СИБИРСКОЕ ОТДЕЛЕНИЕ

А В Т О М Е Т Р И Я

№ 3 1999

УДК 681.322

С. М. Ачасова

(Новосибирск)

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ В РАСПРЕДЕЛЕННОЙ АРХИТЕКТУРЕ

В распределенную архитектуру, базирующуюся на Алгоритме параллельных подстановок, вкладываются быстрые параллельные алгоритмы решения двух задач комбинаторной оптимизации: о максимальном независимом множестве и о минимальном взвешенном вершинном покрытии графа. Первый алгоритм основан на методе Монте – Карло, второй использует двойственность двух оптимизационных задач, одну из которых предстоит решить. Топология связей между элементарными процессорами в синтезируемой архитектуре повторяет топологию связей вершин в графе.

Введение. Известно, что не всегда последовательные алгоритмы решения задач комбинаторной оптимизации могут быть конвертированы в быстрые параллельные алгоритмы. Часто упоминаемым примером такой задачи является задача о максимальном независимом множестве. Дан неориентированный граф $G=(V, E)$, V – множество вершин, $|V|=n$, E – множество ребер, $|E|=m$. Найти подмножество вершин $I \subseteq V$, обладающее следующими свойствами: 1) вершины в I попарно не связаны ребрами, 2) добавление в I любой новой вершины графа лишает множество I свойства 1. Подмножество вершин графа, обладающее перечисленными свойствами, называется максимальным независимым множеством. Очевидный последовательный алгоритм решения этой задачи: в каждой итерации к множеству вершин графа I (первоначально пустому) добавляется вершина i, $i=1,\ldots,n$, если она не является смежной ни с одной из уже вошедших в I вершин, – не может быть конвертирован в параллельный алгоритм с числом итераций, которое выражалось бы функцией от n, растущей существенно более медленно, чем линейная функция.

Это обстоятельство побуждает к поиску совершенно новых подходов, позволяющих создавать изначально параллельные алгоритмы. Мы рассмотрим здесь два таких подхода. Первый из них основан на использовании случайных переменных и метода Монте – Карло [1–3] и показан на примере решения задачи о максимальном независимом множестве, второй основан на факте равенства оптимальных решений двух двойственных оптимизационных задач [4–6] и используется для решения задачи о минимальном взвешенном вершинном покрытии графа. Эта задача формулируется следу-

ющим образом. В графе $G=(V, E)$ со взвешенными вершинами найти подмножество вершин $C \subseteq V$, обладающее следующими свойствами: 1) для любого ребра $(i, j) \in E$ $(i, j \in V)$, по крайней мере, одна из вершин i или j входит в C; 2) суммарный вес вершин, вошедших в C, является минимальным по сравнению со всеми другими подмножествами вершин графа, обладающими свойством 1. Подмножество вершин C называется минимальным взвешенным вершинным покрытием графа. Алгоритм, основанный на рассмотрении двойственных задач, состоит в отыскании приемлемого решения исходной задачи одновременно с определением приемлемого решения задачи, двойственной с ней. Для исходной задачи о минимальном взвешенном вершинном покрытии графа двойственной является задача о максимальной реберной упаковке графа, которая формулируется следующим образом: в графе со взвешенными вершинами назначить веса ребрам так, чтобы сумма весов ребер, инцидентных некоторой вершине, не превышала бы веса этой вершины, и при этом сумма весов всех ребер была бы максимальной по сравнению со всеми другими реберными упаковками. Заметим, что эти задачи являются NP-трудными [7].

Для быстрых параллельных алгоритмов, созданных на основе упомянутых подходов (подробное описание алгоритмов дано в разд. 2 и 3), естественным образом встает вопрос об отображении их в вычислительные устройства распределенной архитектуры. Ключевыми свойствами распределенных архитектур являются массовый параллелизм вычислений и преимущественная локальность связей между процессорными элементами. Рассматриваемые алгоритмы по своей структуре соответствуют этим свойствам. Они предполагают массовые параллельные вычисления (одновременная обработка всех вершин или всех ребер графа) и довольствуются локальными связями в архитектуре с топологией обрабатываемого графа (обработка каждой вершины требует информации только от соседних вершин или инцидентных с нею ребер, а обработка ребра – участия только двух вершин, соединяемых им).

Синтез таких архитектур в настоящей работе основан на оригинальной модели распределенных вычислений (Алгоритме параллельных подстановок [8, 9]), являющейся обобщением Клеточного автомата. Эта модель дает возможность представить алгоритм решения задачи в виде совокупности распределенных в пространстве и выполняемых параллельно во времени копий небольшого числа основных процедур.

1. **Алгоритм параллельных подстановок.** Алгоритм параллельных подстановок – формальная модель распределенных вычислений, предназначенная для организации совместной работы большого числа процессорных элементов с целью решения заданной задачи.

Процессорным элементам присваиваются имена. Множество имен M соответствует структуре входных данных задачи. Обычно это либо набор координат двумерного или трехмерного декартова пространства, если данные задачи располагаются в узлах целочисленной решетки соответствующей размерности, либо множество символов в задачах обработки графов, если топология связей между процессорными элементами принята адекватной топологии исходного графа. Процессорный элемент может находиться в состоянии из алфавита A, M и A – конечные множества. Пара (a, m), где $a \in A$ и $m \in M$, называется клеткой. Конечное множество клеток, в котором нет ни одной пары клеток с одинаковыми именами, называется клеточным массивом K.

Элементарной операцией над клеточным массивом является подстановка $S_1 * S_2 \Rightarrow S_3$, где S_1, S_2 и S_3 – множества клеток, $S_1 = \{(a_1, m_1) \ldots (a_p, m_p)\}$ называется базой, $S_2 = \{(b_1, m_{p+1}) \ldots (b_q, m_{p+q})\}$ – контекстом, $S_3 = \{(c_1, m_1) \ldots (c_p, m_p)\}$ – правой частью подстановки. Заметим, что $c_1, \ldots, c_p$ могут быть как конкретными состояниями, так и функциями от состояний $a_1, \ldots, a_p, b_1, \ldots, b_q$, в последнем случае подстановки называются функциональными. Подстановка $S_1 * S_2 \Rightarrow S_3$ применима к клеточному массиву K, если $S_1 \cup S_2 \subseteq K$. Выполнение подстановки состоит в замене клеток базы S_1 клетками правой части S_3, контекст служит условием применимости подстановки, его клетки не изменяются.

Для компактной записи одинаковых операций, выполняющихся параллельно в клеточном массиве, служит параллельная подстановка $\theta : S_1(m) * S_2(m) \Rightarrow S_3(m)$, в которой $S_1(m) = \{(a_1, \phi_1(m)) \ldots (a_p, \phi_p(m))\}$, $S_2(m) = \{(b_1, \psi_1(m)) \ldots (b_q, \psi_q(m))\}$, $S_3(m) = \{(c_1, \phi_1(m)) \ldots (c_p, \phi_p(m))\}$, где $\phi_i(m)$ $(i = 1, \ldots, p)$ и $\psi_j(m)$ $(j = 1, \ldots, q)$ – функции над множеством имен M с областью значений также в M, называемые именующими. Значения всех именующих функций для любого $m \in M$ должны быть различны. Именующие функции определяют соседство процессорных элементов.

В клеточном массиве K параллельные подстановки $\Phi = \{\theta_1, \ldots, \theta_k\}$ выполняются следующим образом. Пусть $K(t)$ – клеточный массив, являющийся результатом применения к исходному массиву K множества параллельных подстановок Φ в t итерациях. Тогда 1) если ни одна подстановка не применима к $K(t)$, то $K(t)$ есть результат вычисления, иначе 2) все применимые к $K(t)$ подстановки выполняются одновременно, в результате чего $K(t)$ преобразуется в клеточный массив $K(t+1)$ – результат $(t+1)$-й итерации.

Детерминированность вычисления по этой синхронной процедуре обеспечивается непротиворечивостью множества параллельных подстановок Φ, состоящей в том, что применение Φ к K не может порождать множество клеток, в котором есть хотя бы одна пара клеток с одинаковыми именами и разными состояниями. Непротиворечивое множество параллельных подстановок Φ, применяемое к K в соответствии с вышеописанной итерационной процедурой, называется алгоритмом параллельных подстановок.

2. **Задача о максимальном независимом множестве.** Алгоритм Монте – Карло решения задачи о максимальном независимом множестве [2] содержит две основные процедуры, которые выполняются в каждой итерации (сначала – одна, затем – другая). Первая применяется параллельно ко всем вершинам графа, вторая – параллельно ко всем ребрам. В результате выполнения первой процедуры формируется множество X вершин – претендентов на вхождение в независимое множество I. Вершина i включается в X с вероятностью $1/2d(i)$, где $d(i)$ – степень вершины i. Вторая процедура работает с каждой парой вершин, вошедших в X и связанных ребром. В результате применения второй процедуры к каждой такой паре вершин одна из них включается в I, а именно имеющая большую степень, чем другая. Далее те же процедуры выполняются в следующей итерации для графа, получившегося после удаления вершин, включенных в I и всех смежных с ними. И так до тех пор, пока исходный граф не окажется пустым. В результате формируется максимальное независимое множество. В [2] доказано, что с высокой вероятностью этот алгоритм завершает свою работу после $O(\log n)$ итераций.

В описанном простом алгоритме Монте – Карло попарно независимые случайные переменные (их число равно n) генерируются в каждой итерации

по правилу $r(i)=(x+y\times i)\bmod q$, $i=1,\ldots,n$, где x, y – случайно выбранные числа в интервале $[0, q-1]$, q – простое число из интервала $[n, 2n]$, определяющее мощность q^2 дискретного множества точек $\{(x, y)\}$ вероятностного пространства, каждая из которых соответствует одному из вариантов множества X. Введем дополнительные обозначения: $G'=(V', E')$ – подграф графа $G=(V, E)$, $N(i)=\{j\in V':(i,j)\in E'\}$ – соседство для вершины $i\in V'$, $N(W)=\{i\in V':\exists j\in W,(i,j)\in E'\}$ – соседство для подмножества вершин $W\subseteq V'$. Запишем псевдопрограмму алгоритма Монте – Карло (МК). Считаем, что q известно.

Алгоритм МК.

начало
  $G'(V', E') := G(V, E)$; $I := \varnothing$;
  пока $G' \neq \varnothing$, *выполнять*
    начало
      параллельно для всех $i \in V'$
        вычислить $d'(i)$;
        если $d'(i) = 0$, *то* i включить в I и удалить из V';
      случайно выбрать x и y в интервале $[0, q-1]$; $X := \varnothing$;
      параллельно для всех $i \in V'$
        $p(i) := q/2d'(i)$; $r(i) := (x + y\times i)\bmod q$;
        если $r(i) \le p(i)$, *то* i включить в X;
      параллельно для всех $(i, j) \in E'$, *если* $i \in X$ и $j \in X$, *то*
        если $d'(i) \le d'(j)$, *то* i удалить из X, *иначе* j удалить из X;
      $I := I \cup X$; $Y := X \cup N(X)$; $V' := V' \backslash Y$;
    конец
конец

Алгоритм МК можно реализовать в распределенной архитектуре следующего вида. Пусть имеется n узлов – по числу вершин в исходном графе. Топология соединения узлов повторяет топологию связи вершин в графе. Узел имеет имя i, совпадающее с номером вершины графа. Каждый узел i соединен с узлами $j_1^{(i)}, \ldots, j_k^{(i)}$, которые соответствуют вершинам графа, связанным ребрами с вершиной i. Каждый узел i объединяет шесть элементарных процессоров, далее «клеток», с именами $\langle i,1\rangle,\ldots,\langle i,6\rangle$, две из них ($\langle i,5\rangle$ и $\langle i,6\rangle$) являются просто ячейками памяти для констант i и q. Кроме n узлов, соответствующих вершинам графа, есть еще один узел, он имеет имя *ran*, связан с каждым из n узлов и содержит три клетки $\langle ran,1\rangle,\langle ran,2\rangle$ и $\langle ran,3\rangle$, в двух из которых генерируются случайные числа x и y, третья выполняет функцию счетчика. На рис. 1 показан фрагмент архитектуры для вершины i и смежных с нею.

Клетка $\langle i,2\rangle$ вычисляет текущую степень i-й вершины графа $d'(i)=|N(i)|$. Клетка $\langle i,1\rangle$ вычисляет случайное число $r(i)=2d'(i)((x+iy)\bmod q)$ для своего узла. Клетка $\langle i,3\rangle$ находится в состоянии 1, если вершина i входит в граф G', иначе – в состоянии 0. Клетка $\langle i,4\rangle$ находится в состоянии 1, если вершина i входит в X, в состоянии 2, если вершина i входит в I, в состоянии 0, если вершина i не входит ни в X, ни в I.

Алгоритм параллельных подстановок, управляющий работой этой совокупности процессоров, содержит функциональные подстановки, поэтому алфавит состояний клеток, кроме целых чисел, содержит переменные симво-

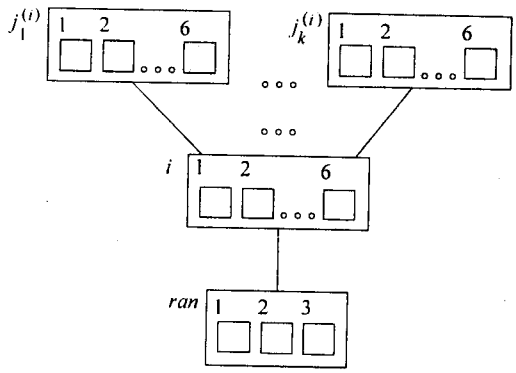

Рис. 1

лы $\{x, y, z_1, z_2, ...\}$, функциональные символы $\{f_1, f_2, ...\}$ и символ безразличного состояния «–»; области определения и значений функций, переменных и символа «–» лежат в интервале целых чисел $[0,...,2\Delta(q-1)]$, где Δ – максимальная степень вершин графа. Алгоритм МК реализуется в распределенной архитектуре шестью параллельными подстановками $\{\theta_1, ..., \theta_6\}$, геометрические образы которых показаны на рис. 2 и 3.

θ_1 : $\{(-, \langle ran, 1\rangle)(-, \langle ran, 2\rangle)(0, \langle ran, 3\rangle)\} \Rightarrow \{(f_1', \langle ran, 1\rangle)(f_1'', \langle ran, 2\rangle)(1, \langle ran, 3\rangle)\}$,
f_1'=<генерируется случайное число $0 \le x \le q-1$>,
f_1''=<генерируется случайное число $0 \le y \le q-1$>.

θ_2 : $\{(-, \langle i, 2\rangle)\} * \{(1, \langle i, 3\rangle)(z_{j_1}, \langle j_1^{(i)}, 3\rangle) ... (z_{j_k}, \langle j_k^{(i)}, 3\rangle)(0, \langle ran, 3\rangle)\} \Rightarrow \{(f_2, \langle i, 2\rangle)\}$,
$f_2 = z_{j_1} + ... + z_{j_k}$.

θ_3 : $\{(-, \langle i, 1\rangle)(1, \langle ran, 3\rangle)\} *$
$\{(z_2, \langle i, 2\rangle)(1, \langle i, 3\rangle)(x, \langle ran, 1\rangle)(y, \langle ran, 2\rangle)(i, \langle i, 5\rangle)(q, \langle i, 6\rangle)\} \Rightarrow$
$\{(f_3, \langle i, 1\rangle)(2, \langle ran, 3\rangle)\}$,
$f_3 = 2z_2((x + iy) \bmod q)$.

θ_4 : $\{(0, \langle i, 4\rangle)(2, \langle ran, 3\rangle)\} * \{(z_1, \langle i, 1\rangle)(1, \langle i, 3\rangle)(q, \langle i, 6\rangle)\} \Rightarrow$
$\{(f_4, \langle i, 4\rangle)(3, \langle ran, 3\rangle)\}$,
$f_4 = 1$, если $z_1 \le q$.

θ_5 : $\{(1, \langle i, 4\rangle)(3, \langle ran, 3\rangle)\} * \{(z_2, \langle i, 2\rangle)(z_2^l, \langle j_l^{(i)}, 2\rangle)(1, \langle j_l^{(i)}, 4\rangle)\} \Rightarrow$
$\{(f_5, \langle i, 4\rangle)(4, \langle ran, 3\rangle)\}$, $l = 1, ..., k$,
$f_5 = 0$, если $z_2 \le z_2^l$.

θ_6 : $\{(1, \langle i, 4\rangle)(1, \langle i, 3\rangle)(-, \langle j_1^{(i)}, 3\rangle) ... (-, \langle j_k^{(i)}, 3\rangle)(4, \langle ran, 3\rangle)\} \Rightarrow$
$\{(2, \langle i, 4\rangle)(0, \langle i, 3\rangle)(0, \langle j_1^{(i)}, 3\rangle) ... (0, \langle j_k^{(i)}, 3\rangle)(0, \langle ran, 3\rangle)\}$.

Начальные состояния клетки $\langle ran, 3\rangle$, а также клеток $\langle i, 1\rangle$, $\langle i, 2\rangle$, $\langle i, 4\rangle$ узла i, $i = 1, ..., n$, равны 0. Начальное состояние клетки $\langle i, 3\rangle$ равно 1, что соответствует равенству $G' = G$. Начальные состояния клеток $\langle ran, 1\rangle$ и $\langle ran, 2\rangle$ определяются методом выработки случайных чисел. Алгоритм завершает свою работу, когда все клетки $\langle i, 3\rangle$, $i = 1, ..., n$, переходят в нулевое состояние. Результат работы алгоритма – максимальное независимое множество – определяется состояниями 2 некоторой части клеток $\langle i, 4\rangle$, $i = 1, ..., n$.

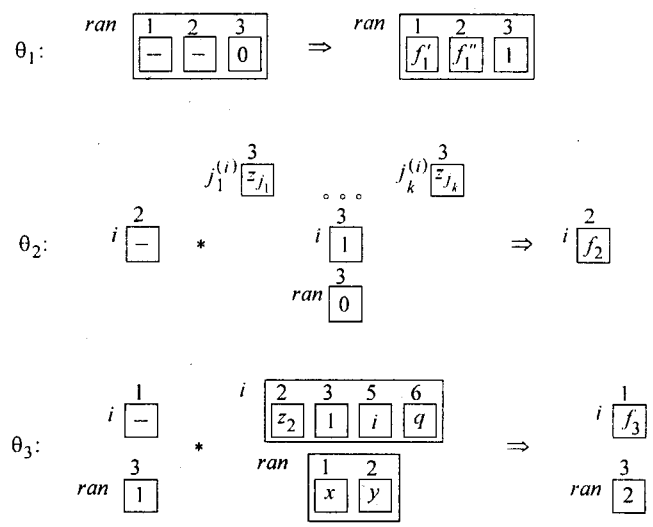

Рис. 2

В каждой итерации в первом такте выполняются подстановки θ_1 и θ_2 для всех узлов i, $i = 1, \ldots, n$, одновременно, θ_1 вырабатывает два случайных числа, θ_2 вычисляет степени вершин подграфа G'. Затем (это уже второй такт итерации) θ_3 вычисляет в каждом узле i свое случайное число. В третьем такте путем параллельного выполнения подстановки θ_4 во всех узлах i формируется подмножество вершин X. В четвертом такте для каждой пары соседних узлов таких, что соответствующие им вершины принадлежат X, выполняется подстановка θ_5, которая включает одну из вершин в независи-

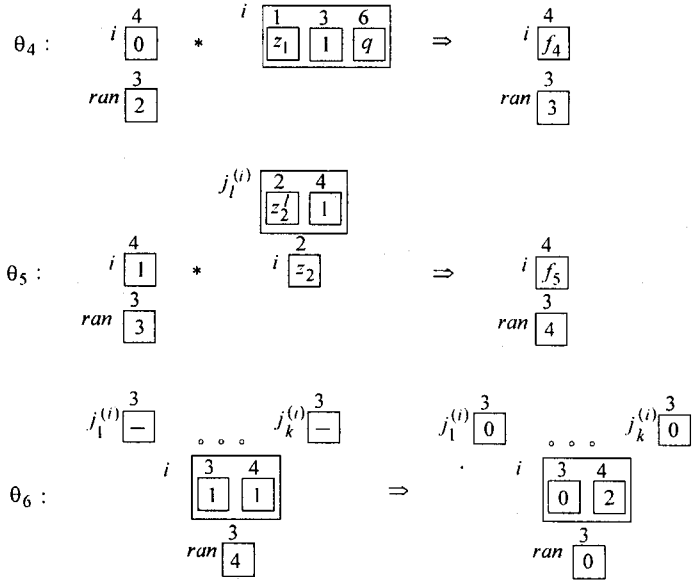

Рис. 3

мое множество. И наконец, в пятом такте путем параллельного выполнения подстановки θ_6 во всех узлах i из графа удаляются все вершины, вошедшие в I и соседние с ними. Порядок выполнения подстановок регулируется изменением состояния клетки $\langle ran, 3\rangle$ и контекстами подстановок. Подстановки непротиворечивы. Алгоритм завершает работу при неприменимости подстановок, когда состояния всех клеток $\langle i, 3\rangle$, $i = 1, \ldots, n$, равны 0 и состояние клетки $\langle ran, 3\rangle$ равно 1.

3. **Задача о минимальном взвешенном вершинном покрытии.** Для приближенных решений двойственных задач о минимальном взвешенном вершинном покрытии графа и о максимальной реберной упаковке выполняется соотношение

$$\sum_{i \in C} w(i) \geq \sum_{(i,j) \in E} w(i,j),$$

где $C \subseteq V$ – вершинное покрытие графа; $w(i)$ – вес вершины i; $w(i,j)$ – вес ребра (i,j). Предполагая, что для любой вершины $i \in C$ и множества $Inc(i)$ ребер, инцидентных вершине i, выполняется неравенство

$$\sum_{(i,j) \in Inc(i)} w(i,j) \geq (1-\varepsilon) w(i),$$

где $\varepsilon \in (0,1)$, получаем

$$(1-\varepsilon) \sum_{i \in C} w(i) \leq \sum_{i \in C} \sum_{(i,j) \in Inc(i)} w(i,j) \leq 2 \sum_{(i,j) \in Inc(i)} w(i,j),$$

откуда следует, что приближенные решения двойственных задач о минимальном вершинном покрытии и максимальной реберной упаковке отличаются от соответствующих оптимальных решений коэффициентом $2/(1-\varepsilon)$. Таким образом, задача о покрытии сводится к задаче об упаковке. Алгоритм ИД (исходная – двойственная (задачи)), начиная с нулевой упаковки, в каждой итерации одновременно как бы увеличивает веса ребер за счет уменьшения весов инцидентных им вершин до тех пор, пока после выполнения некоторого числа итераций не будет сформировано покрытие C такое, что для каждой вершины $i \in C$ $w'(i) \leq \varepsilon w(i)$, где $w'(i)$ – текущий вес вершины i. В каждой итерации все вершины, для которых выполняется упомянутое выше неравенство, включаются в покрытие C и вместе со смежными ребрами удаляются из графа. Чтобы гарантировать построение реберной упаковки (т. е. выполнение условия: сумма весов ребер, инцидентных данной вершине, не должна превышать веса этой вершины), вес ребра (i,j) можно увеличить в каждой итерации только на величину

$$\delta(i,j) = \min\left\{\frac{w'(i)}{d'(i)}, \frac{w'(j)}{d'(j)}\right\},$$

где $d'(i)$ – степень вершины i в соответствующем подграфе $G' \subseteq G$. В [6] доказано, что за $O(\log m)$ итераций завершается построение приближенного решения задачи о минимальном вершинном покрытии, которое не более чем в $2/(1-\varepsilon)$ раз больше оптимального.

Алгоритм ИД.

начало

$G'(V', E') := G(V, E)$; $C := \varnothing$;

параллельно для всех $i \in V'$ $w'(i) := w(i)$;

пока $G' \neq \varnothing$, *выполнять*

начало

параллельно для всех $i \in V'$ вычислить $d'(i)$;

параллельно для всех $(i, j) \in E'$

$$\delta(i, j) = \min\left\{\frac{w'(i)}{d'(i)}, \frac{w'(j)}{d'(j)}\right\};$$

параллельно для всех $i \in V'$

$$w'(i) := w'(i) - \sum_{(i, j) \in Inc(i)} \delta(i, j);$$

если $w'(i) \le \varepsilon w(i)$, *то*

$C := C \cup i$; $V' := V' \setminus i$; $E' := E' \setminus \{(i, j) \in Inc(i)\}$;

конец

конец

Алгоритм ИД можно реализовать в распределенной архитектуре следующего вида. Пусть имеется $m+n$ узлов, n из них соответствуют вершинам, m – ребрам графа. Топология архитектуры соответствует топологии графа. Узел, соответствующий вершине, имеет имя i, узел, соответствующий ребру, – имя $[i, j]$. Каждый узел i соединен с узлами $[i, j_1^{(i)}], \ldots, [i, j_k^{(i)}]$, где $k = d(i)$ – степень вершины i. Узел i объединяет семь клеток $\langle i,1\rangle, \ldots, \langle i, 7\rangle$, две из них ($\langle i,5\rangle$ и $\langle i,6\rangle$) являются ячейками памяти для констант $w(i)$ и ε, $\langle i, 7\rangle$ – счетчик. Узел $[i, j]$ содержит две клетки $\langle [i, j], 1\rangle, \langle [i, j], 2\rangle$. На рис. 4 показан фрагмент архитектуры для вершины i и смежных с нею.

Клетки $\langle i,1\rangle$ и $\langle i,2\rangle$ вычисляют соответственно текущие величины веса $w'(i)$ и степени $d'(i)$ i-й вершины графа. Клетки $\langle i,3\rangle$ и $\langle i,4\rangle$ могут иметь состояние 1 или 0, что означает для $\langle i,3\rangle$ вхождение или невхождение вершины i в подграф G', а для $\langle i,4\rangle$ вхождение или невхождение вершины i в покрытие C. Клетка $\langle [i, j], 1\rangle$ вычисляет величину $\delta(i, j)$. Клетка $\langle [i, j], 2\rangle$ в состоянии 1 информирует о том, что ребро (i, j) входит в подграф G', в состоянии 0 – об обратном.

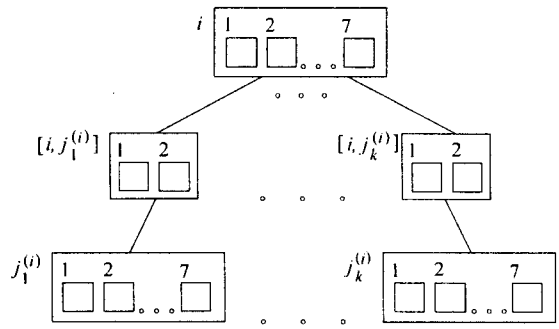

Рис. 4

Алгоритм ИД выполняют пять параллельных подстановок $\theta_1, \ldots, \theta_5$, далее приведены их формульные выражения, а на рис. 5 и 6 – геометрические образы.

θ_1 : $\{(-, \langle i, 2\rangle)(0, \langle i, 7\rangle)\} * \{(1, \langle i, 3\rangle)(z_{j_1}, \langle [i, j_1^{(i)}], 2\rangle) \ldots (z_{j_k}, \langle [i, j_k^{(i)}], 2\rangle)\} \Rightarrow$
$\{(f_1, \langle i, 2\rangle)(1, \langle i, 7\rangle)\}$,
$f_1 = z_{j_1} + \ldots + z_{j_k}$.

θ_2 : $\{(-, \langle [i, j], 1\rangle)(1, \langle i, 7\rangle)(1, \langle j, 7\rangle)\} *$
$\{(z_{i1}, \langle i, 1\rangle)(z_{i2}, \langle i, 2\rangle)(z_{j1}, \langle j, 1\rangle)(z_{j2}, \langle i, 2\rangle)(1, \langle i, 3\rangle)(1, \langle j, 3\rangle)\} \Rightarrow$
$\{(f_2, \langle [i, j], 1\rangle)(2, \langle i, 7\rangle)(2, \langle j, 7\rangle)\}$,
$f_2 = \min\{z_{i1}/z_{i2};\ z_{j1}/z_{j2}\}$.

θ_3 : $\{(z_1, \langle i, 1\rangle)(2, \langle i, 7\rangle)\} * \{(1, \langle i, 3\rangle)(z_{j_1}, \langle [i, j_1^{(i)}], 1\rangle) \ldots (z_{j_k}, \langle [i, j_k^{(i)}], 1\rangle)\} \Rightarrow$
$\{(f_3, \langle i, 1\rangle)(3, \langle i, 7\rangle)\}$,
$f_3 = z_1 - [z_{j_1} + \ldots + z_{j_k}]$.

θ_4 : $\{(0, \langle i, 4\rangle)(3, \langle i, 7\rangle)\} * \{(z_1, \langle i, 1\rangle)(1, \langle i, 3\rangle)(z_5, \langle i, 5\rangle)(z_6, \langle i, 6\rangle)\} \Rightarrow$
$\{(f_4, \langle i, 4\rangle)(4, \langle i, 7\rangle)\}$,
$f_4 = 1$, если $z_1 \le z_5 z_6$.

θ_5 : $\{(1, \langle i, 3\rangle)(4, \langle i, 7\rangle)(-, \langle [i, j_1^{(i)}], 2\rangle) \ldots (-, \langle [i, j_k^{(i)}], 2\rangle)\} * \{(1, \langle i, 4\rangle)\} \Rightarrow$
$\{(0, \langle i, 3\rangle)(0, \langle i, 7\rangle)(0, \langle [i, j_1^{(i)}], 2\rangle) \ldots (0, \langle [i, j_k^{(i)}], 2\rangle)\}$.

В каждом узле i начальное состояние клетки $\langle i, 1\rangle$ равно весу $w(i)$ вершины i в исходном графе, начальное состояние клетки $\langle i, 3\rangle$ равно 1 (что соответствует равенству $V' = V$), а клеток $\langle i, 2\rangle, \langle i, 4\rangle$ и $\langle i, 7\rangle$ – 0. В каждом узле $[i, j]$ клетка $\langle [i, j], 1\rangle$ имеет начальное состояние 0, клетка $\langle [i, j], 2\rangle$ – 1 (что соответствует равенству $E' = E$). Алгоритм завершает свою работу, когда все клетки $\langle i, 3\rangle, i = 1, \ldots, n$, переходят в нулевое состояние. Результат работы алгоритма: минимальное взвешенное вершинное покрытие определяется единичными состояниями некоторых клеток $\langle i, 4\rangle, i = 1, \ldots, n$.

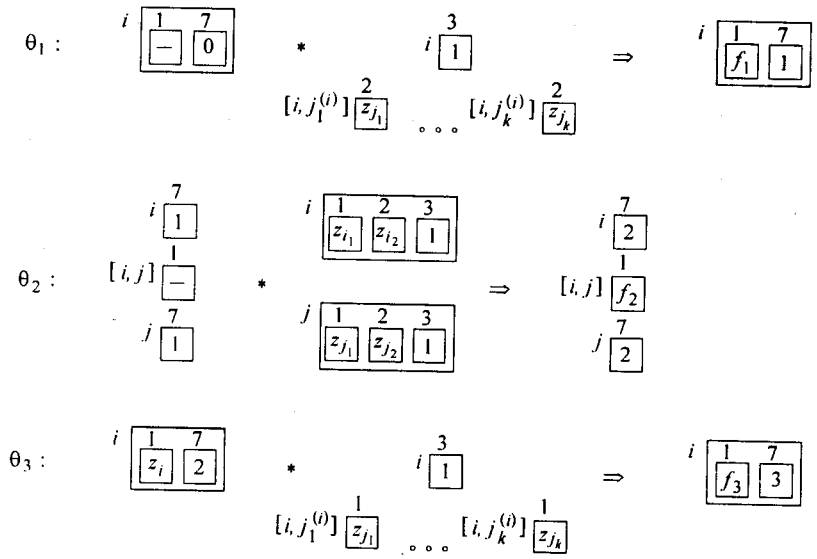

Рис. 5

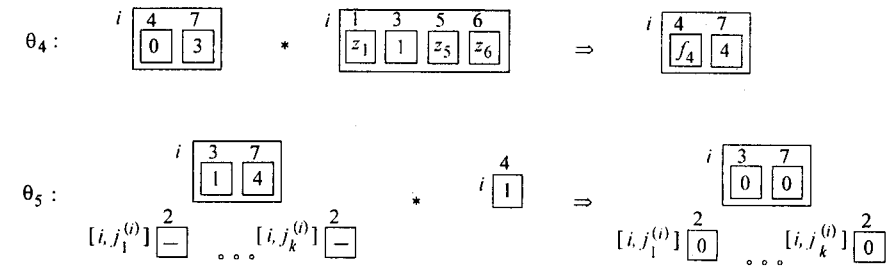

Рис. 6

В каждой итерации одна за другой, начиная с первой, выполняются все пять параллельных подстановок. Подстановка θ_1 работает с каждым узлом i и соседними с ним узлами $[i, j_l^{(i)}], l = 1, \ldots, k$, и вычисляет текущую степень $d'(i)$ вершины графа i; θ_2 применяется к каждому узлу $[i, j]$ и соседним узлам i и j и находит подходящую добавку к весу ребра $(i, j) \in E'$; θ_3, как и θ_1, работает с каждым узлом i и соседними с ним узлами $[i, j_l^{(i)}]$ и модифицирует (уменьшает) вес каждой вершины в соответствии с тем, как должны бы измениться (увеличиться) веса инцидентных с этой вершиной ребер; θ_4 применяется только к узлам i и выявляет вершины, для которых выполнилось условие вхождения в покрытие C; и наконец, θ_5 применяется к каждому узлу i и соседним с ним узлам $[i, j_l^{(i)}]$ и удаляет из подграфа G' ребра, инцидентные вершинам, включенным в C. Подстановки с очевидностью непротиворечивы. Алгоритм завершает свою работу при неприменимости подстановки θ_1.

Заключение. Оригинальная модель распределенных вычислений – Алгоритм параллельных подстановок – применена для синтеза распределенной архитектуры, предназначенной для реализации параллельных алгоритмов решения двух задач комбинаторной оптимизации: о максимальном независимом множестве и о минимальном взвешенном вершинном покрытии. Подходы, примененные при синтезе этих алгоритмов (один из них основан на методе Монте – Карло, другой – на двойственности оптимизационных задач), объединяются тремя интересными свойствами: 1) на их основе создаются изначально параллельные алгоритмы; 2) структура алгоритмов позволяет реализовать их в распределенных архитектурах с преимущественно локальными связями между элементарными процессорами; 3) ключевая компонента в оценке временной сложности параллельного решения упомянутых оптимизационных задач – число необходимых итераций – является логарифмом от входного параметра задачи. Алгоритмы параллельных подстановок, приведенные в разд. 2 и 3, реализуют все эти свойства в распределенной архитектуре, имеющей топологию исходного графа, для которого решается та или другая задача.

СПИСОК ЛИТЕРАТУРЫ

1. **Carp R., Wigderson A.** A fast parallel algorithm for the maximal independent set problem // Journ. ACM. 1985. **32**. P. 762.
2. **Luby M.** A simple parallel algorithm for the maximal independent set problem // SIAM. Journ. Comput. 1986. **15**, N 4. P. 1036.

3. **Clementi A., Rolim J. D. P., Urland E.** Randomized parallel algorithms // Lect. Notes in Comput. Sci. 1996. **1054**. P. 25.

4. **Papadimitriou C. H., Steiglitz K.** Combinatorial Optimization. Algorithms and Complexity. N. Y.: Prentice-Hall, 1982.

5. **Khuller S., Vishkin U., Young N.** Primal-dual parallel approximation technique applied to weighted set and vertex cover // Journ. Algorithms. 1994. P. 280.

6. **Bovet D. P., Clementi A., Crescenzi P., Silvestri R.** Parallel approximation of optimization problems // Lect. Notes in Comput. Sci. 1996. **1054**. P. 7.

7. **Гэри М., Джонсон Д.** Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.

8. **Achasova S., Bandman O., Markova V., Piskunov S.** Parallel substitution algorithm. Theory and Application // World Scientific. Singapore, 1994.

9. **Ачасова С. М., Бандман О. Л.** Корректность параллельных вычислительных процессов. Новосибирск: Наука, 1990.

Поступила в редакцию 20 апреля 1998 г.