УДК 004.032.26

НЕЙРОСЕТЕВОЙ МЕТОД РЕШЕНИЯ НЕЛИНЕЙНОЙ ЗАДАЧИ ОПТИМАЛЬНОГО РАСПРЕДЕЛЕНИЯ НЕОДНОРОДНОГО РЕСУРСА

[1]Жук А.А., [2]Булойчик В.М.

[1]*УО «ВА РБ», г. Минск, Республика Беларусь, k210@tut.by*
[2]*УО «ВА РБ», г. Минск, Республика Беларусь*

Реферат. Статья посвящена особенностям решения задачи целочисленного нелинейного программирования с помощью рекуррентной нейронной сети. Этот нейросетевой метод является разновидностью метода описанного в [1]. Рассмотрены теоретические аспекты его использования. Представлены результаты имитационного моделирования предлагаемой нейросети и оценки эффективности ее применения.

Abstract. Article is devoted features of the decision of a problem of integer nonlinear programming by means of a recurrent neural network. This neural network the method is a version of a method described in [1]. Theoretical aspects of its use are considered. Results of imitating modelling offered neural network and an estimation of efficiency of its application are presented.

На практике часто встречаются задачи, целью которых является поиск оптимального варианта распределения средств. Примерами таких задач являются: распределение огневых средств ПВО по средствам воздушного нападения противника для максимизации числа уничтоженных целей в налете; распределение неоднородных сил по районам действий для максимизации полной вероятности обнаружения цели и т. д. При этом целевая функция $U(x)$ и система ограничений таких задач имеют следующий вид

$$U(x)=\sum_{i=1}^{m} c_j\left[1-\prod_{j=1}^{n}(1-p_{ij}x_{ij})\right]\to\max, \tag{1}$$

$$\begin{cases}\sum\limits_{j=1}^{n} x_{ij}=1, \quad i=1,2,\ldots,m,\\ x_{ij}\in\{0,1\}, \quad i=1,2,\ldots,m, \quad j=1,2,\ldots,n,\end{cases} \tag{2}$$

где m, n – константы, предопределяющие размерность задачи (сложность);

c_j – константа, определяющая важность j-го мероприятия;

p_{ij} – значение показателя эффективности i-го средства при выполнении j-го мероприятия;

x_{ij} – параметр, принимающий значение 1, если i-е средство назначается для выполнения j-го мероприятия, и 0, если i-е средство не назначается.

Формализация задачи в виде целевой функции (1) и системы (2) представляет собой задачу целочисленного нелинейного программирования.

Для решения подобных задач вообще и задачи распределения в частности используются различные точные и приближенные методы комбинаторной оптимизации. В большинстве случаев, методом, гарантирующим нахождение оптимального решения, является полный перебор всех возможных вариантов. Однако множество вариантов допустимых решений таких задач быстро растет с увеличением размерности входных данных, что делает на практике неприемлемым использование метода полного перебора.

Тем не менее, во многих областях деятельности, и особенно в военной, часто необходимо оперативно решать задачи рассматриваемого класса. При этом приближенное решение задачи, полученное в приемлемое время, более ценно, чем точное решение, найденное через недопустимый интервал времени. Данное обстоятельство и стимулировало развитие различных приближенных методов решения комбинаторно-оптимизационных задач, среди которых наибольший интерес (с точки зрения рассматриваемой задачи) представляют нейросетевые методы.

В рамках военно-научной школы «Современные методы и средства математического моделирования военных действий и военно-технических систем» для решения комбинаторно-оптимизационных задач был разработан нейросетевой метод на основе рекуррентной нейронной сети (РНС) матричной архитектуры [1–3]. Для рассматриваемого метода наиболее близким по реализации является метод на основе нейронной сети Хопфилда [4]. Основные отличия в архитектуре данных сетей представлены в таблице 1.

Таблица 1 – Особенности нейронных сетей

Нейронная сеть Хопфилда	**Рекуррентная нейронная сеть матричной архитектуры**
Сигнал на входы нейронов с их выходов не подается	На входы нейронов подается сигнал с их выходов
Размерность решаемой задачи определяется симметричной матрицей $n \times n$	Размерность решаемой задачи определяется матрицей $m \times n$
При начальной инициализации весовых коэффициентов нейронов входной сигнал определяется значением вектора параметров x целевой функции решаемой задачи	При начальной инициализации на входы нейронов подаются дополнительные параметры, характеризующие ограничения решаемой задачи

Указанные в таблице 1 особенности позволяют:

на основе сети Хопфилда решать задачи комбинаторной оптимизации только с учетом ограничений, накладываемых на саму целевую функцию с симметричной матрицей эффективности;

на основе матричной РНС при решении задачи комбинаторной оптимизации дополнительно учитывать ограничения в виде системы линейных уравнений.

Архитектура матричной РНС, которая соответствует условиям (1) и (2) представлена на рисунке 1.

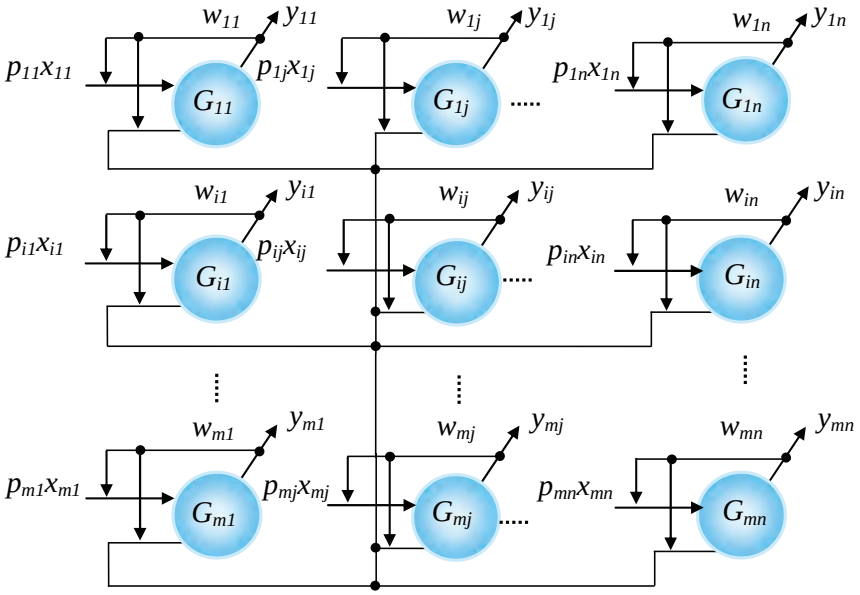
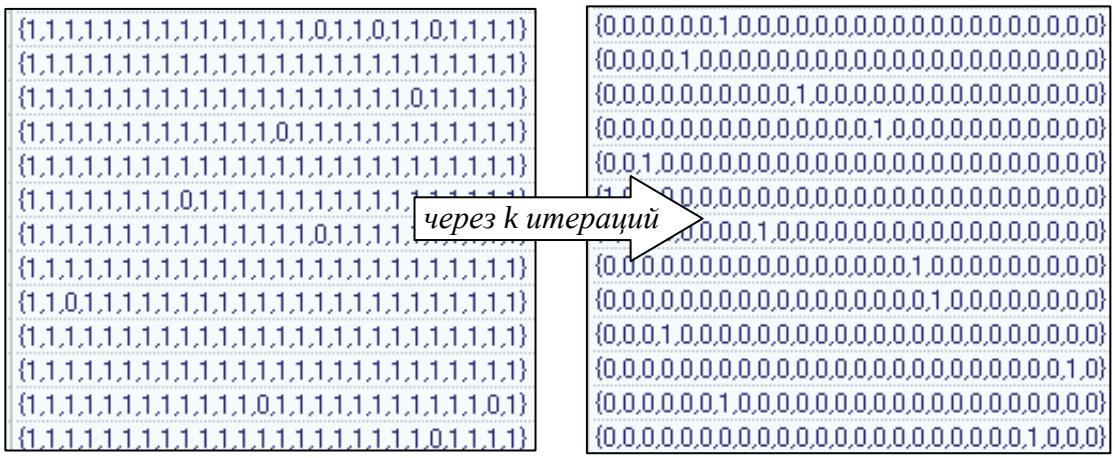


Рисунок 1 – Рекуррентная нейронная сеть матричной архитектуры

Решение задачи с помощью данной сети основано на установлении соответствия между функцией $E(w)$ вычислительной энергии РНС и целевой функцией (1). Выразив весовые коэффициенты w нейронов РНС через параметры x решаемой задачи, имеется возможность за время переходных процессов в сети найти квазиоптимальное решение. Пример перехода нейронов РНС в устойчивое состояние представлен на рисунке 2.



а) б)

Рисунок 2 – Состояние нейронов при переходе РНС в устойчивое состояние:
а) на первом шаге; б) на *k-м* шаге

На рисунке 2 представлены два фрагмента одной области (двумерный массив) оперативной памяти (ОП) ПЭВМ при отладке имитационной модели РНС в среде программирования Borland C++ Builder. Здесь состояния нейронов РНС обозначены 1 или 0. Первый фрагмент ОП (рисунок 2а) демонстрирует состояние нейронов РНС в начале переходного процесса. Через *k* итераций РНС приходит в устойчивое состояние (рисунок 2б). При этом полученные значения выходов y_{ij} активных нейронов (рисунок 1) образуют искомый результат целевой функции (1).

Функция энергии сети, минимизация которой соответствует целевой функции решаемой задачи, имеет вид

$$E(w)=\sum_{ux=1}^{m}\sum_{ui=1}^{n}\left\{\gamma_1\sum_{j=1}^{n}(w_{ux,j}-1)^2+\gamma_2\sum_{i=1}^{m}\sum_{j=1}^{n}(w_{i,j}-m)^2+\gamma_3\sum_{i=1}^{m}c_j\left[1-\prod_{j=1}^{n}(1-p_{ux,ui}w_{i,j})\right]\right\}\rightarrow\min, \qquad (3)$$

где γ_1, γ_2, γ_3 – положительные величины, определяемые эмпирически (порядок использования данных величин подробно рассмотрен в [4, 5]);

w_{ij} – значения синаптических связей нейронов сети.

В этом выражении первое слагаемое требует не более одной единицы в каждой строке матрицы, что соответствует ограничению (2). Второе слагаемое удовлетворяет требованию наличия ровно *m* единиц средств назначения в матрице распределения. Третье слагаемое соответствует целевой функции (1) задачи. При этом второе и третье слагаемые требуют наличия обратной связи «сам на себя».

Передаточные значения синаптических связей w_{ij} определяются в соответствии с выражением

$$w_{ij}^{k}=\frac{1+\tanh\left(\frac{y_{ij}^{k}}{u_0}\right)}{2}, \qquad (4)$$

где y_{ij}^{k} – значение состояния нейронов сети на *k*-й итерации;

u_0 – коэффициент, принимающий значения в диапазоне (0...1] и влияющий на скорость перехода РНС в устойчивое состояние и точность решения.

В этом случае формируемый РНС сигнал посредством нелинейной функции преобразуется в дискретный выходной сигнал, величина которого изменяется от 0 до 1 (рисунок 2).

Для исследования нейросетевого метода решения рассматриваемой задачи была разработана и реализована в среде программирования Borland C++ Builder имитационная модель РНС (рисунок 3).

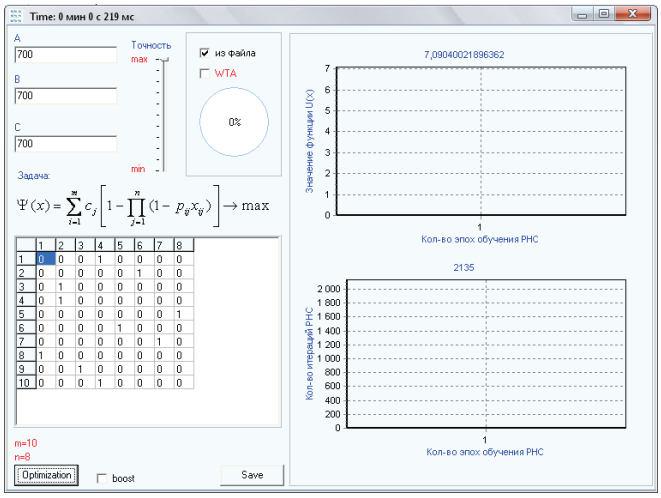


Рисунок 3 – Интерфейс программной реализации РНС матричной архитектуры

Для улучшения сходимости решения, на завершающих итерациях, в алгоритме предлагаемого нейросетевого метода дополнительно выполнялся расчет, при котором в активном состоянии ($y_{ij} = 1$) устанавливался нейрон с наибольшим выходным значением в строке, а остальные нейроны делались неактивными ($y_{ij} = 0$).

В качестве примера применения РНС рассматривалось решение следующей задачи. Имеется восемь районов поиска ($n = 8$), в одном из которых находится цель. Априорная вероятность нахождения цели c_j в каждом j-м районе равна единице ($c_1 = c_2 = ... = c_8 = 1$). Требуется распределить десять разнородных поисковых единиц ($m = 10$) по районам так, чтобы полная вероятность обнаружения цели была максимальной. При этом каждая из поисковых единиц должна обязательно назначаться на какой-либо из районов поиска. Вероятности обнаружения цели p_{ij} различными поисковыми единицами в каждом из районов заданы матрицей (рисунок 4).

	A	B	C	D	E	F	G	H	I	J
1		1	2	3	4	5	6	7	8	
2	1	0,58	0,12	0,82	0,86	0,18	0,78	0,03	0,64	
3	2	0,73	0,03	0,45	0,07	0,54	0,07	0,62	0,27	
4	3	0,63	0,47	0,06	0,74	0,86	0,02	0,55	0,18	
5	4	0,16	0,98	0,53	0,71	0,91	0,13	0,88	0,27	
6	5	0,79	0,12	0,57	0,03	0,84	0,96	0,06	0,09	
7	6	0,78	0,41	0,57	0,91	0,97	0,15	0,48	0,82	
8	7	0,80	0,70	0,42	0,24	0,08	0,16	0,26	0,94	
9	8	0,40	0,76	0,05	0,13	0,51	0,89	0,76	0,51	
10	9	0,64	0,77	0,00	0,61	0,74	0,30	0,78	0,84	
11	10	0,55	0,33	0,86	0,48	0,67	0,09	0,87	0,28	
12										
13		1	2	3	4	5	6	7	8	
14	1	0	0	1	0	0	0	0	0	1
15	2	1								1
16	3	0								1
17	4	0								1
18	5	0								1
19	6	0								1
20	7	0								1
21	8	0								1
22	9	0								1
23	10	0								1
24		1								
25										
26	7,3									
27										
28										
29										
30										

Рисунок 4 – Исходные данные и ограничения для поиска решения задачи комбинаторной оптимизации с помощью табличного процессора Microsoft Excel

На рисунке 4, в диапазоне ячеек [*B*2, *C*3, ..., *I*11] таблицы Microsoft Excel, записаны значения условной вероятности обнаружения цели p_{ij} *i*-ой поисковой единицей в *j*-м районе. В окне «Поиск решения» табличного процессора Microsoft Excel (рисунок 4) представлены максимизируемая целевая функция (1) и ограничения (2) задачи.

Оценка эффективности решения задачи предлагаемым нейросетевым методом выполнялась при различных исходных данных условной вероятности обнаружения p_{ij} ($p_{ij} \in [0 \div 1]$), а сложность задачи определялась размером 10×8 ($m \times n$).

В качестве показателей эффективности применения РНС использовались: средняя относительная ошибка и время решения задачи.

За точное решение принималось значение, полученное с помощью алгоритма нелинейной оптимизации средства «Поиск решения» табличного процессора Microsoft Excel (рисунок 4).

Результаты применения рассматриваемых методов для 14 первых реализаций случайно сгенерированных значений матриц условных вероятностей обнаружения p_{ij} представлены в таблице 2.

Таблица 2 – Результаты решения задачи нелинейной оптимизации

№ реализации	Методы решения			
	Алгоритм нелинейной оптимизации средства «Поиск решения» табличного процессора Microsoft Excel		Нейросетевой метод на основе РНС матричной архитектуры	
	Показатели решения задачи			
	Время решения задачи *t*, мс	Максимальное значение целевой функции *U*(*x*)	Время решения задачи *t*, мс	Максимальное значение целевой функции *U*(*x*)
1	1150	7,1	204	7,13
2	1150	7,4	641	7,33
3	2300	6,8	78	7,01
4	2300	7,16	125	7,11
5	2300	7,35	125	7,31
6	2300	7,55	125	7,44
7	2300	7,1	156	7
8	2300	7,1	641	6,98
9	2300	6,96	187	7,11
10	3600	7,09	219	7,06
11	2300	7,04	157	7,01
12	2300	7,33	250	7,23
13	2300	7,15	125	7,1
14	3400	7,194	406	7,296

Анализ таблицы 2 показывает, что среднее время решения задачи для нейросетевого метода составило 245 мс, а для средства «Поиск решения» табличного процессора Microsoft Excel – 2307 мс. Среднее значение целевой функции (1) для нейросетевого метода – 7,151, для метода нелинейной оптимизации приложения Microsoft Excel – 7,166. Таким образом, решение задачи целочисленного нелинейного программирования нейросетевым методом дает более быстрое решение. Его использование для задачи размерностью 10×8 позволило уменьшить время решения в 9,4 раза и обеспечить точность не менее чем 99,8 % (с относительной ошибкой не более 0,2 %). При этом, максимальная практически возможная ошибка ξ, допущенная с доверительной вероятностью 0,9 при определении средней ошибки решения, составила 7,7 %. Как показывают расчеты, для уменьшения максимальной практической до-

пущенной ошибки результатов исследования до 5 % требуется число примеров (реализаций) увеличить, как минимум в 2,5 раза.

Следует также отметить, что динамика РНС состоит в многократном циклическом пересчете матрицы весовых коэффициентов и заданных ограничений сети, при котором к каждому элементу матрицы применяется одинаковый набор процедур. Все это дает возможность реализации параллелизма и ускорения вычислений РНС при обработке данных на ПЭВМ. Например, такие вычисления могут быть распараллелены на графическом процессоре с применением технологии CUDA [6]. По предварительной оценке это позволит дополнительно сократить время решения задачи от 10 до 30 раз.

Список использованных источников

1. Способ динамической обработки данных при решении задачи комбинаторной оптимизации : пат. BY 21989 / А. А. Жук, В. М. Булойчик. – Опубл. 13.03.2018.

2. Жук, А. А. Оптимизация распределения ресурсов посредством рекуррентной нейронной сети / А. А. Жук, В. М. Булойчик // Вестн. Воен. акад. Респ. Беларусь. – 2016. – № 2. – С. 62–70.

3. Жук, А. А. Эффективность решения задачи распределения ресурсов искусственной нейронной сетью / А. А. Жук, В. М. Булойчик // Вестн. Воен. акад. Респ. Беларусь. – 2018. – № 2. – С. 26–32.

4. J. J. Hopfield, D. W. Tank. «Neural» Computation of Decisions in Optimization Problems // Biological Cybernetics. – 1985. – № 52. – P. 141–152.

5. Булойчик, В.М. Решение транспортной задачи на электронной карте местности с помощью искусственных нейронных сетей / В.М. Булойчик, Д.М. Скрипко // Весці НАН Беларусі. Сер. фіз.–тэхн. навук. – 2007. – №1.– С. 104–110.

6. Сандерс, Дж. Технология CUDA в примерах: введение в программирование графических процессоров / Дж. Сандерс, Э. Кэндрот. – М.: ДМК Пресс, 2013. – 232 с.