2. Поттосин Ю.В., Шестаков Е.А. Компактные таблицы и их использование при декомпозиции систем полностью определенных булевых функций // Идентификация образов. Вып.2. – Минск: Ин-т техн. Кибернетики НАН Беларуси, 2001. – С. 107-120.
3. Pottosin Yu., Shestakov E. Decomposition of systems of completely specified Boolean functions using their compact table representation // Boolean Problems. 4th International Workshop. – Freiberg (Sachen), 2000. – P. 135-142
4. Поттосин Ю.В., Шестаков Е.А. Ортогонализация системы полностью определенных булевых функций // Логическое проектирование. Вып.5. – Минск: Ин-т техн. кибернетики НАН Беларуси, 2000. – С. 107-115.
5. Закревский А.Д. Алгоритмы синтеза дискретных автоматов. – М.: Наука, 1971. – 512 с.
6. Шнейдер А.А. Классификация и анализ эвристических алгоритмов раскраски вершин графа // Кибернетика. – 1984. – № 4. – С. 15-22.

УДК 681.3

Головко В.А., Трики Ч., Савицкий Ю.В.

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ФУНКЦИОНИРОВАНИЯ И ОБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ

ВВЕДЕНИЕ

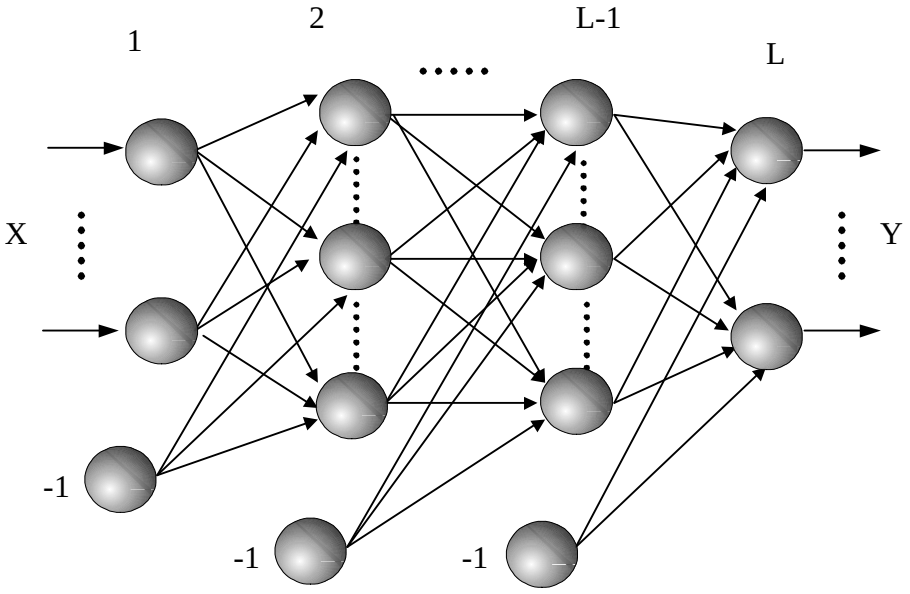
Один из основных факторов, определяющих увеличение темпов развития и эффективности практического использования интеллектуальных систем обработки информации в различных областях знаний, является наличие высокопроизводительных вычислительных средств и соответствующих алгоритмов обработки, позволяющих сократить временную сложность процессов обработки. Общий метод увеличения производительности – организация параллельной обработки, т.е. одновременное решение задач или совмещение во времени этапов решения одной задачи. Реализация метода возможна только при наличии нескольких обрабатывающих устройств в составе вычислительной системы, способных функционировать параллельно. При этом используются те или иные особенности задач или потоков задач, что позволяет осуществить тот или иной параллелелизм.

Многослойные нейронные сети с прямым распространением информации (Multilayer Perceptron, MLP) представляют собой регулярную структуру, состоящую из соединенных между собой синаптическими связями нейронных элементов, которые функционируют по единому алгоритму вычисления выходной активности вектора сигналов, поступающих на вход сети. Процесс обучения MLP по методу обратного распространения ошибки (Back Propagation Error, BPE [1]) также состоит из выполнения совокупности однотипных итерационных процедур модификации синаптических связей каждого нейроэлемента сети. Следует также отметить, что серьезным ограничением нейросетевых технологий является высокая вычислительная и временная сложность градиентных процедур обучения, зависящая как от размеров обучающих множеств, так и от количества нейронных элементов сети [1].

Очевидно, что исходя из структуры, функционирования и обучения существуют необходимость и условия для проектирования параллельных нейросетевых алгоритмов для многокомпонентных вычислительных архитектур.

1. ОБЩИЕ ПРИНЦИПЫ ПАРАЛЛЕЛИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ АЛГОРИТМОВ

Одним из наиболее распространенных типов параллелелизма является *параллелелизм независимых ветвей*. Суть его заключается в том, что при решении большой задачи могут быть выделены отдельные независимые части (ветви про-



Головко Владимир Адамович. *К.т.н., профессор каф. ЭВМ и систем Брестского государственного технического университета,зав. лабораторией Искусственных нейронных сетей.*
Трики Чефи. *Профессор университета г. Калабрии (Италия), факультет Электроники, информатики и систем.*
Савицкий Юрий Викторович. *К.т.н., доцент кафедры ЭВМ и систем Брестского государственного технического университета.*

граммы), которые при наличии нескольких обрабатывающих устройств могут обрабатываться параллельно и независимо друг от друга.

Между операторами, реализующими некоторый алгоритм обработки, существуют *зависимости по управлению и по данным*, обусловливающие порядок исполнения. Оператор может исполняться, если он указан в числе последующих одного из уже выполненных операторов (зависимость по управлению) и для него готовы входные операнды (зависимость по данным). Зависимость по данным может возникать, когда несколько операторов используют один и тот же операнд (регистр, ячейку памяти или какой-либо другой элемент памяти). Одни операторы используют этот элемент памяти для записи, другие — для чтения. Зависимость по данным как раз указывает на то, в каком порядке должны производиться операции записи и чтения совместно используемого операторами элемента памяти. В последовательной программе зависимости по данным реализуются через зависимости по управлению. Вследствие этого, а также того, что указание следующего оператора выполняется простой записью его за исполняемым, считается общепризнанным фактом, что последовательные программы более естественны, легче разрабатываются и отлаживаются [2].

Таким образом, распараллеливание последовательной программы предполагает разбиение этой программы на процессы P_i, $i=\{1, 2, ..., N\}$, где N — число процессов. Для каждого процесса имеется множество его входных данных I_i, и множество выходных (вырабатываемых процессом P_i) данных O_i. Два процесса P_1 и P_2 могут выполняться независимо и параллельно, если между ними нет зависимости по данным: $I_1 \cap O_2 = 0$ и $I_2 \cap O_1 = 0$. Эти условия гарантируют, что выходные данные процесса P_2 не могут затереть входные данные процесса P_1 и, соответственно, выходные данные процесса P_1 не могут затереть входные данные процесса P_1. Эти требования могут быть распространены на произвольное число процессов, которые могут выполняться независимо и параллельно [2].

2. АРХИТЕКТУРА И ФУНКЦИОНИРОВАНИЕ MLP

Рассматриваемая в работе архитектура MLP приведена на рисунке 1. Эта многослойная структура содержит 1 входной, 1 выходной и $L-2$ скрытых слоев нейроэлементов. Выходная активность нейронных элементов в случае использования сигмоидной функции активации определяется выражением:

$$g(S_j^{[l]}) = \left(1 + e^{-S_j^{[l]}}\right)^{-1}, \qquad (1)$$

где $S_j^{[l]}$ - взвешенная сумма входной активности рассматриваемого элемента:

$$S_j^{[l]} = \sum_{i=0}^{N^{[l-1]}} y_i^{[l-1]} w_{ij}^{[l]} . \qquad (2)$$

Активность нейронов выходного слоя L в ряде случаев (например, при использовании MLP в задачах аппроксимации, моделирования и прогнозирования временных рядов и хаотических процессов [2]) может определяться линейной функцией активации с коэффициентом K:

$$f(S_j^{[L]}) = KS_j^{[L]} . \qquad (3)$$

Фаза функционирования (*Forward Propagation, FP*) состоит из итеративного вычисления активности элементов, начиная от входного распространяющего слоя 1 до выходного слоя MLP согласно выражений (2) и (1) (либо (2) и (3) для слоя L в случае использования в выходном слое линейной активационной функции).

3. АЛГОРИТМ ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

Рассматриваемый далее в схемах параллелизации алгоритм обратного распространения ошибки, или BPE, кратко описан в работе в рамках следующих обозначений:

- $y_j^{[l]}$ - *выходная активность нейрона* j *слоя* l ;
- $w_{ij}^{[l]}$ - *взвешенная связь между нейроном* i *слоя* $l-1$ *и нейроном* j *в слое* l ;
- x^p - *входной вектор обучающего эталона* p *в множестве обучения* $T(X,D)$, $p=1,\ldots,P$;
- d^p - *соответствующий выходной вектор обучающего эталона* p *в множестве обучения* $T(X,D)$;
- L - *число слоев сети;*
- N^l - *количество нейронов в слое* l ;
- P - *количество эталонов обучения;*
- t - *текущая итерация обучающего алгоритма.*

В данных определениях $w_{0j}^{[l]}$ - порог нейронного элемента j слоя l ($y_0^{[l-1]} = -1$).

BPE реализует метод градиентного спуска с целью нахождения значений весовых коэффициентов, минимизирующих функцию квадрата ошибки согласно выражению:

$$E(t) = \sum_{p=1}^{P} E^p(t) = \sum_{p=1}^{P} \sum_{k=1}^{N^{[L]}} \left(y_k^{[L]}(t) - d_k^{[L]}\right)^2 . \qquad (4)$$

Весовые коэффициенты нейрона j слоя l для обучающего эталона p модифицируются (Updating, UPD) в соответствии со следующим итерационным правилом:

$$w_{ij}^{[l]}(t+1) = w_{ij}^{[l]}(t) - \alpha \frac{\partial E^p}{\partial w_{ij}^{[l]}}, \qquad (5)$$

где $\alpha > 0$ - шаг обучения;

$\frac{\partial E^p}{\partial w_{ij}^{[l]}}$ - градиент ошибки, определяемый выражением:

$$\frac{\partial E^p}{\partial w_{ij}^{[l]}} = \frac{\partial E^p}{\partial y_j^{[l]}} \frac{\partial y_j^{p,[l]}}{\partial S_j^{p,[l]}} \frac{\partial S_j^{p,[l]}}{\partial w_{ij}^{[l]}} = \gamma_j^{p,[l]} g'(S_j^{p,[l]}) y_i^{p,[l-1]} \qquad (6)$$

В выражении (6) $\gamma_j^{p,[l]}$ - ошибка нейронного элемента [1].

Перед обучением веса инициализируются небольшими случайными значениями. Обучающая итерация (5) выполняется для всех эталонов множества обучения, пока не будет достигнуто приемлемое значение погрешности (4). Очевидно, что отдельная фаза алгоритма BPE состоит в вычислении градиента ошибки (6) для модификации весов на каждой итерации t для каждого нейроэлемента сети для каждого эталона обучения.

Для конструирования параллельных алгоритмов обучения и функционирования MLP, наряду с константным шагом α (что характерно для классических схем BPE), в работе используется схема определения шага $\alpha^{p,[l]}(t)$, адаптируемого на каждой итерации обучения (*Adaptive Training Step,*

ATS). Полное описание метода приведено в [1]. В этом случае фаза ATS реализуется совместно с BPE.

4. СХЕМА ПОЭЛЕМЕНТНОЙ (ГРУППОВОЙ) ПАРАЛЛЕЛИЗАЦИИ В ПРЕДЕЛАХ ОДНОГО СЛОЯ

Анализ стадий функционирования (FP) и обучения (BPE, UPD) сети позволяет выделить такие группы НЭ, в пределах которых является возможным одновременное выполнение вышеуказанных операций. Такими группами являются наборы НЭ одного слоя MLP. Между отдельными нейронами одного слоя отсутствует зависимость по управлению и данным, так как результаты вычислений для НЭ одного слоя l зависят от данных слоев $l-1, l+1$ и не зависит от НЭ рассматриваемого слоя.

Общая идея предложенного алгоритма поэлементной параллелизации заключается в следующем. Процесс обучения, заключающийся в модификации весов НЭ для каждого из эталонов p, состоит в пределах одной итерации t из последовательности фаз:

- FP;
- BPE;
- UPD.

При выполнении каждой из них каждый НЭ слоя l выполняется на отдельном обрабатывающем модуле (Processing Module, PM). Затем результаты вычислений на стадии межмодульного обмена передаются всем PM системы, которые далее реализуют алгоритмы вычислений над НЭ следующего слоя. Количество PM должно быть не меньше, чем максимальное количество НЭ в одном из слоев сети: $K = max(N^l), l = 1,...,L$. По аналогичному принципу организуются фазы BPE и UPD алгоритма обучения, выполняемые после того, как на текущей итерации t вычислены выходные активности $Y[l]$ нейронов всех слоев l. На рисунке 2 изображена структура связей PM и наборы данных, необходимые для организации вычислений и межмодульного обмена. На рисунках 3–5 отражена временная последовательность выполнения FP, BPE и UPD соответственно. Здесь и далее цветом выделены фазы межмодульного обмена.

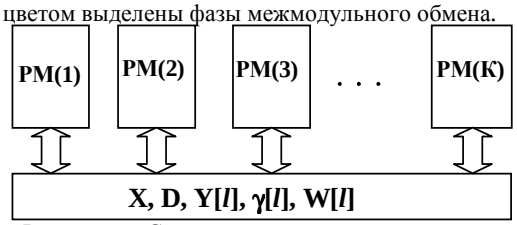


Рисунок 2 - Структура схемы параллелизации:

X/D – множества входных/выходных эталонов; $Y[l]$ – выходные значения нейронов слоя l; $\gamma[l]$ – ошибки нейронов слоя l, $W[l]$ – весовые коэффициенты слоя l.

Недостатками предложенной схемы является необходимость использования значительного количества PM при большом размере MLP, а также недостаточную загрузку PM из-за малой трудоемкости вычислений для FP, BPE и UPD, приходящихся на один НЭ. Поэтому более приемлемым решением в данном случае является модификация рассмотренного метода, названная в рамках работы - *групповая параллелизация в пределах одного слоя.*

Суть новой схемы состоит в том, что каждый PM при выполнении FP, BPE и UPD производит вычисления не над одним нейроном слоя l, а над *группой нейронов* рассматриваемого слоя. НЭ каждого слоя l разбиваются на K групп (K – количество PM). Каждый PM выполняет вычислительные действия при обучении и функционировании MLP над определенной группой нейронов. Поэтому структура этой схемы параллелизации будет аналогичной.

Достоинство метода заключается в возможности гибкого распределения вычислительной работы между имеющимися PM и адаптации параллельного алгоритма на существующую конфигурацию вычислительной системы.

5. КОНВЕЙЕРНАЯ СХЕМА ПОСЛОЙНОЙ ПАРАЛЛЕЛИЗАЦИИ

Описанный в разделе 4 параллелизм вычислительных алгоритмов для НЭ одного слоя можно отобразить на иную схему параллелизации, основанную на конвейерном принципе организации вычислений. Суть конвейерной обработки заключается в разбиении исходного алгоритма на последовательность стадий, реализуемых на отдельных PM, связанных между собой коммуникационной средой в вычислительный конвейер. Такая архитектура системы способна осуществлять алгоритм обработки над потоком данных, непрерывно поступающих на входной PM конвейера, обеспечивая непрерывную обработку данных и высокую загрузку обрабатывающих устройств.

В рассматриваемой схеме количество PM равняется количеству слоев MLP, над НЭ которых нужно выполнять вычисления: $K=L$. Каждый PM реализует вычисления FP, BPE и UPD над НЭ одного слоя. После выполнения цикла конвейера (когда все PM завершают свои вычисления для текущего эталона p) на стадии межмодульного обмена каждый PM передает результаты обработки следующему PM и принимает новые данные от предыдущего PM. При этом первый PM конвейера загружает в конвейер следующий эталон $p+1$ обучающей выборки. Процесс продолжается до тех пор, пока не будут обработаны все эталоны из множества обучения. Структура описанной схемы и информационных потоков приведена на рисунке 6. Процесс вычислений для этапов FP, BPE и UPD приведен соответственно на рисунке 7, рисунке 8 и рисунке 9.

PM(1)	**FP(1) p=1**		**FP(2) p=1**		**FP(3) p=1**	**…**	**FP(1) p=2**		**FP(2) p=2**		**FP(3) p=2**	**…**
PM(2)	**FP(1) p=1**		**FP(2) p=1**		**FP(3) p=1**	**…**	**FP(1) p=2**		**FP(2) p=2**		**FP(3) p=2**	**…**
					…							
PM(K)	**FP(1) p=1**		**FP(2) p=1**		**FP(3) P=1**	**…**	**FP(1) p=2**		**FP(2) p=2**		**FP(3) p=2**	**…**
Фазы вычислений	**1**		**2**		**3**	**...**	**1**		**2**		**3**	**...**
Фазы межмодульного обмена		**1**		**2**		**...**		**1**		**2**		**...**

T

Рисунок 3 - Выполнение фазы FP с использованием вычислительных модулей PM(1) – PM(K).

PM(1)	**…**	**BPE(3) p=1**		**BPE(2) p=1**		**BPE(1) p=1**		**…**	**BPE(2) p=2**		**BPE(1) p=2**	**…**
PM(2)	**…**	**BPE(3) p=1**		**BPE(2) p=1**		**BPE(1) p=1**		**…**	**BPE(2) p=2**		**BPE(1) p=2**	**…**
					…							
PM(K)	**…**	**BPE(3) p=1**		**BPE(2) p=1**		**BPE(1) p=1**		**…**	**BPE (2) p=2**		**BPE(1) p=2**	**…**
Фазы вычислений	**…**	**3**		**2**		**1**		**...**	**2**		**1**	**…**
Фазы межмодульного обмена	**…**		**3**		**2**		**1**	**...**		**2**		**…**

T

Рисунок 4 - Выполнение фазы BPE с использованием вычислительных модулей PM(1) – PM(K).

PM(1)	**UPD(1) P=1**		**UPD(2) p=1**		**UPD(3) p=1**	**…**	**UPD(1) p=2**		**UPD(2) p=2**	**…**
PM(2)	**UPD(1) P=1**		**UPD(2) p=1**		**UPD(3) p=1**	**…**	**UPD(1) p=2**		**UPD(2) p=2**	**…**
					…					
PM(K)	**UPD(1) P=1**		**UPD(2) p=1**		**UPD(3) p=1**		**UPD(1) p=2**		**UPD(2) p=2**	**…**
Фазы вычислений	**1**		**2**		**3**	**...**	**1**		**2**	**...**
Фазы межмодульного обмена		**1**		**2**		**...**		**1**		**...**

T

Рисунок 5 - Выполнение фазы UPD с использованием вычислительных модулей PM(1) – PM(K).

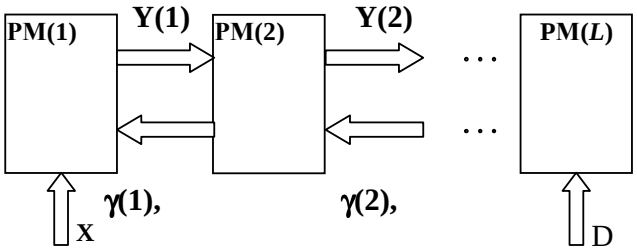


Рисунок 6 - Структура схемы параллелизации: $Y(l)$ – выходные значения нейронов слоя l, $\gamma(l)$ – ошибка нейронов слоя l, $W(l)$ – весовые коэффициенты слоя l.

Данная схема параллелизации реализует один из вариантов группового метода обучения – изменение весовых коэффициентов нейронных элементов происходит только после того, как будут вычислены выходные значения нейронных элементов и их ошибки для всех эталонов обучающего множества. Изменение весов происходит с учетом каждого эталона.

6. СХЕМА ГРУППОВОЙ ПАРАЛЛЕЛИЗАЦИИ НА УРОВНЕ МНОЖЕСТВА ОБУЧЕНИЯ

Предлагаемая схема основана на существующем в алгоритме обучения *параллелизме данных* – эталонов обучающего множества. Главная идея заключается в разделении обучающего набора эталонов T на группы $T(1), T(2), \ldots, T(K)$ и использовании K копий MLP для обработки каждой из групп. Каждая MLP обучается на своем обучающем подмножестве. В процессе обучения на периодических стадиях межмодульного взаимодействия реализуется обмен весовыми коэффициентами НЭ сети.

Соответствующие веса НЭ складываются между собой в каждой MLP. В результате выполняется обучение на основе всего набора эталонов T. Очевидно, что при организации параллельного вычислительного процесса может быть использована любая из рассмотренных ранее схем параллелизации. Наиболее просто вычислительную систему можно организовать в виде K обрабатывающих модулей, соединенных коммуникационной средой. В каждом из них реализуются стадии FP, BPE, UPD для собственной копии MLP. На стадиях межмодульного обмена выполняется обмен весовыми коэффициентами нейронных элементов между PM вычислительной системы.

7. РЕАЛИЗАЦИЯ СХЕМ ПАРАЛЛЕЛИЗАЦИИ С ИСПОЛЬЗОВАНИЕМ MPI

Для организации эффективных параллельных приложений для широкого круга вычислительных систем и комплексов в последнее время наиболее широко применяется современный интерфейс параллельного программирования Message Passing Interface (MPI). MPI – это большая библиотека функций для передачи сообщений, которая позволяет реализовывать приложения, полностью использующие параллельную архитектуру используемой вычислительной системы. MPI предоставляет единый механизм взаимодействия ветвей внутри парал-

PM(1)	**FP(1) p=1**		**FP(1) p=2**		**FP(1) p=3**		**FP(1) p=4**		**…**		
Выходная активность нейронов слоя 1											
PM(2)			**FP(2) p=1**		**FP(2) p=2**		**FP(2) p=3**		**FP(2) p=4**		**…**
Выходная активность нейронов слоя 2											
…											
Фазы вычислений	**1**		**2**		**3**		**4**		**5**		**…**
Фазы межмодульного обмена		**1**		**2**		**3**		**4**		**5**	

P

Рисунок 7 - Выполнение фазы прямого распространения сигнала, используя вычислительные модули PM(1) – PM(L).

PM(*l*)	**BPE (*l*) p=1**		**BPE(*l*) p=2**		**BPE(*l*) p=3**		**BPE(*l*) p=4**		**…**		
Выходная активность нейронов слоя 1											
PM(*l* -1)			**BPE(*l* -1) p=1**		**BPE(*l* -1) p=2**		**BPE(*l* -1) p=3**		**BPE(*l* -1) p=4**		**…**
Выходная активность нейронов слоя 2											
…											
Фазы вычислений	**1**		**2**		**3**		**4**		**5**		**…**
Фазы межмодульного обмена		**1**		**2**		**3**		**4**		**5**	**…**

P

Рисунок 8 - Выполнение фазы обратного распространения сигнала, используя вычислительные модули PM(1) –PM(L).

PM(*l*)	**UPD(*l*) p=1**		**UPD(*l*) p=2**		**UPD(*l*) p=3**		**UPD(*l*) p=4**		**…**		
Выходная активность нейронов слоя 1											
PM(*l* -1)			**UPD(*l*-1) p=1**		**UPD (*l*-1) p=2**		**UPD(*l*-1) p=3**		**UPD(*l*-1) p=4**		**…**
Выходная активность нейронов слоя 2											
…											
Фазы вычислений	**1**		**2**		**3**		**4**		**5**		**…**
Фазы межмодульного обмена		**1**		**2**		**3**		**4**		**5**	

T

Рисунок 9 - Выполнение фазы изменения весовых коэффициентов и порогов нейронных элементов, используя вычислительные модули PM(1) – PM(L).

лельного приложения независимо от машинной архитектуры (однопроцессорные/многопроцессорные с общей памятью / раздельной памятью), взаимного расположения ветвей (на одном процессоре / на разных процессорах) [2]. Исходя из этого, для реализации описанных в работе схем параллелизации была использована библиотека MPI for Windows, позволяющая разрабатывать параллельные приложения в системе программирования C++. В качестве PM для организации па-

раллельных вычислений были использованы ПЭВМ Pentium, работающие под управлением операционной системы Windows NT и объединенные в комплекс локальной вычислительной сетью Ethernet. Разработанные консольные приложения схем параллелизации тестировались на основе задач аппроксимации и прогнозирования с большими размерностями обучающих множеств (более 2000 эталонов). Эксперименты продемонстрировали как высокую перспективность использования параллельных вычислений в нейросетевых технологиях, так и необходимость применения более совершенных специализированных технических средств для реализации задач подобного рода.

БЛАГОДАРНОСТИ

Авторы благодарят международную организацию INTAS, фонды фундаментальных исследований Республики Беларусь и Российской Федерации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Головко В. А. Нейроинтеллект: теория и применение. Организация и обучение нейронных сетей с прямыми и обратными связями. Брест: Изд. БПИ, 1999. – С. 264.
2. Корнеев В. В. Параллельные вычислительные алгоритмы. – М.: «Нолидж», 1999. – С. 320.

УДК 681.3

Димаков В.М

МОДЕЛЬ НЕЙРОННОЙ СЕТИ ДЛЯ РЕШЕНИЯ ЗАДАЧИ О КРАТЧАЙШЕМ МАРШРУТЕ: ВОПРОС СХОДИМОСТИ И ЕДИНСТВЕННОСТИ РЕШЕНИЯ

В данной статье описывается один из вариантов решения задачи о кратчайшем маршруте на базе предлагаемой модели нейронной сети. Описываемая архитектура модели позволяет решить задачу оптимальным образом на базе вычислительных машин с параллельной архитектурой. Дается обоснование работы модели и нахождения ею оптимального и единственного решения. Рассмотрен также вариант использования данной парадигмы нейронной сети для решения другой комбинаторной задачи.

1. ВВЕДЕНИЕ

Задача о кратчайшем маршруте является классической проблемой [1,2] и для нее существует достаточно красивых и оригинальных решений, как, например, алгоритм Дейкстры [2] и его модификации, метод динамического программирования [3] и т.д. Существует также вариант нейросетевого решения на базе сети Хопфилда [4,5], для которого требуется задание множества параметров, что делает его не совсем приемлемым [6]. Таким образом, описанные выше алгоритмы решения дают превосходные результаты, которые могут быть приемлемыми или неприемлемыми в зависимости от типа решаемых задач и способа реализации. Поэтому в [6] был предложен метод решения, предполагающий, что задача будет решаться преимущественно аппаратными средствами, реализующими данный вид модели нейронной сети, не требующей предварительного обучения. В [6] было также проведено тестирование работы данной модели в сравнении с алгоритмом Дейкстры и методом динамического программирования, которое показало, что предложенный вариант не уступает по качеству решения, но значительности проигрывает в скорости получения результата. Это связано с тем, что модель нейронной сети представляет собой вычислительную модель параллельной аналоговой архитектуры, требующей для моделирования на компьютере с одиночном потоком команд и одиночным потоком данных (SISD) значительных временных затрат. Недостатком вышеупомянутой статьи являлось то, что в ней не было дано обоснование, что модель предложенной нейронной сети всегда будет находить решение за конечное число шагов и это решение будет всегда единственным и оптимальным. Целью данной статьи является устранение данного недостатка и в ней будет показано, каким образом модель нейронной сети получает единственное и оптимальное решение.

2. ОБЩАЯ АРХИТЕКТУРА МОДЕЛИ

Общая архитектура модели нейронной сети представлена на рисунке 1 [6]. Из рисунка видно, что она имеет пирамидальную архитектуру, где верхний слой определяет одно из оптимальных решений, предложенных остальными слоями.

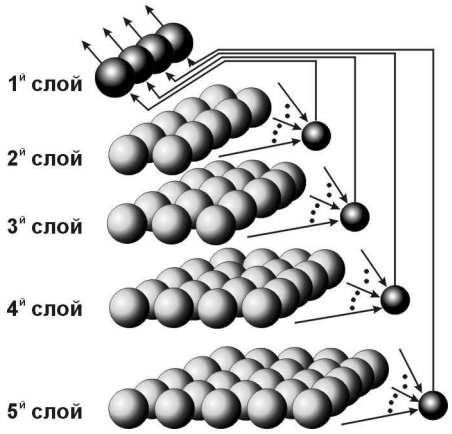


Рисунок 1 – Общая архитектура модели нейронной сети для задачи о кратчайшем маршруте.

На рисунке 2 показана архитектура верхнего слоя. Он представляет собой модификацию классической модели конкурентной нейронной сети [7], работающей по принципу «победитель берет все».

Димаков Валентин Михайлович. *Сотрудник фирмы Buck Engineering GmbH, инженер-программист в области автоматизации производства.*