный комплекс автоматизирует часть этапов распараллеливания и является удобным в использовании.

Проведенная работа закладывает необходимый фундамент для дальнейшей программной реализации алгоритмов распараллеливания, а также доработки и усовершенствования разработанной автоматизированной системы.

Литература

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления // СПб.: БХВ-Петербург, 2002. – 608 с.

2. Адуцкевич Е.В., Лиходед Н.А., Соболевский П.И. Распараллеливание аффинных гнезд циклов при заданном распределении операций по процессорам // Весці НАН Беларусі. Сер. фіз.-мат. навук. – 2005. – 3. – С. 105–111.

3. Адуцкевич Е.В., Лиходед Н.А. Согласованное получение конвейерного параллелизма и распределения операций и данных между процессорами // Программирование. – 2006. – Т. 32, 3. – С. 54–65.

ШАБЛОН КЛАССА РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЕРА

М.П. Ревотюк, А.М. Полоневич
Беларусь, г. Минск

Задача коммивояжера, как известно, порождается во многих случаях оптимизации управления дискретными процессами, легко формулируется, но трудно решается. Объект рассмотрения - представление процедуры решения таких задач шаблонами функций и классов языка C++, допускающих специализацию на условия применения. Цель исследования - построение шаблона базового класса точного решения задачи коммивояжера методом ветвей и границ с минимизацией требующейся памяти и объема передаваемых данных при распараллеливании [1,2].

Задача коммивояжера здесь рассматривается в классической постановке и формулируется следующим образом: задана матрица расстояний $C = \|c_{ij},\ i, j = \overline{1,n}\|$ между любым из n городов, необходимо найти цикл минимальной длины однократного посещения каждого города (гамильтонов цикл минимальной длины).

Формальная модель задачи коммивояжера имеет вид:

$$\begin{cases} \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} c_{ij}\cdot x_{ij} \Rightarrow \min; \\ \sum\limits_{i=1}^{n} x_{ij} = 1; \quad \sum\limits_{j=1}^{n} x_{ij} = 1; \\ x_{ij} \geq 0, i, j = \overline{1,n}; \\ u_i - u_j + nx_{ij} \leq n-1,\ i = \overline{2,n},\ j = \overline{2,n},\ i \neq j. \end{cases} \qquad (1)$$