УДК 004.032.26(06)

Решение задачи коммивояжера с использованием рекуррентной нейронной сети

М.С. Тарков

Институт физики полупроводников им. Акад. А.К. Ржанова Сибирского отделения Российской академии наук, просп. Акад. М.А. Лаврентьева, 13, Новосибирск, 630090
E-mail: tarkov@isp.nsc.ru

Тарков М.С. Решение задачи коммивояжера с использованием рекуррентной нейронной сети // Сиб. журн. вычисл. математики / РАН. Сиб. отд-ние. — Новосибирск, 2015. — Т. 18, № 3. — С. 337–347.

Предложен новый алгоритм (NWTA-алгоритм) решения задачи коммивояжера. Алгоритм основан на использовании рекуррентной нейронной сети Хопфилда, метода WTA ("Winner takes all") формирования цикла и метода 2-opt его оптимизации. Особенностью предложенного алгоритма является использование метода частичных (префиксных) сумм для ускорения решения системы уравнений сети Хопфилда. Для получения дополнительного ускорения выполнено распараллеливание предложенного алгоритма на графическом процессоре с использованием технологии CUDA. На ряде примеров из библиотеки TSPLIB с числом городов от 51 до 2392 показано, что NWTA-алгоритм находит приближенные решения задачи коммивояжера (относительное увеличение длины маршрута по сравнению с оптимальной составляет $0.03 \div 0.14$). При большом числе городов (130 и выше) время работы NWTA-алгоритма в $4 \div 24$ раз меньше времени работы эвристического алгоритма LKH, посредством которого получены оптимальные решения для всех примеров из TSPLIB.

DOI: 10.15372/SJNM20150308

Ключевые слова: *задача коммивояжера, нейронная сеть Хопфилда, 2-opt, технология CUDA, LKH-алгоритм.*

Tarkov M.S. Solving the traveling salesman problem using a recurrent neural network // Siberian J. Num. Math. / Sib. Branch of Russ. Acad. of Sci. — Novosibirsk, 2015. — Vol. 18, № 3. — P. 337–347.

A new algorithm (NWTA-algorithm) for solving the traveling salesman problem (TSP) is proposed. The algorithm is based on the use of the Hopfield recurrent neural network, the WTA method ("Winner takes all") for the cycle formation, and 2-opt optimization method. A special feature of the algorithm proposed is in the use of the method of partial (prefix) sums to accelerate the solution of the system of the Hopfield network equations. For attaining additional acceleration, the parallelization of the algorithm proposed is performed on GPU with the CUDA technology. Several examples from the TSPLIB library with the number of cities from 51 to 2,392 show that the algorithm proposed finds approximate solutions of the TSP (a relative increase in the length of the route with respect to the optimum is $0.03 \div 0.14$). With a large number of cities (130 and more) the NWTA-algorithm running duration on the CPU is in $4 \div 24$ times less than the duration of the heuristic LKH algorithm giving optimal solutions for all TSPLIB examples.

Keywords: *traveling salesman problem, recurrent neural Hopfield network, 2-opt, CUDA technology, LKH algorithm.*

1. Введение

Рассмотрим задачу коммивояжера для n городов. Известны стоимости C_{ij} перемещения коммивояжера между городами i, j, $i \neq j$; коммивояжер, выходя из одного города, должен посетить $n-1$ других городов, заходя по одному разу в каждый, и вернуться в

исходный. Требуется определить порядок обхода городов, при котором суммарная стоимость перемещений коммивояжера между городами минимальна. Задачу коммивояжера можно также сформулировать как задачу поиска гамильтонова цикла минимального веса в полном графе, ребра которого имеют положительные веса.

Локальный поиск с k соседями для замены ребер (k-opt) является наиболее широко используемым эвристическим методом решения задачи коммивояжера; k-opt является алгоритмом улучшения тура коммивояжера, где на каждом шаге k ребер текущего тура заменяются другими ребрами так, чтобы получился более короткий тур. В оригинальном варианте эвристики Лина–Кернигана [1] допустимы k-замены, которые могут быть представлены в виде 2- или 3-замен с последующими 2-заменами. В работах Гельсгауна [2–4] представлена LKH-версия алгоритма Лина–Кернигана, которая допускает любые замены, представимые в виде k-замен для любых $k \in 2, \ldots, n$. Посредством LKH-алгоритма получены оптимальные туры коммивояжера для всех примеров библиотеки TSPLIB [5].

При больших размерностях задачи коммивояжера более предпочтительным является метод, построенный на основе рекуррентной нейронной сети (в том числе при реализации на параллельных вычислительных системах), если достигаемая сетью точность решения удовлетворительна. Это требование, называемое принципом рациональной строгости и предполагающее отказ от абсолютно точного решения задачи, характерно для ситуаций, когда решение задачи необходимо получить в максимально сжатые сроки [6]. Данная работа ставит своей целью разработку и анализ быстродействующего алгоритма приближенного решения задачи коммивояжера.

В отличие от традиционных методов комбинаторной оптимизации рекуррентные нейронные сети Хопфилда [7–12] не занимаются перебором вариантов. Сеть Хопфилда сходится к локально-оптимальному решению задачи, но промежуточные состояния нейронов образуют не допустимое решение, а приближение к нему, которое задается матрицей действительных чисел из интервала $(0, 1)$.

2. Постановка задачи

Задача коммивояжера может быть сформулирована как задача о назначении [13–16]:

$$\min \sum_{i=1}^{n} \sum_{j \neq i} C_{ij} v_{ij} \tag{1}$$

при ограничениях:

$$\begin{gathered} v_{ij} \in \{0, 1\}, \\ \sum_{i=1}^{n} v_{ij} = 1, \quad j = 1, 2, \ldots, n, \\ \sum_{j=1}^{n} v_{ij} = 1, \quad i = 1, 2, \ldots, n, \end{gathered} \tag{2}$$

$$v \text{ образует гамильтонов цикл.} \tag{3}$$

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

3. Решение задачи нейронной сетью

Для решения задачи используется матрица нейронов размером $n \times n$. Задача (1)–(3) решается рекуррентной нейронной сетью Хопфилда–Вана, которая описывается дифференциальным уравнением

$$\frac{\partial u_{ij}(t)}{\partial t} = -\eta\left(\sum_{k=1}^{n} v_{ik}(t) + \sum_{l=1}^{n} v_{lj}(t) - 2\right) - \lambda C_{ij} \exp\Big(-\frac{t}{\tau}\Big), \tag{4}$$

где $v_{ij} = f\big(u_{ij}(t)\big)$, $f(u) = 1/\big(1+\exp(-\beta u)\big)$, β, η, λ, τ — параметры сети.

Чтобы уменьшить вероятность преждевременного возврата к исходной строке i_{start}, заменим в (4) стоимость C_{ij} на величину

$$C'_{ij} = \begin{cases} C_{ij}, & j \neq i_{\text{start}}, \\ pC_{ij}, & j = i_{\text{start}}, \end{cases}$$

где $p \gg 1$ — коэффициент штрафа.

С учетом этой замены разностный аналог уравнения (4) имеет вид

$$u_{ij}^{T+1} - u_{ij}^{T} = \Delta t\left[-\eta\left(\sum_{k=1}^{n} v_{ik}^{T} + \sum_{l=1}^{n} v_{lj}^{T} - 2\right) - \lambda C'_{ij} \exp\Big(-\frac{T}{\tau}\Big)\right] \tag{5}$$

и решается итерационным методом, пока не будет выполнено условие

$$\sum_{k=1}^{n} v_{ik}^{T} + \sum_{l=1}^{n} v_{lj}^{T} - 2 < \varepsilon,$$

где T — номер итерации, ε — заданная точность выполнения ограничений (2), Δt — временной шаг. Далее выполняется преобразование полученной матрицы решения $\|v_{ij}\|$ [14, 15] с использованием принципа WTA (“Winner takes all”):

- $i = i_{\text{start}}$;
- в i-й строке матрицы отыскивается максимальный элемент $v_{ij_{\max}}$, $j_{\max}$ — номер столбца с максимальным элементом; выполняется преобразование $v_{ij_{\max}} = 1$, все остальные элементы строки i и столбца с номером $j_{\max}$ обращаются в нуль;
- далее происходит переход к строке $j_{\max}$ и описанные действия повторяются, пока не произойдет возврат к строке i_{start}, что будет означать завершение построения цикла.

Если возврат к строке i_{start} произошел раньше, чем в матрице $\|v_{ij}\|$ значение 1 получили n элементов, то это означает, что длина построенного цикла меньше n. В этом случае все описанные действия повторяются.

Полученный в результате цикл оптимизируется с использованием локального поиска 2-opt. Для удобства далее описанный здесь алгоритм обозначим как NWTA.

4. Последовательная реализация алгоритма. Метод частичных сумм

В [15] показано, что рекуррентная сеть Хопфилда–Вана дает хорошие результаты при решении системы уравнений (5) методом Зейделя. Нетрудно видеть, что решение задачи (1)–(3) связано с многократным вычислением одних и тех же частичных сумм, входящих в суммы (2).

Во избежание избыточных вычислений произведем расчет частичных сумм (индекс T опустим):

– "вертикальные" частичные суммы:

$$V_{ij} = \sum_{k=i}^{n} v_{kj}, \quad V'_{ij} = \sum_{k=1}^{i} v'_{kj}, \quad i, j = 1, \ldots, n,$$

– "горизонтальные" частичные суммы:

$$H_{ij} = \sum_{l=j}^{n} v_{il}, \quad H'_{ij} = \sum_{l=1}^{j} v'_{il}, \quad i, j = 1, \ldots, n,$$

где v'_{ij} — обновленное по Зейделю значение v_{ij}.

Пусть $s_{ij} = \sum_{k=1}^{n} v^*_{ik} + \sum_{l=1}^{n} v^*_{lj} - 2$, где массив v^* включает элементы v_{ki} при $k = i$, $l = j, \ldots, n$ и $k = i+1, \ldots, n$, $l = 1, \ldots, n$ и обновленные по Зейделю элементы v'_{kl} при $k = 1, \ldots, i-1$, $l = 1, \ldots, n$ и $k = i$, $l = 1, \ldots, j-1$.

Тогда в (5) получаем

$$s_{11} = V_{11} + H_{11}. \tag{6}$$

Используя (6), по формуле (5) вычисляем новое значение v'_{11} и полагаем

$$V'_{11} = H'_{11} = v'_{11}.$$

Для остальных элементов первой строки матрицы $\|v_{ij}\|$ выполняем следующие вычисления.

1. Вычисляем $s_{1j} = H'_{1,j-1} + V_{1j} + H_{1j}$, $j = 2, \ldots, n$.
2. Используя s_{1j}, вычисляем v'_{1j} (итерация (5) по Зейделю) и полагаем:

$$V'_{1j} = v'_{1,j}, \quad H'_{1j} = v'_{1,j} + H'_{1,j-1}, \quad j = 2, \ldots, n.$$

Для остальных строк с номерами $i = 2, \ldots, n$ при $j = 1, \ldots, n$ выполняем следующие вычисления.

1. Вычисляем:

$$s_{ij} = V'_{i,j-1} + V_{ij} + H_{ij}, \quad j = 1, \ldots, n,$$
$$s_{ij} = s_{ij} + H'_{i,j-1}, \quad j = 2, \ldots, n.$$

2. Используя s_{ij}, вычисляем v'_{ij} (итерация (5) по Зейделю) и полагаем:

$$V'_{ij} = v'_{ij} + V'_{i-1,j}, \quad H'_{ij} = v'_{1j}, \quad j = 1, \ldots, n, \quad H'_{ij} = H'_{ij} + H'_{1,j-1}, \quad j = 2, \ldots, n.$$

Метод частичных сумм позволяет сократить время решения системы (5) при построении гамильтоновых циклов в графах распределенных вычислительных систем с $O(n^3)$ до $O(n^2)$. Показано [16], что использование метода частичных сумм при решении системы (5) позволяет сократить время построения гамильтонова цикла в трехмерном торе в десятки раз.

5. Тестирование алгоритма

Предложенный выше алгоритм NWTA тестировался на примерах задачи коммивояжера из библиотеки TSPLIB [5]. Для каждого примера выполнено 10 запусков алгоритма. Начальные значения u_{ij}^0, $i, j = 1, \ldots, n$, — случайные равномерно распределенные в интервале $(-0.5, 0.5)$, параметры системы уравнений (5): $\beta = 0.1$, $\eta = 10$, $\lambda = 1$, $\tau = 10^3$, $p = 10^6$.

В табл. 1 представлены результаты тестирования алгоритма на центральном процессоре. Число в названии примера равно числу городов n в задаче коммивояжера. Здесь: $t_{\min}$, t_{aver} и $t_{\max}$ — минимальное, среднее и максимальное время выполнения алгоритма в секундах соответственно; $D_{\min}$, D_{aver} и $D_{\max}$ — минимальная, средняя и максимальная длина найденного маршрута соответственно.

Таблица 1. Решение задачи коммивояжера алгоритмом NWTA на центральном процессоре

Пример	$t_{\min}$	t_{aver}	$t_{\max}$	$D_{\min}$	D_{aver}	$D_{\max}$
eil51	0.00620	0.00622	0.00631	445.2	445.2	445.2
lin105	0.00964	0.00975	0.0103	14765	14765	14765
ch130	0.0198	0.01983	0.0201	6598	7164	7227
d198	0.0464	0.0467	0.0477	16570	16570	16570
a280	0.139	0.141	0.1414	2753	2799	2830
lin318	0.0904	0.0915	0.0944	44486	44512	44731
pcb442	0.178	0.1798	0.1839	55304	55982	56434
pr1002	0.943	0.963	0.992	281647	284750	289144
u1432	1.898	1.924	1.953	166207	167182	169248
u2152	5.761	5.848	5.983	71891	72552	73096
pr2392	5.396	5.493	5.604	404497	409285	413209

Случайный выбор начальных значений активаций u_{ij}, $i, j = 1, \ldots, n$, нейронов приводит к изменению результатов решения задачи от одного запуска алгоритма к другому. Важно знать, каков разброс результатов работы алгоритма. Из табл. 1 следует, что на рассмотренных примерах $\sigma_t = (t_{\max} - t_{\min})/t_{\min}$ не превышает 7% величины $t_{\min}$, а $\sigma_D = (D_{\max} - D_{\min})/D_{\min}$ не превышает 10% величины $D_{\min}$. Для оценки качества NWTA выполнено сравнение этого алгоритма с эвристическим алгоритмом LKH [2–4], который позволил построить оптимальные маршруты для всех примеров библиотеки TSPLIB. В экспериментах использована реализация [4] алгоритма LKH.

В табл. 2 представлены: $t_{\min}$, t_{aver} и $t_{\max}$ — минимальное, среднее и максимальное время выполнения алгоритма LKH в секундах соответственно; D_{opt} — соответствующие длины оптимальных маршрутов, полученных алгоритмом LKH.

Рис. 1 показывает, что начиная с примера ch130 ($n = 130$) алгоритм NWTA существенно ускоряет решение задачи коммивояжера по сравнению с алгоритмом LKH. На рис. 2 представлено максимальное относительное увеличение длины маршрута при использовании алгоритма NWTA.

Таблица 2. Время решения задачи коммивояжера алгоритмом LKH

Пример	$t_{\min}$	t_{aver}	$t_{\max}$	D_{opt}
eil51	0	0.01	0.02	426
lin105	0	0.01	0.02	14379
ch130	0.06	0.09	0.11	6110
d198	0.39	0.49	0.59	15780
a280	0.42	0.51	0.63	2579
lin318	0.39	0.69	0.97	42029
pcb442	2.16	2.21	2.3	50778
pr1002	7.27	7.70	8.16	259045
u1432	21.53	22.28	23.38	152970
u2152	72.20	76.36	79.11	64253
pr2392	132.22	135.43	141.33	378032

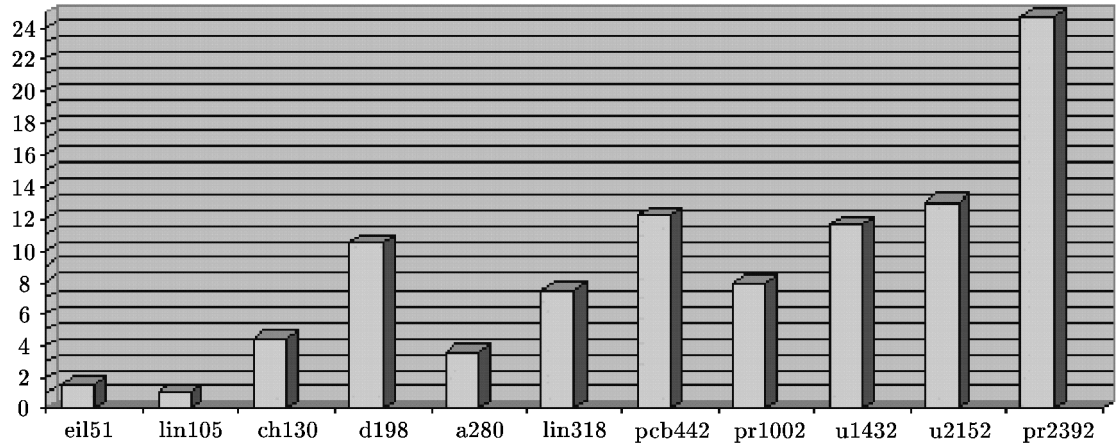

Рис. 1. Отношение $t_{\text{aver}}(\text{LKH})/t_{\text{aver}}(\text{NWTA})$

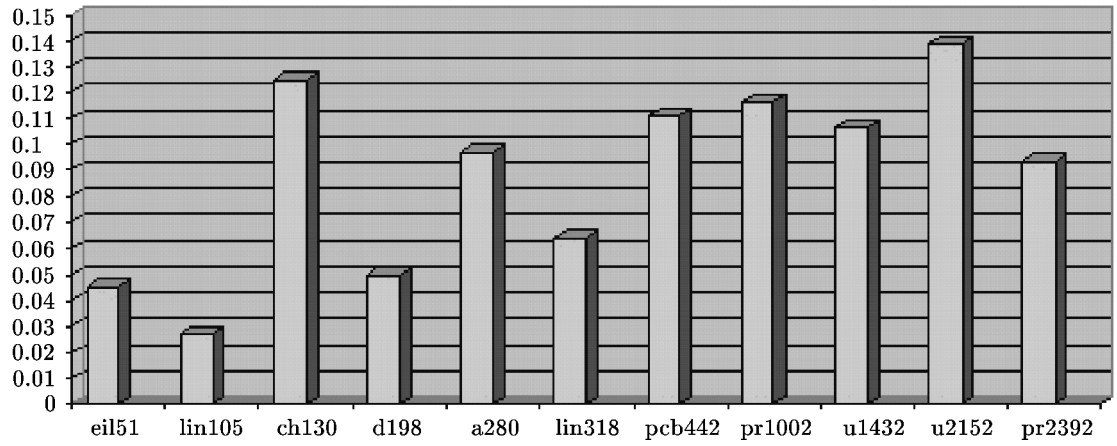

Рис. 2. Относительное увеличение длины маршрута

В табл. 3 показаны результаты решения задачи коммивояжера алгоритмом NWTA без использования метода частичных сумм. Здесь t — время решения задачи в секундах, D — длина найденного маршрута. Из сравнения таблиц 2 и 3 следует, что алгоритм частичных сумм существенно сокращает время решения задачи коммивояжера (при числе городов более 1000 на один-два порядка).

Таблица 3. Решение задачи коммивояжера алгоритмом NWTA на центральном процессоре без использования метода частичных сумм

Пример	t	D
eil51	0.0227	474
lin105	0.0283	15325
ch130	0.0947	6724
d198	0.379	16774
a280	1.823	2804
lin318	0.723	46171
pcb442	2.095	56549
pr1002	25.56	286490
u1432	170.349	166199
u2152	866.578	72530
pr2392	399.427	410106

На рис. 3 приведен построенный алгоритмом NWTA маршрут для примера u2152 из библиотеки TSPLIB.

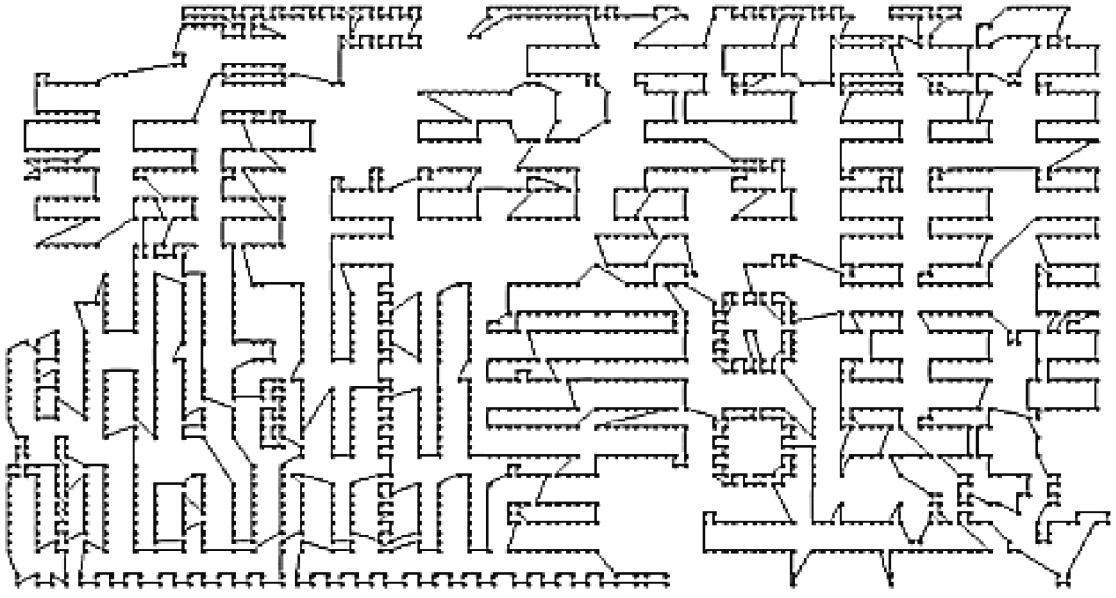

Рис. 3. Маршрут для примера u2152

6. Распараллеливание алгоритма NWTA

В работе выполнено частичное распараллеливание алгоритма NWTA на GPU (Graphic Processing Unit) с использованием технологии CUDA [17]. CUDA (Compute Unified Device Architecture) — это интегрированная среда, позволяющая разрабатывать

программы на языках C и/или C^{++}, которые запускают параллельное исполнение специальных функций ядра на графической карте, поддерживающей технологию CUDA. Эта графическая карта в данном контексте называется устройством, а компьютер, на котором установлено устройство, называется хостом (host). Функции ядра выполняются параллельно нитями, которые объединяются в блоки одинакового размера. Блоки и нити внутри блоков формируют сетку, которая в ходе выполнения функции отображается на мультипроцессоры GPU (Graphic Processing Unit) и их (скалярные) процессоры соответственно.

При выборе этапов алгоритма NWTA для отображения на GPU учтены следующие факторы, снижающие эффект использования GPU.

1. Выделение памяти на устройстве и передача данных между хостом и устройством требуют значительных временных затрат.
2. Тактовая частота устройства NVIDIA Geforce 9600 (650 МГц) значительно ниже тактовой частоты центрального процессора Intel Core Duo (2500 МГц).
3. Устройство NVIDIA Geforce 9600 не поддерживает тип данных double, что также увеличивает время выполнения алгоритма.

Таким образом, перенос алгоритма с хоста на устройство может дать положительный эффект лишь в том случае, когда он достаточно хорошо распараллеливается, т. е. когда время его параллельного выполнения на устройстве в сумме со временем передачи данных между хостом и устройством будет меньше времени его выполнения на центральном процессоре.

На графическом процессоре реализованы следующие этапы алгоритма NWTA.

1. Вычисление начальных состояний нейронов v_{ij}, $i, j = 1, \ldots, n$.
2. Вычисление начальных значений вертикальных частичных сумм V_{ij}, $i, j = 1, \ldots, n$.
3. Вычисление начальных значений горизонтальных частичных сумм H_{ij}, $i, j = 1, \ldots, n$.

Таблица 4. Решение задачи коммивояжера алгоритмом NWTA с использованием GPU

Пример	$t_{\min}$	t_{aver}	$t_{\max}$	$D_{\min}$	D_{aver}	$D_{\max}$
eil51	0.0361	0.0372	0.0386	467.18	467.18	467.18
lin105	0.0391	0.0404	0.0415	15115.00	15115.00	15115.00
ch130	0.042	0.0435	0.0464	6771.00	6946.00	6965.00
d198	0.0518	0.0527	0.0557	16419.00	16419.00	16419.00
a280	0.0659	0.0673	0.0693	2761.00	2830.00	2838.00
lin318	0.0762	0.0775	0.0791	46246.00	46371.00	46385.00
pcb442	0.116	0.119	0.124	56872.00	57261.00	57844.00
pr1002	0.471	0.488	0.520	282087.00	285276.00	287992.00
u1432	0.905	0.963	1.065	167126.00	169111.00	171114.00
u2152	2.086	2.277	2.488	71925.00	72891.00	73757.00
pr2392	2.646	2.738	2.976	406134.00	410320.00	415012.00

В табл. 4 приведены результаты распараллеливания алгоритма NWTA. Из табл. 4 следует, что на рассмотренных примерах $\sigma_t = (t_{\max} - t_{\min})/t_{\min}$ не превышает 20% величины $t_{\min}$, а $\sigma_D = (D_{\max} - D_{\min})/D_{\min}$ не превышает 2.6% величины $D_{\min}$. На рис. 4 приведено сравнение среднего времени t_{aver}(CPU) выполнения алгоритма NWTA на центральном процессоре и времени t_{aver}(CPU+GPU) выполнения распараллеленного варианта NWTA. Из рис. 4 следует, что в рассмотренных примерах распараллеливание дает положительный эффект (t_{aver}(CPU)/t_{aver}(CPU + GPU)>1), когда число городов не меньше 280.

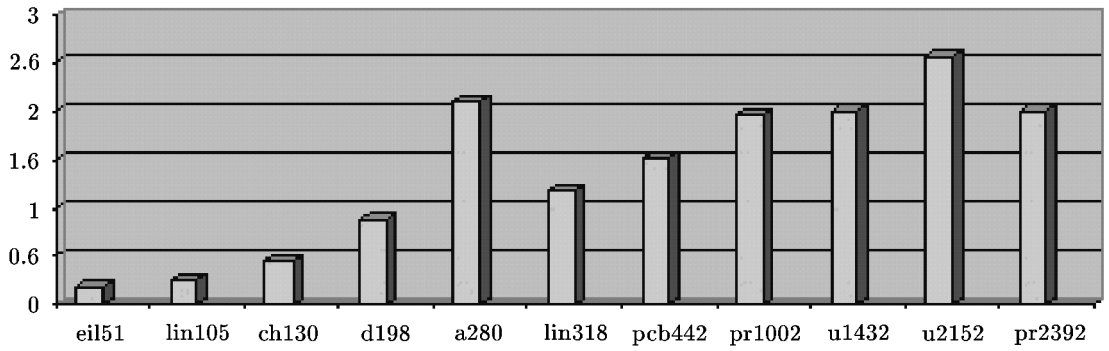

Рис. 4. Отношение $t_{\text{aver}}(\text{CPU})/t_{\text{aver}}(\text{CPU} + \text{GPU})$

7. Заключение

Предложен и исследован алгоритм NWTA приближенного решения задачи коммивояжера, основанный на использовании рекуррентной нейронной сети Хопфилда–Вана, модифицированного метода WTA и 2-opt. Для ускорения решения системы уравнений сети Хопфилда–Вана предложен метод частичных (префиксных) сумм. Показано, что при большом числе городов (130 и выше) алгоритм NWTA решает задачу коммивояжера существенно быстрее (на рассмотренных примерах из библиотеки TSPLIB до 24 раз), чем известный эвристический алгоритм LKH (алгоритм Лина–Кернигана–Гельсгауна), посредством которого удалось получить оптимальные маршруты для всех примеров библиотеки TSPLIB. При этом алгоритм NWTA увеличивает длину маршрута по сравнению с оптимальным на $3 \div 14\%$.

При использовании технологии CUDA удается получить дополнительное ускорение работы алгоритма NWTA (в рассмотренных примерах из библиотеки TSPLIB до 2.5 раз) при числе городов не менее 280.

Литература

1. **Lin S., Kernigan B.W.** An effective heuristic algorithm for the travelling salesman problem // Oper. Res. — 1973. — Vol. 21, iss 2. — P. 498–516.
2. **Helsgaun K.** An effective implementation of the Lin–Kernighan traveling salesman heuristic // European J. of Oper. Res. — 2000. — Vol. 126. — P. 106–130.
3. **Helsgaun K.** General k-opt submoves for the Lin–Kernighan TSP heuristic // Math. Prog. Comp. — 2009. — Vol. 1. — P. 119–163.
4. LKH Version 2.0.7 (November 2012). — http://www.akira.ruc.dk/ keld/research/LKH/lkh.exe
5. TSPLIB. — http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html
6. **Ефимов В.В., Козырев Г.И. и др.** Нейрокомпьютеры в космической технике. Кн. 17. / В.В. Ефимов — М.: Радиотехника, 2004.
7. **Hopfield J.J., Tank D.W.** “Neural” computation of decisions in optimization problems // Biological Cybernetics. — 1985. — Vol. 52, № 3. — P. 141–152.
8. **Smith K.A.** Neural networks for combinatorial optimization: a review of more than a decade of research // INFORMS J. on Computing. — 1999. — Vol. 11, № 1. — P. 15–34.

9. **Feng G., Douligeris C.** Using hopfield networks to solve traveling salesman problems based on stable state analysis technique // Proc. Int. Joint Conf. Neural networks. — 2000. — Vol. 6. — P. 521–526.

10. **Агеев А.Д., Балухто А.Н., Бычков А.В. и др.** Нейроматематика. Кн. 6. / А.И. Галушкин. — М.: ИПРЖР, 2002.

11. **Wang J.** Analysis and design of a recurrent neural network for linear programming // IEEE Trans. On Circuits and Systems-I: Fundamental Theory and Applications. — 1993. — Vol. 40, № 9. — P. 613–618.

12. **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.

13. **Siqueira P.H., Steiner M.T.A., and Scheer S.** A new approach to solve the travelling salesman problem // Neurocomputing. — 2007. — Vol. 70. — P. 1013–1021.

14. **Тарков М.С., Дугаров Г.А.** Параллельный алгоритм решения задачи коммивояжера с использованием рекуррентной нейронной сети // Проблемы информатики. — 2010. — № 2. — С. 4–9.

15. **Тарков М.С.** Построение гамильтоновых циклов в графах распределенных вычислительных систем рекуррентными нейронными сетями // Сиб. журн. вычисл. математики / РАН. Сиб. отд-ние. — Новосибирск, 2010. — Т. 13, № 4. — С. 467–475.

16. **Тарков М.С.** Об эффективности построения гамильтоновых циклов в графах распределенных вычислительных систем рекуррентными нейронными сетями // Управление большими системами. Выпуск 43. — М.: ИПУ РАН, 2013. — С. 157–171.

17. **Боресков А.В., Харламов А.А.** Основы работы с технологией CUDA. — М.: ДМК Пресс, 2011.

Поступила в редакцию 21 июля 2014 г.,
в окончательном варианте 19 августа 2014 г.

Литература в транслитерации

1. **Lin S., Kernigan B.W.** An effective heuristic algorithm for the travelling salesman problem // Oper. Res. — 1973. — Vol. 21, iss 2. — P. 498–516.

2. **Helsgaun K.** An effective implementation of the Lin–Kernighan traveling salesman heuristic // European J. of Oper. Res. — 2000. — Vol. 126. — P. 106–130.

3. **Helsgaun K.** General k-opt submoves for the Lin–Kernighan TSP heuristic // Math. Prog. Comp. — 2009. — Vol. 1. — P. 119–163.

4. LKH Version 2.0.7 (November 2012). — http://www.akira.ruc.dk/ keld/research/LKH/lkh.exe

5. TSPLIB. — http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

6. **Efimov V.V., Kozyrev G.I. i dr.** Nejrokomp'yutery v kosmicheskoj tekhnike. Kn. 17. / V.V. Efimov — M.: Radiotekhnika, 2004.

7. **Hopfield J.J., Tank D.W.** "Neural" computation of decisions in optimization problems // Biological Cybernetics. — 1985. — Vol. 52, № 3. — P. 141–152.

8. **Smith K.A.** Neural networks for combinatorial optimization: a review of more than a decade of research // INFORMS J. on Computing. — 1999. — Vol. 11, № 1. — P. 15–34.

9. **Feng G., Douligeris C.** Using hopfield networks to solve traveling salesman problems based on stable state analysis technique // Proc. Int. Joint Conf. Neural networks. — 2000. — Vol. 6. — P. 521–526.

10. **Ageev A.D., Balukhto A.N., Bychkov A.V. i dr.** Nejromatematika. Kn. 6. / A.I. Galushkin. — M.: IPRZhR, 2002.

11. **Wang J.** Analysis and design of a recurrent neural network for linear programming // IEEE Trans. On Circuits and Systems-I: Fundamental Theory and Applications. — 1993. — Vol. 40, № 9. — P. 613–618.

12. **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.

13. **Siqueira P.H., Steiner M.T.A., and Scheer S.** A new approach to solve the travelling salesman problem // Neurocomputing. — 2007. — Vol. 70. — P. 1013–1021.

14. **Tarkov M.S., Dugarov G.A.** Parallel'nyj algoritm resheniya zadachi kommivoyazhera s ispol'zovaniem rekurrentnoj nejronnoj seti // Problemy informatiki. — 2010. — № 2. — S. 4–9.

15. **Tarkov M.S.** Postroenie gamil'tonovykh tsiklov v grafakh raspredelennykh vychislitel'nykh sistem rekurrentnymi nejronnymi setyami // Sib. zhurn. vychisl. matematiki / RAN. Sib. otd-nie. — Novosibirsk, 2010. — T. 13, № 4. — S. 467–475.

16. **Tarkov M.S.** Ob effektivnosti postroeniya gamil'tonovykh tsiklov v grafakh raspredelennykh vychislitel'nykh sistem rekurrentnymi nejronnymi setyami // Upravlenie bol'shimi sistemami. Vypusk 43. — M.: IPU RAN, 2013. — S. 157–171.

17. **Boreskov A.V., KHarlamov A.A.** Osnovy raboty s tekhnologiej CUDA. — M.: DMK Press, 2011.