МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ИЗВЕСТИЯ ВЫСШИХ УЧЕБНЫХ ЗАВЕДЕНИЙ

ПРИБОРОСТРОЕНИЕ

ИЗДАНИЕ ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

Журнал издается с января 1958 г.

ШАБЛОН

УДК 621.397.01

Э.И. Ватутин, В.С. Титов

АЛГОРИТМИЧЕСКАЯ ОПТИМИЗАЦИЯ ПРОГРАММНОЙ РЕАЛИЗАЦИИ МЕТОДА ПАРАЛЛЕЛЬНО-ПОСЛЕДОВАТЕЛЬНОЙ ДЕКОМПОЗИЦИИ ГРАФ-СХЕМ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ

Приведено описание результатов профилирования и анализа узких мест программной реализации метода параллельно-последовательной декомпозиции граф-схем параллельных алгоритмов. В результате алгоритмической оптимизации вычислительные затраты на синтез разбиений сокращены в 30 раз.

Ключевые слова: система логического управления, проектирование логических мультиконтроллеров, разбиения, граф-схемы параллельных алгоритмов, алгоритмическая оптимизация.

При проектировании систем логического управления в базисе логических мультиконтроллеров [1, 2] возникает ряд многокритериальных задач дискретной оптимизации, одной из которых является задача поиска разбиения заданной параллельной граф-схемы алгоритма (ПарГСА). Ввиду невозможности отыскания точного решения за приемлемое время для граф-схем реальной размерности для решения указанной задачи разработан ряд эвристических методов, одним из которых является метод параллельно-последовательной декомпозиции [3, 4]. В данной работе приведены результаты профилирования его программной реали-

зации и последующей алгоритмической оптимизации его отдельных этапов. Данная работа является логическим продолжением работы [5].

В процессе расчета значений, записываемых в таблицы включений при распределении субсечений по блокам [6], среди ряда параметров весовой функции определяются приращения сложности сети межблочных связей ΔZ_α и интенсивности межблочных взаимодействий ΔZ_δ. Их расчет производится отталкиваясь от множества дуг межблочной передачи управления, а построение этого множества в свою очередь зависит от функции поиска дуги по номерам инцидентных ей вершин (например, требуется найти дугу, инцидентную вершинам a_3 и a_8, принадлежащим разным блокам разбиения: $a_3 \in A_1$, $a_8 \in A_3$). Схематично это можно представить в следующем виде:

```
S ≔ ∅;
for a_i ∈ S_1 do                  // O(N)
  for a_j ∈ S_2 do                // O(N)
    if v_k=(a_i, a_j) ∈ V then    // O(M) или O(log M)
      S ≔ S ⋃ { k };              // O(1)
```

Можно заметить, что приведенный фрагмент программы находит дуги передачи управления лишь между множествами S_1 и S_2, в то время как для получения всех дуг межблочной передачи управления для заданного разбиения необходимо добавить еще пару вложенных циклов: по $S_1 = A_1, A_2, ..., A_H$ и по $S_2 = A_1, A_2, ..., A_H$, $S_1 \neq S_2$. Однако при расчете приращений ΔZ_α и ΔZ_δ с использованием суммирования параметров $\beta(v_i)$ и $\delta(v_i)$ дуг [7] для включения (ρ_i, A_j) можно положить $S_1 = \bigcup_{k=1, k\neq j}^{H} A_k$ и $S_2 = \rho_i$, т.е. рассматривать только вновь добавляемые дуги межблочной передачи управления (рис. 1). (Более сложные способы расчета интенсивности межблочных взаимодействий (через клики на отношении совместимости дуг или с использованием дерева фрагментов) по-

требуют рассмотрения всех дуг межблочной передачи управления, а не только «новых».)

При реализации подпрограммы по алгоритму линейного (унарного) поиска время выполнения функции поиска дуги (проверка $v_k = \left(a_i, a_j\right) \in V$ с выдачей номера дуги *k*) занимает 25% от общего времени построения разбиения. С целью уменьшения вычислительной сложности функции принято решение о замене линейного поиска на бинарный (двоичный). Как известно [8], бинарный поиск сокращает число выполняемых действий с $O(n)$ до $O(\log n)$, где *n* в данном случае – число дуг в обрабатываемом графе, однако при этом требует, чтобы исходный массив был отсортирован. В процессе сортировки массива дуг полагается, что $v_i < v_j$ в том случае, если

$$\left[\left(a^{нач}\left(v_i\right) = a^{нач}\left(v_j\right)\right) \wedge \left(a^{кон}\left(v_i\right) < a^{кон}\left(v_j\right)\right)\right] \vee$$
$$\vee\left[\left(a^{нач}\left(v_i\right) \neq a^{нач}\left(v_j\right)\right) \wedge \left(a^{нач}\left(v_i\right) < a^{нач}\left(v_j\right)\right)\right],$$

при этом начальные и конечные вершины дуг v_i и v_j сравниваются по их номерам. Другими словами, порядок дуг в массиве при сортировке определяется соотношением номеров начальных вершин или, при их совпадении, – конечных вершин. Пример исходного массива дуг и результат его сортировки приведен на рис. 2.

Процедура сортировки выполняется 3 раза (для исходной ПарГСА, ПарГСА после преобразований и скелетного графа), что занимает 0,05% от общего времени построения разбиения (ее влияние почти не заметно на фоне других преобразований). Процедура поиска дуги выполняется более 300000 раз, а ее вклад в общее время построения разбиения после замены алгоритма линейного поиска на бинарный сокращается с 25% до 16%. В результате оптимизации алгоритма поиска асимптотическая временная сложность алгоритма выделения «новых» дуг межблочной передачи управления уменьшается с $O\left(N^2 M\right)$ до

$O\left(N^2 \log M\right)$, где N – число вершин в граф-схеме алгоритма, M – число дуг. Практический выигрыш во времени синтеза разбиения заключается в его снижении со 183 мс до 156 мс (на 14,8%).

Несмотря на проведенную оптимизацию и полученное сокращение времени синтеза разбиений время выполнения функции поиска дуги по прежнему вносит достаточно большой вклад. Ввиду невозможности ее дальнейшей оптимизации вызывает интерес попытка снижения числа ее вызовов, т.е. оптимизация алгоритмов работы подпрограмм верхнего уровня.

Нахождение множества дуг межблочной передачи управления можно выполнить путем сопоставления вершинам алгоритма цветов. Цвет 0 будем приписывать вершинам, еще не размещенным по субсечениям и вершинам блока A_j; цвет 1 – вершинам блоков разбиения, за исключением A_j; цвет 2 – вершинам субсечения ρ_i. При этом дугу будем считать дугой межблочной передачи управления в том случае, если она соединяет две разноцветные вершины ненулевого цвета (рис. 3).

В случае необходимости выделения всех дуг межблочной передачи управления распределение цветов меняется: нераспределенные по блокам вершины окрашиваются в цвет 0, вершины блока разбиения A_k, $k = \overline{1, H}$ (H – число блоков в разбиении) – в цвет k, а вершины субсечения ρ_i – в цвет j.

Схематично процедуру выделения дуг межблочной передачи управления можно представить следующим образом:

```
// Получение множества вершин блоков разбиения без Aj - O(H*N)
S1 ≔ ∅;
for k ≔ 1 to H do
  if k ≠ j then
    S1 ≔ S1 ⋃ Ak;

// Раскраска вершин - O(N)
for k ≔ 1 to N do
  case ak of
    ak∈S1: цвет[ak] ≔ 1;
    ak∈ρi: цвет[ak] ≔ 2;
```

```
  else
    цвет[a_k] := 0;
  end;

// Построение множества дуг межблочной передачи управления – O(M)
S := ∅;
for k := 1 to M do
  if (цвет[a^нач(v_k)] ≠ 0) ∧
     (цвет[a^кон(v_k)] ≠ 0) ∧
     (цвет[a^нач(v_k)] ≠ цвет[a^кон(v_k)]) then
       S := S ⋃ { k };
```

Асимптотическая временная сложность предложенного алгоритма составляет $O(HN+N+M) \simeq O(HN+M)$, практический выигрыш от его использования заключается в дополнительном уменьшении временных затрат на построение разбиения с 156 мс до 108 мс (на 30,9%).

При анализе программной реализации параллельно-последовательного метода [4] ввиду реализации нескольких способов подсчета интенсивности межблочных взаимодействий также имел место ряд повторяющихся действий, устранение которых позволило дополнительно сократить время синтеза разбиения с 108 мс до 46,6 мс (в 2,3 раза).

После выполнения описанных выше оптимизаций интегральные затраты времени на выполнение различных операций с множествами (объединение, пересечение, проверка принадлежности и др.) составляют 53,4% от общего времени синтеза разбиений. Первоначально для операций с множествами был использован класс TBits, входящий в состав библиотеки VCL. В ходе алгоритмической оптимизации был произведен отказ от его использования в пользу разработанного класса TSet [9], реализующего необходимые операции с множествами с использованием команд SIMD-расширений, поддерживаемых процессором, на языке ассемблера. В результате использования разработанного класса временные затраты дополнительно сокращаются с 46,6 мс до 31,9 мс (на 31,5%).

Построение множества смежных сечений $\Re$ [2] по имеющемуся базовому сечению $\Omega_{\max}$ заключается в последовательном нахождении *u*- и *d*-сечений. Каждое новое сечение Ω_i формируется в результате отыскания множества S^u

или S^d выражений системы Ξ, удовлетворяющих условиям *u*- или *d*-подстановки, и серии последующих подстановок.

Для отыскания очередного сечения в ходе построения множеств S^u или S^d выполняется N_Ξ операций проверки конструктивного включения *R*-выражений, т.е. общее число операций в ходе построения всех сечений составляет $N_\Xi N_\Omega$. Можно заметить, что каждое выражение S_i, $i = \overline{1, N_\Xi}$ в ходе построения множества смежных сечений используется однократно, чем можно воспользоваться с целью уменьшения числа проверок путем использования множества доступных выражений S^+:

```
ℜ ≔ ∅;
S⁺ ≔ Ξ;

// Получение u-сечений
Ω ≔ Ω_max;
repeat
  S^u ≔ ∅;
  for S_i ∈ S⁺ do
    if (S_i.R_2[⊆]Ω) then begin
      S^u ≔ S^u ⋃ { S_i };
      S⁺ ≔ S⁺ \ { S_i };
    end;

  for S_i ∈ S^u do
    Выполнить_подстановку(Ω, S_i.R_2, S_i.R_1);

  ℜ ≔ ℜ ⋃ { Ω };
until (Ω ≠ a_нач);

// Получение d-сечений
Ω ≔ Ω_max;
repeat
  S^d ≔ ∅;
  for S_i ∈ S⁺ do
    if (S_i.R_1[⊆]Ω) then begin
      S^d ≔ S^d ⋃ { S_i };
      S⁺ ≔ S⁺ \ { S_i };
    end;

  for S_i ∈ S^d do
    Выполнить_подстановку(Ω, S_i.R_1, S_i.R_2);

  ℜ ≔ ℜ ⋃ { Ω };
until (Ω ≠ a_кон);
```

Число проверок отношения нестрого включения *R*-выражений сокращается до N_{Ξ}, что ведет к сокращению общего времени поиска разбиения с 31,9 мс до 24,4 мс (на 16,5%).

В работе [10] предложены два необходимых условия отсутствия *r*-изоморфизма у пары *R*-выражений A^R и B^R, для которых производится проверка отношения нестрогого включения $A^R \left[\subseteq\right] B^R$. Первое из них основано на отыскании в составе представления *R*-выражения A^R в виде дерева такого набора листьев, для которого не найден эквивалентный набор листьев в представлении *R*-выражения B^R в виде дерева; второе – на наличии более одного набора листьев в отношении неполной эквивалентности. Оба необходимых условия проверяются в ходе попарного сопоставления наборов листьев, сокращение времени проверки *r*-изоморфизма достигается за счет более раннего выяснения того, что пара *R*-выражений не *r*-изоморфны. Добавление проверки необходимых условий в программную реализацию параллельно-последовательного метода в совокупности со сравнением числа узлов в рассматриваемых деревьях ведет к уменьшению общего времени синтеза разбиения с 27,4 мс до 24,2 мс (на 11,7%).

В работах [10, 11] предложен итеративный алгоритм выяснения отношения нестрогого включения *R*-выражений (точнее, две его модификации, ориентированные на аппаратную и программную реализацию соответственно) на основе распространения отношения эквивалентности в представлении *R*-выражений в виде деревьев снизу-вверх (от наборов листьев к корню). Как показало тестирование, его использование вместо алгоритма рекуррентного сравнения поддеревьев с совпадающими оценками [12] приводит к уменьшению времени синтеза разбиения с 24,2 мс до 21,4 мс (на 11,6%).

Выделение базового сечения отталкиваясь от системы выражений Ξ заключается в попарном выборе выражений S_i и S_j, $i, j = \overline{1, N_{\Xi}}$ с целью попытки выполнения операции *u*- или *d*-постановки. После выполнения подстановки

число N_Ξ выражений системы Ξ сокращается на один, для оставшихся выражений снова производится попарное сопоставление и т.д. Указанные действия продолжаются до тех пор, пока число выражений в системе Ξ не станет равно двум. Число операций проверки отношения нестрогого включения при этом составляет в худшем случае величину $O\left(\underbrace{N_\Xi\left(N_\Xi-1\right)}_{\substack{\text{попарные}\\ \text{сопоставления}}}\underbrace{\left(N_\Xi-2\right)}_{\substack{\text{число}\\ \text{подстановок}}}\right)\simeq O\left(N_\Xi^3\right)$. Если на *i*-м шаге работы алгоритма для пары выражений S_i и S_j невозможно выполнить ни *u*-, ни *d*-подстановку и они не участвуют на *i*-м шаге в подстановке с использованием других выражений, то на $\left(i+1\right)$-м шаге подстановка для пары выражений S_i и S_j по прежнему не будет невозможна. Указанной особенностью можно воспользоваться путем сохранения в специальной матрице M_Ξ информации о предыдущих неудачных попытках выполнения подстановки с целью экономии вычислительного времени. При этом на каждом шаге алгоритма кроме первого вместо $N_\Xi\left(N_\Xi-1\right)$ производится $N_\Xi-1$ сопоставлений для выражения S_k, полученного в ходе выполнения подстановки на *i*-м шаге, что сокращает число операций подстановки до $O\left(\left(N_\Xi-1\right)\left(N_\Xi-2\right)\right)\simeq O\left(N_\Xi^2\right)$, а общее время синтеза разбиения с 21,4 мс до 19,4 мс (на 9,4%).

Таким образом, описанные шаги программной алгоритмической оптимизации позволяют сократить затраты вычислительного времени с 183 мс до 21,4 мс, т.е. в 8,6 раза (в 30 раз с учетом оптимизаций, описанных в [5]), что позволило существенную экономию вычислительного времени при выполнении эксперимента по сравнению методов синтеза разбиений с использованием добровольных распределенных вычислений [13].

СПИСОК ЛИТЕРАТУРЫ

1. Архитектура параллельных логических мультиконтроллеров / *Емельянов С.Г., Зотов И.В., Титов В.С.* М: Высшая школа, 2009. 233 с.

2. *Ватутин Э.И.* Проектирование логических мультиконтроллеров. Синтез разбиений параллельных граф-схем алгоритмов. Saarbrücken: Lambert Academic Publishing, 2011. 292 с.
3. *Ватутин Э.И., Зотов И.В.* Метод формирования субоптимальных разбиений параллельных управляющих алгоритмов // Параллельные вычисления и задачи управления (PACO'04). М.: Институт проблем управления им. В.А. Трапезникова РАН, 2004. С. 884–917.
4. *Ватутин Э.И., Зотов И.В.* Параллельно-последовательный метод формирования субоптимальных разбиений параллельных управляющих алгоритмов // Свидетельство об официальной регистрации программы для ЭВМ № 2005613091 от 28.11.05.
5. *Ватутин Э.И.* Анализ эффективности и программная оптимизация методов синтеза разбиений параллельных алгоритмов логического управления в среде PAE // Известия ЮЗГУ. Серия «Управление, вычислительная техника, информатика. Медицинское приборостроение». № 2. Ч. 1. С. 191–195.
6. *Ватутин Э.И., Волобуев С.В., Зотов И.В.* Комплексный сравнительный анализ качества разбиений при синтезе логических мультиконтроллеров в условиях присутствия технологических ограничений // Труды четвертой международной конференции «Параллельные вычисления и задачи управления» PACO'08. М.: Институт проблем управления им. В.А. Трапезникова РАН, 2008. С. 643–685.
7. *Ватутин Э.И.* Проблема оценки интенсивности межблочного взаимодействия в задаче нахождения субоптимальных разбиений параллельных управляющих алгоритмов / III международный студенческий фестиваль «Образование, наука, производство». Белгород, 2006.
8. http://ru.wikipedia.org/wiki/Двоичный_поиск
9. *Ватутин Э.И.* Библиотека классов обработки множеств с SIMD-оптимизацией // Свидетельство об официальной регистрации программы для ЭВМ № 2007614221 от 03.08.07.
10. *Ватутин Э.И., Зотов И.В., Титов В.С.* Алгоритм и устройство выявления изоморфных вхождений *R*-выражений при построении множества сечений параллельных алгоритмов логического управления // Известия вузов. Приборостроение. 2009. Т. 52, № 2. С. 37–45.
11. *Ватутин Э.И., Зотов И.В., Титов В.С.* Выявление изоморфных вхождений R-выражений при построении множества сечений параллельных алгоритмов логического управления // Информационно-измерительные и управляющие системы. № 11, Т. 7. М.: «Радиотехника», 2009. С. 49–56.
12. Поиск базового сечения в задаче разбиения параллельных алгоритмов / *Ватутин Э.И., Зотов И.В.*; КГТУ. Курск, 2003. 30 с. Рук. деп. в ВИНИТИ 24.11.03 № 2036-В2003.
13. *Ватутин Э.И., Титов В.С.* Использование добровольных распределенных вычислений на платформе BOINC для анализа качества разбиений граф-схем параллельных алгоритмов // Параллельные вычисления и задачи управления (PACO'12). М.: ИПУ РАН, 2012. С. 37–54.

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 10.01.2013

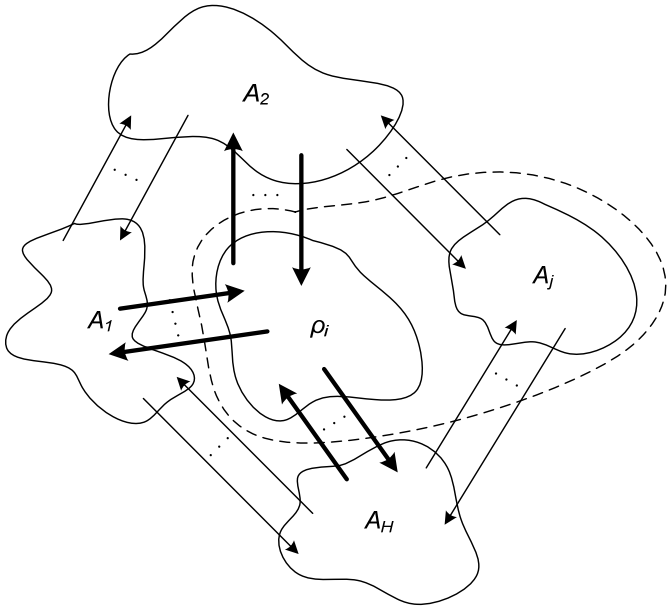

Рис. 1. Иллюстрация к расчету приращений ΔZ_α и ΔZ_δ. Пунктиром показан блок разбиения A_j после включения в него субсечения ρ_i, стрелками изображены дуги межблочной передачи управления, жирными стрелками показаны «новые» дуги передачи управления, возникающие в случае включения субсечения ρ_i в блок A_j

а)

5	0	3	6	6	9	8	1
1	4	2	5	2	6	6	9

б)

0	1	3	5	6	6	8	9
4	9	2	1	2	5	6	6

Рис. 2. Массив дуг до (а) и после (б) сортировки

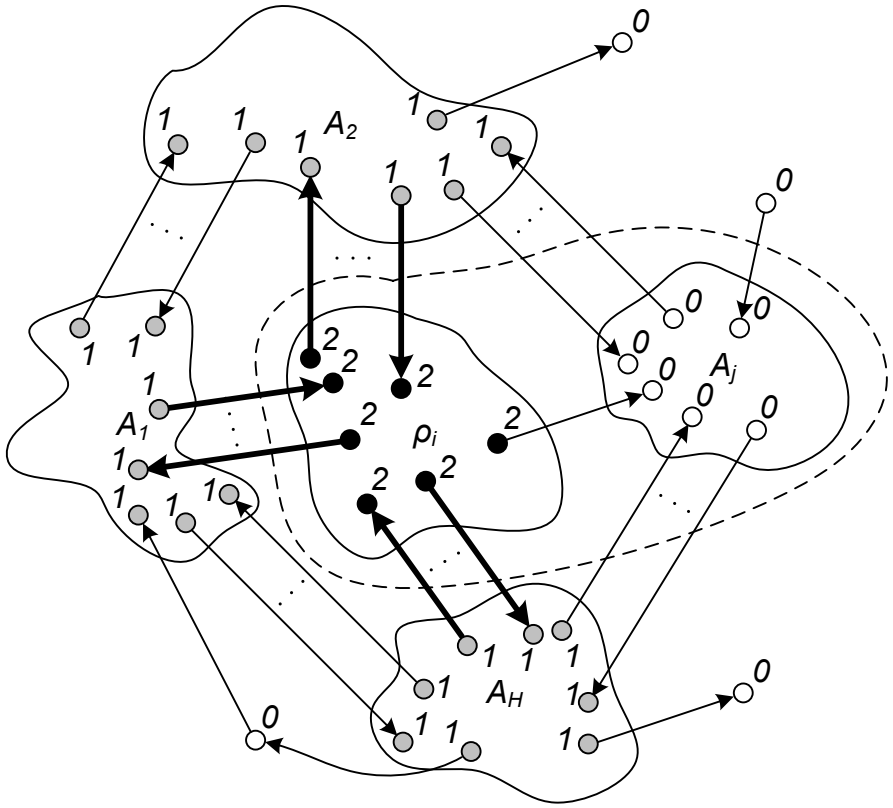

Рис. 3. Раскраска вершин в зависимости от принадлежности к блокам разбиения или субсечению