© 2012 г. М.С. ТАРКОВ, канд. техн. наук
(Институт физики полупроводников
им. А.В. Ржанова СО РАН, Новосибирск)

ОБ ЭФФЕКТИВНОСТИ ПОСТРОЕНИЯ ГАМИЛЬТОНОВЫХ ЦИКЛОВ В ГРАФАХ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ РЕКУРРЕНТНЫМИ НЕЙРОННЫМИ СЕТЯМИ

Рассмотрено построение рекуррентной нейронной сетью гамильтоновых циклов в графе распределенной вычислительной системы с n вершинами. Предложен метод частичных сумм, позволяющий сократить время решения системы дифференциальных уравнений, описывающих нейронную сеть, с $O(n^3)$ до $O(n^2)$. Показано, что нейросетевой алгоритм, использующий метод частичных сумм, не уступает по времени построения цикла известным перестановочным методам.

ON EFFICIENSY OF CONSTRUCTING HAMILTON CYCLES IN GRAPHS OF DISTRIBUTED COMPUTER SYSTEMS BY RECURRENT NEURAL NETWORKS/ M.S. Tarkov (A.V. Rzhanov Institute of Semiconductor Physics SB RAS, Lavrentieva avenue, 13, Novosibirsk 630090, Russia, E-mail: tarkov@isp.nsc.ru). The construction of Hamiltonian cycles in the graph of distributed computing system with n vertices by a recurrent neural network is considered. A method of partial sums, which allows to reduce from $O(n^3)$ to $O(n^2)$ the time of solution of a system of differential equations describing the neural network, is proposed. It is shown that the neural network algorithm using the partial sums is not slower than the known permutation methods.

1. Введение

Распределенная вычислительная система (ВС) [1-5] представляет собой множество элементарных машин (ЭМ), связанных сетью, программно управляемой из этих машин. Каждая ЭМ включает вычислительный модуль (ВМ) и системное устройство (маршрутизатор сообщений). Маршрутизатор функционирует под управлением ВМ и имеет входные и выходные полюса, связанные соответственно с выходными и входными полюсами соседних ЭМ. Структура ВС описывается графом $G_s(V_s, E_s)$, где V_s – множество ЭМ и $E_s \subseteq V_s \times V_s$ – множество связей между ЭМ.

Для распределенных ВС граф $G_p(V_p, E_p)$ параллельной программы обычно определяется как множество V_p ветвей программы (виртуальных элементарных машин), которые взаимодействуют друг с другом по принципу «точка-точка» посредством передачи сообщений по логическим (виртуальным) каналам (одно- и двунаправленным) множества $E_p \subseteq V_p \times V_p$. Для большинства параллельных прикладных программ характерны упорядоченные во времени и регулярные в пространстве схемы взаимодействий между обрабатывающими модулями («линейка», «кольцо», «решетка» и др.). В силу этого для максимальной эффективности информационных взаимодействий современные высокопроизво-

дительные ВС используют регулярные графы $G_s(V_s, E_s)$ межмашинных соединений (гиперкуб и многомерные торы) [3-5]. Гиперкубическая структура описывается графом, известным как m-мерный булевский куб с числом вершин $n = 2^m$. Тороидальные структуры представляют собой m-мерные евклидовы решетки с замкнутыми границами. Группа автоморфизмов E_m такой структуры есть прямое произведение циклических подгрупп C_{N_k}: $E_m = \overset{m}{\underset{k=1}{\otimes}} C_{N_k}$, где N_k – порядок подгруппы C_{N_k}, $\otimes$ – символ прямого произведения. При $m = 2$ получаем двумерный тор (2D-тор) (рис.1), при $m = 3$ получаем 3D-тор.

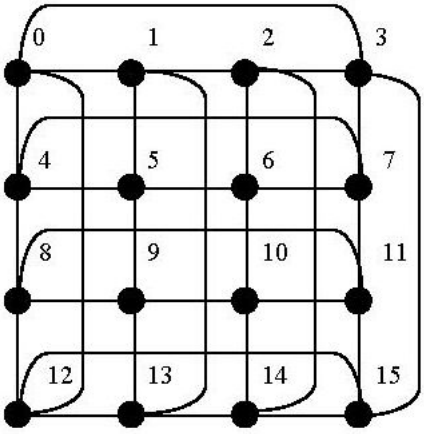

Рис.1. Пример 2D-тора

В силу того, что элементарные машины и межмашинные соединения не являются абсолютно надежными, регулярность структуры ВС может нарушаться в результате выхода из строя ее элементов. Поэтому возникает необходимость разработки алгоритмов вложения графов программ в произвольные графы ВС. В данной работе рассматриваются нейросетевые алгоритмы вложения кольцевых структур параллельных программ в структуры распределенных ВС при условии $V_p = V_s$. Такое вложение сводится к построению гамильтонова цикла в графе ВС и основывается на решении задачи коммивояжера (раздел 2), использующем матрицу расстояний между вершинами графа ВС, причем за единицу принято расстояние между соседними вершинами графа ВС. В разделе 3 предложен метод частичных сумм, позволяющий сократить время решения системы дифференциальных уравнений, описывающих нейронную сеть, с $O(n^3)$ до $O(n^2)$. В разделе 4 проведено сравнение полученных результатов с результатами построения цикла известными перестановочными методами [1, 6-11] и показано, что нейросетевой алгоритм, использующий метод частичных сумм, не уступает им по времени выполнения.

2. О решении задачи коммивояжера рекуррентными нейронными сетями

Решению задач комбинаторной оптимизации нейронными сетями посвящено много работ [12-15]. Наиболее популярными в этом плане являются сети Хопфилда [13], но их применение ограничивается высокой вычислительной сложностью (n^4, где n - характерный размер задачи). Ситуацию можно упростить, если применить следующий подход.

Задача коммивояжера может быть сформулирована как задача о назначении [16,17]:

(1) $$\min \sum_{i=1}^{n} \sum_{j \neq i} C_{ij} x_{ij}$$

при ограничениях

$$x_{ij} \in \{0,1\},$$

(2) $$\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,2,...,n,$$

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,2,...,n.$$

Здесь C_{ij} $(i \neq j)$ – стоимость назначения элемента i в позицию j, что соответствует перемещению коммивояжера из города i в город j; x_{ij} – переменная решения: (если элемент i назначается в позицию j, то $x_{ij} = 1$, иначе $x_{ij} = 0$).

Для решения этой задачи Ваном (J.Wang) [16] предложена рекуррентная нейронная сеть, которая описывается дифференциальным уравнением

(3) $$\frac{\partial u_{ij}(t)}{\partial t} = -\eta\left(\sum_{k=1}^{n} x_{ik}(t) + \sum_{l=1}^{n} x_{lj}(t) - 2\right) - \lambda C_{ij} \exp\left(-\frac{t}{\tau}\right),$$

где $x_{ij} = f(u_{ij}(t));\ f(u) = 1/[1 + \exp(-\beta u)]$.

Разностный вариант этого уравнения имеет вид

(4) $$u_{ij}^{t+1} = u_{ij}^{t} - \Delta t\left[\eta\left(\sum_{k=1}^{n} x_{ik}(t) + \sum_{l=1}^{n} x_{lj}(t) - 2\right) - \lambda C_{ij} \exp\left(-\frac{t}{\tau}\right)\right],$$

где Δt – шаг по времени. Параметры $\Delta t, \eta, \lambda, \tau, \beta$, подбираемые экспериментально, существенно влияют на скорость достижения решения задачи и качество этого решения. Вычислительная сложность выполнения итерации (4) равна $O(n^3)$, если для всех n^2 элементов матрицы $u_{ij}, i, j = 1,...,n$ активаций нейронов заново рассчитывать сумму $\sum_{k=1}^{n} x_{ik}(t) + \sum_{l=1}^{n} x_{lj}(t)$.

Решение системы уравнений (3) можно ускорить следующим алгоритмом [19], использующим принцип WTA:

1. Породить матрицу $\|x_{ij}(0)\|$ случайных значений $x_{ij}(0) \in [0,1]$. Итерации уравнения (4) продолжать до тех пор, пока для всех $i, j = 1,...,n$ не выполнится неравенство

$$\sum_{k=1}^{n} x_{ik}(t) + \sum_{l=1}^{n} x_{lj}(t) - 2 \leq \delta,$$

где δ – заданная точность выполнения ограничений (2).

2. Выполнить преобразование полученной матрицы решения $\|x_{ij}\|$:

2.1. $i = 1$.

2.2. В i-й строке матрицы найти максимальный элемент $x_{i,j_{\max}}$, $j_{\max}$ – номер столбца с максимальным элементом.

2.3. Выполнить преобразование $x_{i,j_{\max}} = 1$. Все остальные элементы i-й строки и столбца с номером $j_{\max}$ обнулить. Перейти к строке с номером $j_{\max}$.

Действия 2.2 и 2.3 повторять, пока не произойдет возврат к первой строке, что будет означать завершение построения цикла.

3. Если возврат к строке 1 произошел раньше, чем в матрице $\|x_{ij}\|$ значение 1 получили n элементов, то это означает, что длина построенного цикла меньше n. В этом случае шаги 1 и 2 повторить.

3. Метод частичных сумм

В [18] показано, что рекуррентная сеть Вана (1-3) дает хорошие результаты при решении системы уравнений (4) методом Зейделя. Нетрудно видеть, что решение задачи (1-3) связано с многократным вычислением одних и тех же частичных сумм, входящих в суммы

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,...,n,$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,...,n.$$

Во избежание избыточных вычислений произведем расчет частичных сумм:

– «вертикальные» частичные суммы

$$v_{ij} = \sum_{k=i}^{n} x_{kj}, v'_{ij} = \sum_{k=1}^{i} x'_{kj}, \; i, j = 1,...,n \; ;$$

– «горизонтальные» частичные суммы

$$h_{ij} = \sum_{l=j}^{n} x_{il}, h'_{ij} = \sum_{l=1}^{j} x'_{il}, \; i, j = 1,...,n \; ,$$

где x'_{ij} – обновленное по Зейделю значение x_{ij} .

Обозначим сумму $s_{ij} = \sum_{k=1}^{n} x^*_{ik}(t) + \sum_{l=1}^{n} x^*_{lj}(t)$, где массив x^* включает элементы x_{kl}, где $k = i, \; l = j,...,n$ и $k = i+1,...,n, \; l = 1,...,n$, и обновленные по Зейделю элементы x'_{kl}, где $k = 1,...i-1, \; l = 1,...,n$ и $k = i, \; l = 1,..., j-1$.

Тогда в (4) получаем

(5) $$s_{11} = v_{11} + h_{11} \; .$$

Используя (5), по формуле (4) вычисляем новое значение x'_{11} и полагаем

$v'_{11} = h'_{11} = x'_{11}$.

Для остальных элементов первой строки матрицы $\|x_{ij}\|$ выполняем следующие вычисления:

1. Вычисляем $s_{1j} = h'_{1,j-1} + v_{1j} + h_{1j}$, $j = 2,...,n$.

2. Используя s_{1j} , вычисляем x'_{1j} по (4) и полагаем

$$v'_{1j} = x'_{1j},$$

$$h'_{1j} = x'_{1j} + h'_{1,j-1}, j = 2,...,n.$$

Для остальных строк с номерами $i = 2,...,n$ при $j = 1,...,n$:

1. Вычисляем

$s_{ij} = v'_{i-1,j} + v_{ij} + h_{ij}, j \in \{1,...,n\}$,

$s_{ij} = s_{ij} + h'_{i,j-1}, \ j \in \{2,...,n\}$.

2. Используя s_{ij}, вычисляем x'_{ij} и полагаем

$v'_{ij} = x'_{ij} + v_{i-1,j}$,

$h'_{ij} = x'_{1j}, j \in \{1,...,n\}$,

$h'_{ij} = h'_{ij} + h'_{1,j-1}, j \in \{2,...,n\}$.

Предложенный здесь метод частичных сумм позволяет сократить время решения системы (4) при построении гамильтоновых циклов в графах распределенных вычислительных систем с $O(n^3)$ до $O(n^2)$.

В таблице 1 приведены времена (в секундах) построения гамильтоновых циклов в двумерном торе (см. рис. 1) с $n = m \times m$ вершинами, $m \in \{12,16,20,24,28,32\}$, t_{trad} – время построения цикла традиционным способом (без учета повторяемости сумм), t_{part} – время построения при использовании метода частичных сумм (процессор Pentium ® Dual-Core CPU E 52000, 2,5 Ггц).

Таблица 1. Времена построения гамильтонова цикла в двумерном торе рекуррентной нейронной сетью

n	144	256	400	576	784	1024
t_{trad}	0,125	1,062	5,203	15,92	39,8	178,734
t_{part}	0,016	0,063	0,188	0,5	1,156	2,047

Особенно большой выигрыш метод частичных сумм дает при построении гамильтоновых циклов в трехмерных торах с большим числом (тысячи) вершин. При этом используется разбиение трехмерного тора на двумерные.

В общем виде эта задача решается так [18]:

1. Разбить исходный граф системы на $k \geq 2$ связных подграфов.
2. В каждом подграфе построить гамильтонов цикл, используя вышеописанный алгоритм.
3. Объединить гамильтоновы циклы подграфов в один гамильтонов цикл.

Для разбиения исходного графа системы на связные подграфы можно применить алгоритмы, предложенные в работе [2].

Для объединения двух циклов R_1 и R_2 достаточно наличия в графе ВС цикла $ABCD$ длины 4 такого, что ребро AB принадлежит циклу R_1, а ребро CD – циклу R_2 (см. рис. 2).

Объединить циклы R_1 и R_2 в один цикл можно следующим образом:

1. Найти цикл *ABCD*, обладающий указанным выше свойством.
2. Исключить из цикла ребро *AB* и пронумеровать вершины цикла R_1 последовательно так, чтобы вершина *A* получила номер 0, а вершина *B* – номер $L_1 - 1$, где L_1 - длина цикла R_1. Включить в цикл ребро *BC*.

3. Исключить ребро CD и пронумеровать вершины цикла R_2 последовательно так, чтобы вершина *C* получила номер L_1, а вершина *D* – номер $L_1 + L_2 - 1$, где L_2 - длина цикла R_2. Включить в цикл ребро *DA*. Единый цикл длины $L_1 + L_2$ построен.

Циклы R_1, R_2 и результирующий цикл выделены на рисунке 2 жирными линиями. Ребра, не входящие в указанные циклы, выделены пунктиром.

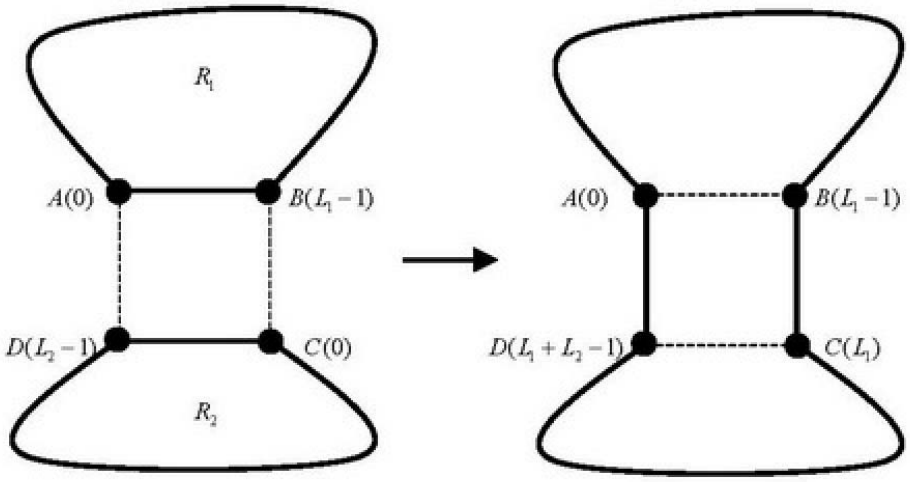

Рис. 2. Объединение циклов

Трехмерный тор можно рассматривать как совокупность двумерных торов, связанных ребрами между собой. Поэтому гамильтонов цикл для трехмерного тора можно построить следующим образом:

1. Построить гамильтоновы циклы для всех двумерных торов.
2. Объединить циклы в один алгоритмом, предложенным выше.

Очевидно, что если гамильтоновы циклы для двумерных торов оптимальны, то и результирующий гамильтонов цикл трехмерного тора также оптимален.

Таблица 2. Построение 3D-тора с использованием метода расщепления

m	16	20	24	28	32
$n = m^3$	4096	8000	13284	21952	32768
t_{RNseq}	13,19	65,25	243,7	707,5	3354
t_{RNpart}	2,437	7,656	18,64	41,08	84,47

В таблице 2 приведены времена (в секундах) построения оптимальных гамильтоновых циклов для трехмерных торов с большим (тысячи) числом вершин: t_{RNseq} – время работы алгоритма без использования частичных сумм, t_{RNpart} – время работы алгоритма с использованием частичных сумм на процессоре Intel Pentium Dual-Core CPU E 52000, 2,5 Ггц. Из таблицы 2 в частности следует, что время построения цикла в 3D-торе из 32768 вершин удалось сократить с 3354 секунд (около часа) до 84,4 секунд (около 1,5 минут).

Таким образом, результаты проведенных экспериментов показывают, что в совокупности с методом расщепления трехмерного тора на двумерные алгоритм, использующий частичные суммы позволяет существенно ускорить построение циклов в трехмерных торах с десятками тысяч вершин.

4. Перестановочный алгоритм вложения графа программы в граф вычислительной системы

Пусть граф $G_p(V_p,P_p)$ параллельной программы рассматривается как множество V_p вершин (ветвей программы) и функция

$$G_p : V_p \times V_p \to \{0,1\},$$

удовлетворяющая

$$G_p(x,y) = G_p(y,x),\ G_p(x,x) = 0$$

для любых $x, y \in V_p$. Равенство $G_p(x,y) = 1$ означает, что существует ребро между вершинами x и y, то есть $(x,y) \in E_p$. Аналогично граф $G_s = (V_s, E_s)$ определяется как множество вершин (элементарных машин – ЭМ) V_s и функция

$$G_s : V_s \times V_s \to \{0,1\}.$$

Здесь E_s - множество ребер (линий связи между ЭМ).

Пусть $|V_p| = |V_s| = n$. Обозначим вложение ветвей параллельной программы в ЭМ как одно-однозначную функцию $f_m : V_p \to V_s$. Качество вложения можно определить как число ребер графа программы, совпавших с ребрами графа ВС. Назовем это число мощностью $|f_m|$ вложения f_m и определим его следующим выражением [6] (критерий-максимум качества вложения):

(6) $$|f_m| = (1/2) \sum_{x \in V_p, y \in V_p} G_p(x,y) G_s(f_m(x), f_m(y)).$$

В [1, 10, 11] предложен следующий подход (МВ-алгоритм) к решению задачи вложения. Пусть некоторым образом задано начальное вложение вершин графа программы в вершины графа ВС. Например, $f_m(x) = x$, то есть номера ветвей графа совпадают с номерами содержащих эти ветви машин. Пусть $e_p(x)$ - окружение (множество соседей) вершины x на графе G_p и $e_s(x)$ – ее окружение на графе G_s. Для каждой вершины $x \in V_p$ протестируем перестановку вершин i и j, удовлетворяющих условию

(7) $$i \in e_p(x)\ \&\ i \notin e_s(x)\ \&\ j \in e_s(x)\ \&\ state(j) = 0.$$

Условие $state(j) = 0$ означает, что вершина j еще не подвергалась перестановке в окружении $e_s(x)$, в противном случае $state(j) = 1$. Если перестановка не ухудшает качество вложения f_m, мы ее фиксируем. Такой подход основан на предположении о высокой вероятности ситуации, когда такая перестановка, увеличивающая количество вершин $i \in e_p(x)$ в окружении $e_s(x)$, улучшит (или, по крайней мере, не ухудшит) значение критерия качества $|f_m|$. Число тестируемых перестановок при одноразовом обходе всех

вершин $x \in V_p$ не превышает значения $v_p v_s n,\ n = |V_p|$, где v_p и v_s - максимальные степени вершин графов G_p и G_s соответственно. При $v_p v_s < n$ такой подход обеспечит сокращение объема вычислений по сравнению с известными перестановочными алгоритмами [6-9], итерация которых имеет сложность $O(n^2)$. Разработана основанная на проверке условия (7) процедура Search поиска локального экстремума функции $|f_m|$ в МВ-алгоритме вложения.

Ниже представлен МВ-алгоритм для критерия (6). Здесь: *TASK* – описание G_p, *VS* – текущее вложение f_m, *BEST* – лучшее найденное вложение f_m, *card(f)* – значение критерия для вложения f, *card (TASK)* – значение критерия для вложения графа G_p в себя, *BEST* ← *VS* – создание копии BEST вложения VS, *Rand-interchange (VS)* – случайная перестановка n пар вершин во вложении VS, *Output(BEST)* – вывод наилучшего вложения f_m.

```
Procedure MB;
begin
 done=false; BEST ← VS;
  if (card(VS)=card(TASK)) done=true;
    while (done ≠ true)   do
      begin Search(VS,TASK);
        if (card(VS) > card(BEST))
          begin BEST ← VS; Rand_exchange(VS) end
        else     done=true;
      end
  Output(BEST)
end.
```

На рис.3. представлен шаг МВ-алгоритма при вложении кольца в двумерный тор. Здесь на торе жирными отрезками показаны ребра графа программы, совпадающие с ребрами графа ВС.

В таблице 3 производится сравнение времен (в секундах) построения гамильтонова цикла двумя алгоритмами – МВ-алгоритмом и рекуррентной нейронной сетью с использованием метода частичных сумм. Из этой таблицы следует, что эти алгоритмы сравнимы по эффективности. Следует учитывать, что МВ-алгоритм рассчитан на использование процессора с фиксированной запятой, в то время как нейронная сеть использует процессор с плавающей запятой. Этот факт нужно учитывать при выборе алгоритма для конкретной машины в зависимости от скорости выполнения алгоритма.

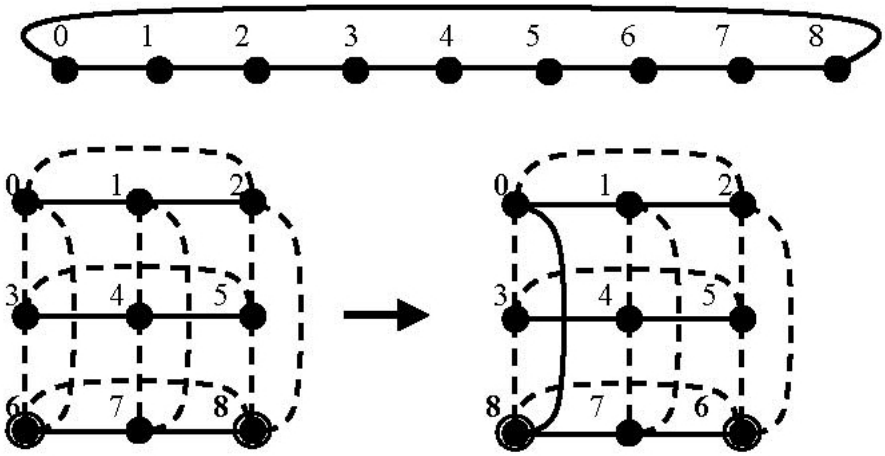

Рис.3. Шаг MB-алгоритма вложения кольца в 2D-тор

Таблица 3. Времена построения гамильтонова цикла в двумерном торе MB-алгоритмом (t_{MB}) и рекуррентной нейронной сетью с использованием метода частичных сумм (t_{part})

n	144	256	400	576	784	1024
t_{MB}	0,047	0,14	0,343	0,719	1,328	2,25
t_{part}	0,016	0,063	0,188	0,5	1,156	2,047

5. Выводы

Рассмотрено построение рекуррентной нейронной сетью гамильтоновых циклов в графе распределенной вычислительной системы с n вершинами. Предложен метод частичных сумм, позволяющий сократить время решения системы дифференциальных уравнений, описывающих нейронную сеть, с $O(n^3)$ до $O(n^2)$. Показано, что нейросетевой алгоритм, использующий метод частичных сумм, не уступает по времени построения цикла известным перестановочным методам.

Перестановочные алгоритмы не нуждаются в процессоре с плавающей запятой и подборе параметров, но работают с нерегулярными структурами данных (списковое представление графа), что может усложнить их распараллеливание. Нейросетевые алгоритмы нуждаются в процессоре с плавающей точкой и подборе параметров, но работают с регулярными структурами данных, а потому имеют более высокий потенциальный параллелизм. Таким образом, выбор алгоритма для построения гамильтоновых циклов в графах распределенных вычислительных систем определяется конфигурацией аппаратуры процессоров, на которых они выполняются. Сравнение параллельных версий рассмотренных алгоритмов является целью дальнейших исследований.

СПИСОК ЛИТЕРАТУРЫ

1. *Тарков М.С.* Вложение структур параллельных программ в структуры живучих распределенных вычислительных систем // Автометрия. 2003. том 39. №3. С.84 –96.
2. *Тарков М.С.* Децентрализованное управление ресурсами и заданиями в живучих распределенных вычислительных системах // Автометрия. 2005. том 41. №5. С.81 – 91.
3. *Palmer J.F.* The NCUBE family of parallel supercomputers//SCS Conf. on Multiprocessors and Array Processors. San Diego. CA (USA). 14-16 Jan. 1987. P. 177-188.
4. *Yu H., Chung I-Hsin, Moreira J.* Topology Mapping for Blue Gene/L Supercomputer // Proc. of the ACM/IEEE SC2006 Conf. On High Performance Networking and Computing. November 11-17, 2006, Tampa, FL, USA. ACM Press, 2006. P.52–64.
5. *Абрамов С.М., Заднепровский В.Ф., Шмелев А.Б., Московский А.А.* СуперЭВМ ряда 4 семейства «СКИФ»: Штурм вершины суперкомпьютерных технологий // Вестник Нижегородского университета им. Н.И. Лобачевского. 2009. №5. С.200-210.
6. *Bokhari S.H.* On the Mapping Problem // IEEE Trans. Comput. 1981. vol. C-30. no.3. pp.207-214.
7. *Lee S.-Y., Aggarval J.K.* A Mapping Strategy for Parallel Processing // IEEE Trans. Comput. 1981. vol. C-36. no.4. P. 433-442.
8. *Agarwal T., Sharma A., Kale L.V.* Topology-aware task mapping for reducing communication contention on large parallel machines// Proceedings of IEEE International Parallel and Distributed Processing Symposium, April 2006.
9. *Bollinger S. W., Midkiff S. F.* Heuristic Technique for Processor and Link Assignment in Multicomputers // IEEE Transactions on Computers. 1991. vol. 40. no. 3. P. 325-333.
10. *Тарков М.С.* Алгоритм отображения структур параллельных программ в структуры распределенных вычислительных систем // Труды Пятого Международного семинара «Распределенная обработка изображений (РОИ-95)». Новосибирск, 1995. С.124-129.
11. *Tarkov M.S.* Mapping Parallel Programs Onto Distributed Robust Computer Systems // Proc. of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics. Berlin, 1997. P.365-370.
12. *Тарков М.С.* Нейрокомпьютерные системы. М.: Интернет-Ун-т Информ. Технологий: Бином. Лаборатория знаний, 2006. 142 с.
13. *Hopfield J.J., Tank D.W.* "Neural"computation of decisions in optimization problems // Biological Cybernetics. 1985. vol.52. no.3. P.141-152.
14. *Меламед И.И.* Нейронные сети и комбинаторная оптимизация // Автоматика и телемеханика. 1994. №4. С.3-40.
15. *Smith K.A.* Neural Networks for Combinatorial Optimization: A Review of More Than a Decade of Research // INFORMS Journal on Computing. 1999. vol.11. no.1. P.15-34.
16. *Hung D.L., Wang J.* Digital hardware realization of a recurrent neural network for solving the assignment problem // Neurocomputing. 2003. vol. 51. P.447-461.
17. *Siqueira P.H., Steiner M.T.A., Scheer S.* A new approach to solve the travelling salesman problem // Neurocomputing. 2007. vol.70. P.1013-1021.
18. *Тарков М.С.* Построение гамильтоновых циклов в графах распределенных вычислительных систем рекуррентными нейронными сетями// Сиб. журнал вычислительной математики. 2010. №4. С.467-475.