ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

УДК 621.397.01

Э. И. Ватутин, В. С. Титов

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

Описаны результаты 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 на базе логических мультиконтроллеров [1, 2] возникает ряд многокритериальных задач дискретной оптимизации, одной из которых является задача поиска разбиения заданной параллельной граф-схемы алгоритма. Ввиду невозможности отыскания точного решения в течение приемлемого времени для граф-схем реальной размерности с целью решения указанной задачи разработан ряд эвристических методов, в частности, метод параллельно-последовательной декомпозиции [3, 4]. В настоящей статье, являющейся продолжением работы [5], приведены результаты профилирования его программной реализации и последующей алгоритмической оптимизации его отдельных этапов.

В процессе расчета значений для таблиц включений при распределении субсечений по блокам [6], в частности, определяются параметры весовой функции приращения сложности сети межблочных связей ΔZ_α и интенсивности межблочных взаимодействий ΔZ_δ. Расчет этих параметров производится на основе множества дуг межблочной передачи управления, особенности построения которого, в свою очередь, зависят от способа 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 в приведенном фрагменте программы дуги передачи управления находятся лишь между множествами S_1 и S_2, в то время как с целью 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 дуг $\beta(v_i)$ и $\delta(v_i)$ [7] для включения (ρ_i, A_j) можно положить

$$S_1 = \bigcup_{k=1,\, k\neq j}^{H} A_k \text{ и } S_2 = \rho_i,$$

т.е. рассматривать только вновь добавляемые дуги межблочной передачи управления (рис. 1). На рисунке пунктиром показан блок разбиения A_j после включения в него субсечения ρ_i, стрелками изображены дуги межблочной передачи управления, жирными стрелками — „новые“ дуги передачи управления, возникающие в случае включения субсечения ρ_i в блок A_j. (Более сложные способы расчета интенсивности межблочных взаимодействий — через выделение отношений совместимости дуг или с использованием дерева фрагментов — потребуют рассмотрения всех дуг межблочной передачи управления, а не только „новых“.)

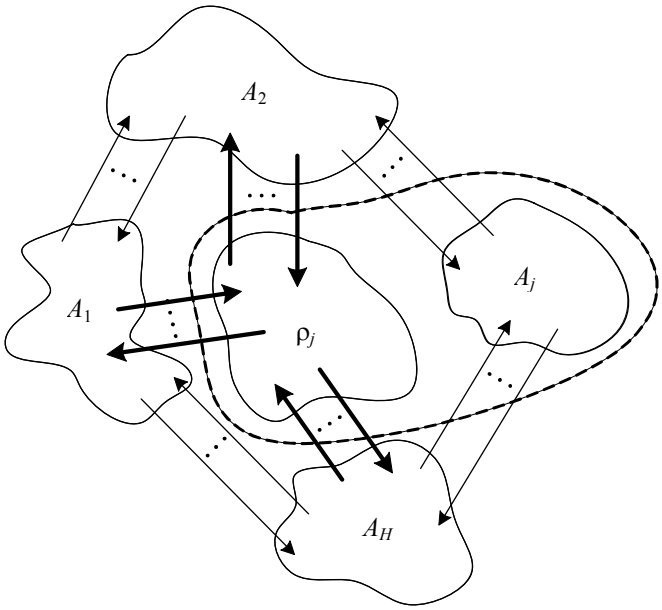

Рис. 1

При реализации подпрограммы по алгоритму линейного (унарного) поиска время поиска дуги (проверка $v_k = (a_i, a_j) \in V$ с выдачей номера дуги k) занимает 25 % от общего времени построения разбиения. С целью уменьшения вычислительной сложности операции линейный поиск заменен бинарным (двоичным). Как известно [8], при бинарном поиске число выполняемых действий сокращается с $O(n)$ до $O(\log n)$, где n в данном случае — число дуг в обрабатываемом графе, однако требуется, чтобы исходный массив был отсортирован. При сортировке массива дуг полагается $v_i < v_j$ в том случае, если

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

сравниваются номера начальных и конечных дуг v_i и v_j. Другими словами, порядок дуг в массиве при сортировке определяется соотношением номеров начальных вершин или, при их совпадении, конечных вершин. На рис. 2 приведены пример исходного массива дуг (*а*) и результаты его сортировки (*б*).

Процедура сортировки выполняется три раза (для исходной параллельной граф-схемы алгоритма, после ее преобразований и для скелетного графа), это занимает всего 0,05 % от общего времени построения разбиения. Процедура поиска дуги выполняется более 300 000 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 $O\left(N^2M\right)$ 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 — со 183 до 156 мс (снижение на 14,8 %).

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

Несмотря на проведенную оптимизацию и сокращение времени синтеза разбиений функция поиска дуги по-прежнему занимает достаточно большую часть времени. Ввиду невозможности ее дальнейшей оптимизации представляется перспективной попытка снижения числа вызовов, т.е. оптимизация алгоритмов работы подпрограмм верхнего уровня.

Множество дуг межблочной передачи управления можно найти путем сопоставления вершинам алгоритма цветов. Значение цвета 0 будем приписывать вершинам, еще не размещенным по субсечениям и вершинам блока A_j; 1 — вершинам блоков разбиения, за исключением A_j; 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).

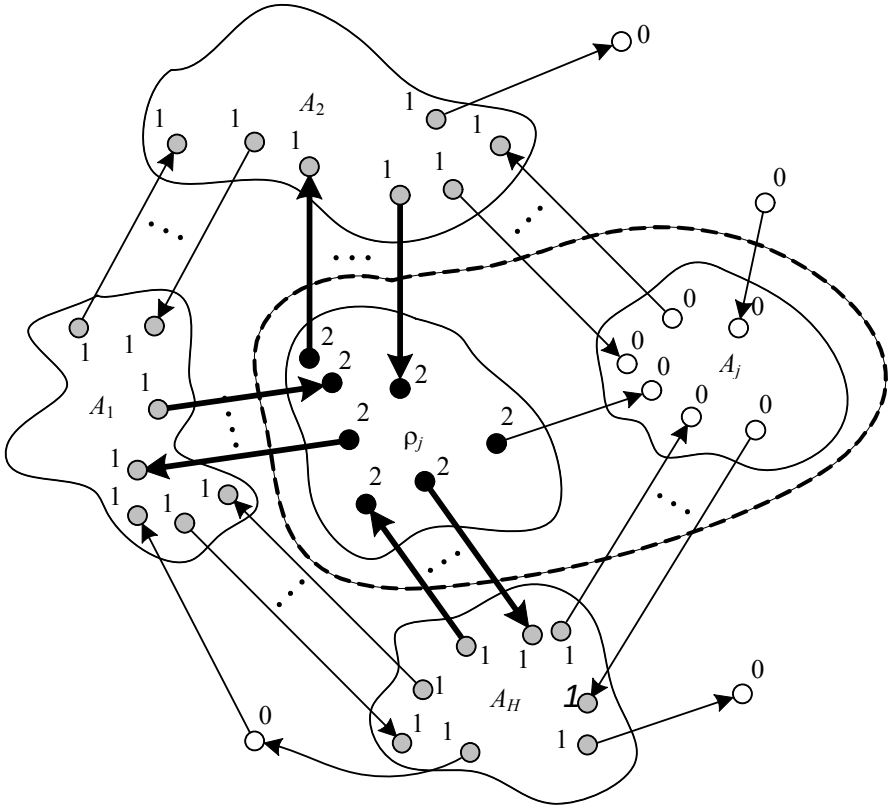

Рис. 3

Если необходимо выделять все дуги межблочной передачи управления, распределение цветов меняется: не распределенные 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[ak] := 0;
  end;

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

Асимптотическая временная сложность предложенного алгоритма составляет $O(HN + N + M) \simeq O(HN + M)$, практическое преимущество от его использования заключается в дополнительном уменьшении времени построения разбиения с 156 до 108 мс (на 30,9 %).

В программной реализации параллельно-последовательного метода [4] вследствие применения нескольких способов подсчета интенсивности межблочных взаимодействий ряд действий повторялся. Устранение таких повторов позволило дополнительно сократить время синтеза разбиения с 108 до 46,6 мс (в 2,3 раза).

После описанной оптимизации суммарные затраты времени на выполнение различных операций с множествами (объединение, пересечение, проверка принадлежности и др.) составляют 53,4 % от общего времени синтеза разбиений. Для операций с множествами использован разработанный класс TSet [9], выполняющий необходимые операции с использованием команд SIMD-расширений, поддерживаемых процессором, на языке ассемблер. Благодаря использованию разработанного класса временные затраты дополнительно сокращаются с 46,6 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 выра-

жение 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 := Ωmax;
repeat
  Su := ∅;
  for Si ∈ S+ do
    if (Si.R2[⊆]Ω) then begin
      Su := Su ⋃ { Si };
      S+ := S+ \ { Si };
    end;

  for Si ∈ Su do
    Выполнить_подстановку(Ω, Si.R2, Si.R1);

  ℜ := ℜ ⋃ { Ω };
until (Ω ≠ aнач);

// Получение d-сечений
Ω := Ωmax;
repeat
  Sd := ∅;
  for Si ∈ S+ do
    if (Si.R1[⊆]Ω) then begin
      Sd := Sd ⋃ { Si };
      S+ := S+ \ { Si };
    end;

  for Si ∈ Sd do
    Выполнить_подстановку(Ω, Si.R1, Si.R2);

  ℜ := ℜ ⋃ { Ω };
until (Ω ≠ aкон);
```

Число проверок отношения нестрогого включения *R*-выражений сокращается до N_Ξ, что ведет к сокращению общего времени поиска разбиения с 31,9 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 $A^R[\subseteq]B^R$:

1) в составе представления *R*-выражения A^R в виде дерева отыскивается такой набор листьев, для которого не найден эквивалентный набор листьев в представлении *R*-выражения B^R в виде дерева;

2) наличие 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 раннего выявления отсутствия

r-изоморфности пары *R*-выражений. 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 до 24,2 мс (на 11,7 %).

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

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

$$O\left(\underbrace{N_\Xi\left(N_\Xi-1\right)}_{\substack{\text{попарные}\\ \text{сопоставления}}}\underbrace{\left(N_\Xi-2\right)}_{\substack{\text{число}\\ \text{подстановок}}}\right)\simeq O\left(N_\Xi^3\right).$$

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 $(i+1)$-м шаге подстановка для пары выражений S_i и S_j по-прежнему не будет невозможна. Указанной особенностью можно воспользоваться, сохранив в специальной матрице M_Ξ информацию о предыдущих неудачных попытках подстановки для 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 до 19,4 мс (на 9,4 %).

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

Работа выполнена в рамках программы „Научные и научно-педагогические кадры инновационной России на 2009—2013 годы“ (проект 14.B37.21.0598 „Теоретические основы и методы использования распределенных и высокопроизводительных вычислительных систем для решения дискретных оптимизационных задач“).

СПИСОК ЛИТЕРАТУРЫ

1. *Емельянов С. Г., Зотов И. В., Титов В. С.* Архитектура параллельных логических мультиконтроллеров. М.: Высш. 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.: Ин-т 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 // Изв. 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 // Тр. 4-й Междунар. конф. „Параллельные вычисления и задачи управления" PACO'08. М.: Ин-т 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 Междунар. студенческий фестиваль „Образование, наука, производство". Белгород, 2006.

8. [Электронный ресурс]: <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 // Изв. 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. 2009. Т. 7, № 11. С. 49—56.

12. *Ватутин Э. И., Зотов И. В.* Поиск базового сечения в задаче разбиения параллельных алгоритмов. Курск: КГТУ, 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.

Сведения об авторах

Эдуард Игоревич Ватутин — канд. техн. наук, доцент; Юго-Западный государственный университет, кафедра вычислительной техники, Курск; E-mail: evatutin@rambler.ru

Виталий Семенович Титов — д-р техн. наук, профессор; Юго-Западный государственный университет, кафедра вычислительной техники, Курск; заведующий кафедрой; E-mail: titov-kstu @rambler.ru

Рекомендована Юго-Западным государственным университетом

Поступила в редакцию 18.02.13 г.