уменьшать толщину электрооптического материала или применять более эффективные электрооптические среды. Однако это связано с технологическими сложностями, например, явлением выгиба тонких электрооптических пластин, увеличением их хрупкости, необходимостью прецизионного сошлифовывания электрооптических пластин, значительной стоимостью материалов. В этой связи привлекательны, например, электрооптические полимеры, наносящиеся жидкостным распылением или центрифугированием в виде слоев толщиной в единицы десятки микрометров, обладающие большими электрооптическими коэффициентами (в десятки – сотни пм/В), хорошей пробивной прочностью (в единицы - десятки В/мкм). Важным этапом создания макетного образца дефлектора также является формирование тонких слаборасходящихся световых пучков на входе конструкции.

Список использованных источников

1.Патент 4,930,853 США МПК^{6} G02B 6/10. Electrooptic deflector / Grego G., заявитель и патентообладатель Centro Studi E Laboratori Telecomunicazioni S.P.A.: заявл. 05.06.1990 г. опубл. 24.07.1989 г

2.Патент 6,449,084 В1 США МПК^{7} G02F 1/29. Optical deflector /Yanping G.: заявитель и патентообладатель - Yanping G.; заявл. 09.05.2000 г. опубл. 10.09.2002 г.

3.Конойко А.И., Федоринчик М.П. Физические основы построения устройства оптической обработки сигналов. Мн : БГУИР, 2007. - С. 72.

4.Ярив А., Юх П. Оптические волны в кристаллах. - М.: Мир, 1987. - С. 616

5.Сонин А.С., Василевская А.С. Электрооптические кристаллы. М.: Атомиздат, 1971. - 327 с.

6.Кузьминов Ю.С. Сегнетоэлектрические кристаллы для управления лазерным излучением, 1982. - 400 с

7.Гурзадян Г.Г. Нелинейно-оптические кристаллы. Свойства и применение в квантовой электронике. М.: Радио и связь, 1991. - 160 с.

АЛГОРИТМ ПРАВИЛЬНОЙ НУМЕРАЦИИ ДВУХПОЛЮСНОГО ОРИЕНТИРОВАННОГО ГРАФА

Д.А. Попова-Коварцева

Самарский государственный аэрокосмический университет , г. Самара

Введение

Разработка параллельных программ представляется более сложной процедурой по сравнению с созданием аналогичных последовательных кодов программ. Отчасти это связано с архитектурными особенностями современных многопроцессорных вычислительных систем параллельной обработки. Учитывая широкую распространенность вычислительных систем с распределенной памятью, для организации информационного взаимодействия между процессорами часто используется интерфейс передачи данных MPI. Несмотря на широкие возможности, которые предоставляет для разработчиков параллельных программ библиотека MPI, ее применение сопровождается рядом трудностей, связанных с необходимостью рациональной организации управления потоками данных между процессорами и синхронизации вычислительных процессов. Неслучайно, что в последнее время разрабатываются высокоуровневые средства параллельного программирования, базирующиеся на MPI и облегчающие создание параллельных программ. К таким средствам относится и система PGRAPH [1, 2]

В системе PGRAPH модель параллельного алгоритма представляется в графическом виде, близкой к формализму представления блок-схем. Описание параллельных программ в виде граф-моделей позволяет реализовать образный, визуальный стиль представления алгоритмов программ и открывает широкие возможности их анализа с помощью методов теории графов.

Основные положения

Система PGRAPH предоставляет пользователю визуальные средства формирования графических образов разрабатываемых алгоритмов, представляемых в виде графа управления, в котором в вершинах граф-модели реализуются функциональные преобразования данных, а на дугах графа проверяются условия возможности перехода к следующей вершине графа. Различаются три типа дуг: последовательные, управляемые логическими условиями (предикатами); параллельные (безусловные) дуги, «открывающие» параллельную ветку программы и терминирующие дуги, завершающие выполнение параллельной ветки программы. Несколько упрощая ситуацию, условимся обозначать последовательные вершины графа квадратами, начало параллельной ветви программы – пустым кружочком, терминирующие вершины – «залитым» кружочком. Примеры граф-моделей программ приведены на рис. 1.

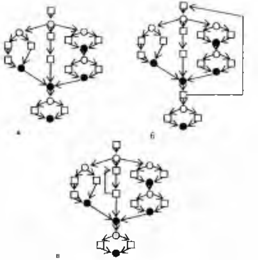

Рис. 1 Примеры граф-программ

По аналогии с последовательно-параллельными схемами соединения ненадежных элементов, используемых в теории надежности, можно ввести в рассмотрение «классические» последовательно-параллельные модели алгоритмов (см. рис.1,а). Первоначально определим понятие *классического Р-графа* следующим образом: граф, состоящий из одной вершины, является Р-графом; последовательное или параллельное соединение Р-графов является Р-графом; других Р-графов нет. Далее будем использовать алгебраическое определение ориентированного графа, заимствованное из работы [3], где под орграфом понимается пара $\vec{G}=(V,\alpha)$. Здесь V множество вершин орграфа, $\alpha \subseteq V \times V$ отношение на множестве V (пара $(u,v)\in\alpha$ называется дугой графа).

Р-граф можно охарактеризовать как направленных двухполюсных графов [3]. Граф $\vec{G}=(V,\alpha)$ называется направленным, если отношение α антисимметрично, т.е. в графе отсутствуют встречные дуги или, что одно и то же , в графе нет контуров длины 2. В качестве полюсов рассматриваются один источник (корень графа) и один сток (концевая вершина). Очевидно, что по построению Р-граф относится к классу направленных графов, поскольку в нем отсутствуют встречные дуги. Для определения принадлежности графа к классу направленных графов удобно использовать теорему 3.33 работы [3]. Теорема утверждает, что в орграфе $\vec{G}$ существует правильная нумерация вершин тогда и только тогда, когда $\vec{G}$ - бесконтурный граф. В данном случае говорят, что вершины графа *правильно* *пронумерованы.* $v_1, v_2, ..., v_n$, если из $(v_i, v_j) \in \alpha$ следует неравенство $i \le j$. Таким образом, проверка свойств орграфа на принадлежность классу направленных графов сводится к построению алгоритма правильной нумерации графа.

Однако, в практических приложениях при построении моделей параллельных алгоритмов в системе PGRAPH используются не только классические схемы Р-графа. В принципе орграф может содержать циклы (см. рис. 1, случаи б и в). В этом случае необходимо произвести предварительное расконтуривание графа, из которого удалить все встречные дуги. Теорема 3.36 [3] позволяет производить подобные операции без нарушения связности графа. Последняя утверждает, что минимальное расконтуривание связного графа не нарушает связности. Фактически расконтуривание превращает двухполюсный орграф в классический Р-граф.

Алгоритм нумерации

Введем понятие ранга вершины двухполюсного графа. *Рангом* $r(v)$ *вершины* v графа $\bar{G} = (V, \alpha)$ называется наибольшая из длин простых путей, начинающейся из корневой вершины и заканчивающейся в вершине v.

Алгоритм нумерации состоит из двух этапов. На первом этапе для всех вершин графа вычисляются ранги и производится при необходимости его расконтуривание. На втором - реализуется правильная нумерация вершин графа.

Вершиной ветвления будем называть вершину в которую входят только помеченные дуги и которая имеет по крайней мере две исходящие дуги. *Вершиной поглощения* назовем вершину в которую входят одна или несколько непомеченных дуг. Дополнительно введем два списка: *вершин ветвления* S_{out}, из которых исходят несколько дуг и *вершин поглощения* S_{in}, в которые входят несколько вершин, а также операции добавления и удаления из списков его элементов: $add(\cdot,\cdot)$, $del(\cdot,\cdot)$.

Шаг 1. Текущей вершине v_{i_k}, рассматриваемой на *k*-й итерации алгоритма, присваивается ранг: $r(v_{i_k}) = r((v_{i_{k-1}}, v_{i_k})) + 1$ - если она не вершина поглощения и $r(v_{i_k}) = \max\limits_{\alpha} r((v_\alpha, v_{i_k})) + 1$ - в противном случае (v_α вершины, входящие в вершину v_{i_k}). Если вершина v_{i_k} является вершиной ветвления, то выполняется операция пополнения списка S_{out}: $add(S_{out}, v_{i_k})$.

Шаг 2. Для вершины v_{i_k} выбирается непомеченная дуга перехода в следующую вершину. Выбранная дуга помечается рангом вершины v_{i_k} $r((v_{i_k}, v_{i_{k+1}})) = r(v_{i_k})$. Если все исходящие дуги помечены, то выполняется операция $del(S_{out} \cdot v_{i_k})$.

Шаг 3. Анализируем статус следующей вершины $v_{i_{k+1}}$.

Шаг 4. Если следующая $v_{i_{k+1}}$ не является вершиной поглощения, то в качестве текущей вершины рассматривается вершина $v_{i_{k+1}}$ и переходим к шагу 1. Иначе, при улови $v_{i_{k+1}} \notin S_{in}$, выполняется операция пополнения списка S_{in}: $add(S_{in} \cdot v_{i_{k+1}})$. Если не все дуги, входящие в вершину $v_{i_{k+1}}$ помечены, то из списка S_{out} выбирается последний элемент $v_j \in S_{out}$, который рассматривается как текущий. Переходим к шагу 2. Если все дуги, входящие в вершину $v_{i_{k+1}}$ помечены, то $del(S_{in} \cdot v_{i_{k+1}})$ и в качестве текущей вершины выбирается $v_{i_{k+1}}$. Переходим к шагу 1.

Шаг 5. Если список S_{out} пуст, а список S_{in} имеет элементы $S_{in} = [v_{i_1}, v_{i_2}, ..., v_{k_{i+1}}]$, то из графа удаляются дуги, входящие в вершину $v_{i_{k+1}}$, выполняется операция $del(S_{in} \cdot v_{i_{k+1}})$. В качестве текущей вершины выбирается вершина $v_{i_{k+1}}$. Переходим к шагу 1.

Список использованных источников

1.Коварцев А.Н. Автоматизация разработки и тестирования программных средств. - Самара: СГАУ. 1999. - 150 с.

2.Жидченко В.В., Коварцев А.Н. Моделирование синхронных параллельных вычислений при построении математических моделей сложных систем // Системный анализ и информационные технологии: Первая международная конференция САИТ-2005: Труды конференции. В 2т. Т.2. – М.: КомКнига, 2005. – С. 154-160.

3.Богомолов А.М., Салий В.Н. Алгебраические основы теории дискретных систем. – М.: Наука. 1997. - 368 с.