УДК 004.032.26:004.75

Гулаєва Н. М.

ЛОКАЛЬНІ АЛГОРИТМИ: ЗВ'ЯЗОК ЗІ ШТУЧНИМИ НЕЙРОННИМИ МЕРЕЖАМИ ТА ГЕНЕТИЧНИМИ АЛГОРИТМАМИ

У статті показано, що кожному локальному за Анісімовим алгоритму можна поставити у відповідність штучну нейронну мережу, яка реалізує цей алгоритм, і навпаки. Наведено приклади побудови таких алгоритмів та нейронних мереж. Показано також, що генетичні алгоритми на основі індивідів є локальними за Анісімовим алгоритмами.

Штучні нейронні мережі показали свою ефективність при розв'язуванні практично важливих задач, пов'язаних з обробкою інформації. Як зазначається в [1, 2], очевидною є наявність кількох джерел ідеології нейроінформатики: це уявлення про будову мозку та процеси навчання, що дають можливість розв'язувати слабо формалізовані задачі, та проблема ефективного паралелізму (адже нейроінформатика постачає універсальні дрібнозернисті паралельні архітектури для розв'язання різних класів задач). Огляд задач, що ефективно розв'язуються засобами нейроінформатики, а також існуючих реалізацій штучних нейронних мереж (і програмних, і апаратних), можна знайти, наприклад, у [2–4].

Обґрунтуванням експансії штучних нейронних мереж можуть служити теореми повноти для функцій, що обчислюються штучними нейронними мережами. Ці теореми доводились багатьма авторами [1, 5–8]. В даній роботі показується зв'язок між алгоритмами, що обчислюються штучними нейронними мережами, та локальними алгоритмами.

Формальне визначення локальних алгоритмів та дослідження їх властивостей було дано Ю. І. Журавльовим у [9] та розвинуто рядом авторів [10–12]. У цій роботі під локальним алгоритмом розумітимемо локальний алгоритм за Анісімовим. Нагадаємо основні поняття, введені А. В. Анісімовим у роботі [10].

Нехай D – деяка множина, $\tau : D \to 2^D$ – відображення множини D у множину всіх можливих підмножин множини D. Множину $\tau(d)$ називають *околом* елемента $d \in D$.

В елементах із D запам'ятовується деяка інформація з множини значень M. Інформаційне наповнення множини D задається функцією (взагалі кажучи, частково визначеною) $\mu : D \to M$.

Обчислювальне середовище S локального алгоритму – це множина $\{D, \Phi(D, M)\}$, де $\Phi(D, M)$ – множина всіх відображень D в M. *Стан* обчислювального середовища в кожен момент обробки задається парою (D, μ), де $\mu \in \Phi(D, M)$.

Через $\underline{Інф}(d)$ позначимо поточне значення з M, яке зберігається в елементі $d \in D$ у момент часу, що розглядається. Відповідно $\underline{Інф}(\tau(d))$ – набір значень із M, що зіставляються з елементами із $\tau(d)$.

Поряд з обчислювальною інформацією $\underline{Інф}(d)$ містить також виділене значення f_d, що задає функцію (процедуру, процес, алгоритм), яка зіставляється з елементом d. Функцію f_d визначено на $\underline{Інф}(\tau(d))$; вона набуває значення з M. Обчислення функції f_d в елементі d у певний момент часу t полягає у зміні поточного значення $\underline{Інф}(d)$ на $f_d(\underline{Інф}(\tau(d)))$. При цьому значення $\underline{Інф}(\tau(d))$ розглядаються взятими в момент часу t.

Локальний оператор $O(\mu)$, який застосовується до обчислювального середовища S у стані (D, μ), визначається як одночасне обчислення всіх функцій f_d в елементах $d \in D$.

Локальна умова, визначена на обчислювальному середовищі, задається як виконаність у будь-якому околі деякого локального предиката $p(\underline{Інф}(\tau(d)))$. Точніше, предикат $P(\mu)$ називається *локальним*, якщо існує предикат p, визначений на множинах $\underline{Інф}(\tau(d))$, такий, що $P(\mu) = \forall d\, p\left(\underline{Інф}\left(\tau(d)\right)\right)$; предикат p при цьому називається *зображуючим* для P.

Локальні алгоритми будуються за допомогою певних керуючих конструкцій з використанням локальних операторів та локальних умов як базових операцій. Зауважимо, що для завдання складних обчислень часто буває зручним застосування локальних операторів та локальних умов на елементах, що утворюють певну підмножину елементів з D.

Теорія штучних нейронних мереж, як зазначається в [4], досі перебуває на етапі формування, наслідком чого є розмаїття постановок проблем та основних означень. Автори монографії

[1] навіть порівнюють світ штучних нейронних мереж зі своєрідним «зоопарком».

У цій роботі дотримуватимемось таких формулювань основних понять теорії штучних нейронних мереж.

Кожен штучний нейрон (далі просто нейрон) j характеризується скалярною величиною net_j, що називається *станом* нейрона, та *функцією активації* $F_j(net_j)$, що вираховує значення вихідного сигналу нейрона залежно від величини його стану net_j. Передача сигналів між нейронами здійснюється за допомогою з'єднань з ваговими параметрами, що налаштовуються в процесі навчання нейронної мережі. Нехай кожне k-те з'єднання j-го нейрона характеризується величиною ваги w_{jk}. Нехай a_{jk} – величина активації, породженої k-м з'єднанням j-го нейрона. Ця величина визначається як *дендритне перетворення* $D\left(w_{jk}, \vec{x}_{jk}\right)$, де $\vec{x}_{jk}$ – вектор вихідних сигналів нейронних елементів, що беруть участь в утворенні k-го з'єднання j-го нейрона. Тоді стан нейрона net_j визначається через *функцію стану* $S\left(a_{j1}, a_{j2}, \ldots, a_{jl}\right)$, де l – кількість з'єднань j-го нейрона. Схему перетворення сигналів у нейроні дано на рис. 1.

Зауважимо, що в більшості випадків функція стану нейрона S (яка, фактично, є правилом комбінування вхідних сигналів) задається як звичайне арифметичне підсумовування величин активації кожного з'єднання, тобто

$$net_j = \sum_k D\left(w_{jk}, \vec{x}_{jk}\right) + \theta_j ,$$

де індекс k пробігає всі з'єднання j-го нейрона, θ_j – величина так званого зміщення, або порога, $\vec{x}_{jk}$ – вектор вихідних сигналів нейронних елементів, що беруть участь в утворенні k-го з'єднання j-го нейрона, w_{jk} – величина ваги k-го з'єднання j-го нейрона, D – дендритне перетворення.

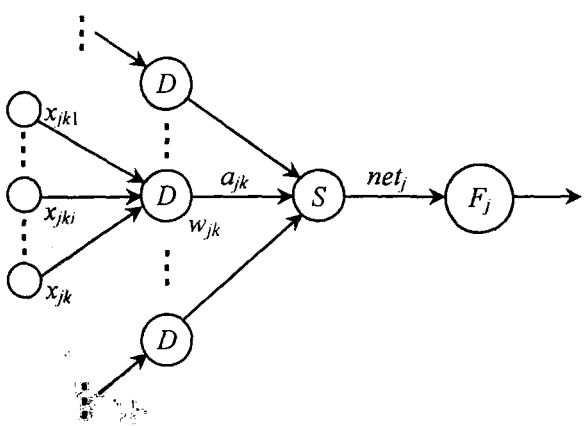

Рис. 1. Схема перетворення сигналів у нейронному елементі

Історично першим та найпоширенішим правилом комбінування вхідних сигналів є лінійне правило:

$$net_j = \sum_k w_{jk} x_{jk} + \theta_j . \qquad (1)$$

Нейронні елементи із вказаною функцією стану називаються Σ-*елементами*.

У [13] зазначається, що біологічно та математично виправданішим є полілінійне правило, згідно з яким

$$net_j = \sum_k w_{jk} \prod_{i \in I_{jk}} x_{jki} + \theta_j , \qquad (2)$$

де I_{jk} – множина нейронних елементів, що беруть участь в утворенні k-го з'єднання j-го нейрона. Нейронні елементи із вказаною функцією стану називаються Σ-Π-*елементами*. Зауважимо, що у випадку повторення деяких входів Σ-Π-елементів маємо вже не полілінійне, а поліноміальне правило.

Нейронна мережа утворюється шляхом з'єднання дугами виходів нейронів з їхніми входами. Вигляд графа міжнейронних з'єднань є однією з класифікаційних ознак нейронних мереж, що розділяє їх, у першу чергу, на мережі без циклів та циклічні. Кожен нейрон виконує досить просту функцію: отримує інформаційні сигнали від інших нейронів або зовнішніх джерел, формує вихідний інформаційний сигнал та передає його іншим нейронам. Легко бачити, що, прийнявши деякі погодження щодо тактування мережі (моментів часу спрацювання нейронів), отримаємо апарат для задання алгоритмів за допомогою нейронних мереж.

Теорема 1. *Функціонування будь-якої нейронної мережі можна подати у вигляді певного локального алгоритму.*

Доведення. Нехай задана нейронна мережа та вказані правила спрацювання нейронів; ваги міжнейронних з'єднань та вихідні сигнали нейронів набувають значення з деякої множини L. Нехай F_j – функція активації j-го нейрона. Тоді маємо:

$$F_j(net_j) = F_j(S(a_{j1}, a_{j2}, \ldots, a_{jn})) =$$
$$= F_j(S(D(w_{j1}, \vec{x}_{j1}), D(w_{j2}, \vec{x}_{j2}), \ldots, D(w_{jn}, \vec{x}_{jn}))), \quad (3)$$

де S – функція стану, D – дендритне перетворення, n – кількість з'єднань j-го нейрона, w_{jk} – величина ваги k-го з'єднання j-го нейрона, $\vec{x}_{jk}$ – вектор вихідних сигналів нейронних елементів, що беруть участь в утворенні k-го з'єднання j-го нейрона.

Побудуємо локальний алгоритм у такий спосіб.

Нехай D – множина нейронних елементів заданої нейронної мережі. Тоді елемент $d \in D$ є штучним нейроном, а його околом $\tau(d)$ є множина нейронів, зв'язаних з нейроном d у графі міжнейронних з'єднань. Виділимо в множині $\tau(d)$ підмножину $\tau^+(d)$ нейронів, що зв'язані з нейроном d вхідними дугами, тобто $d_i \in \tau^+(d)$, якщо d_i бере участь в утворенні певного з'єднання нейрона d.

Покладемо $M = L^{r+1}$, де $r = \max\limits_{d \in D} \left|\tau^+(d)\right|$. Нехай $\underline{Інф}(d)$ – вектор значень $(x_1, x_2, \ldots, x_r, y)$, де x_k задає вагу k-го з'єднання нейрона d, тобто

$$x_k = \begin{cases} w_{dk}, & якщо \quad k \le \left|\tau^+(d)\right|; \\ 0, & якщо \quad k > \left|\tau^+(d)\right|, \end{cases}$$

y – значення вихідного сигналу нейрона d після спрацювання функції активації.

Визначимо функцію $f_d\left(\underline{Інф}\left(\tau(d)\right)\right)$, що набуває значення з M, як функцію, що змінює лише $(r+1)$-шу (останню) координату вектора $\underline{Інф}(d)$. Через $\underline{Інф}(d)\big|_i$ позначимо i-ту координату вектора $\underline{Інф}(d)$ та покладемо

$$f_d(\underline{Інф}(\tau(d)))\Big|_{r+1} = F_d(S(D(x_1, d_{11}, \ldots, d_{1n_1}),$$
$$D(x_2, d_{21}, \ldots, d_{2n_2}), \ \ldots, \ D(x_n, d_{n1}, \ldots, d_{nn_m}))),$$

де F_d – функція активації нейрона d, S – функція стану, D – дендритне перетворення, n – кількість з'єднань нейрона d, d_{ij} – елементи, що беруть участь в утворенні i-го з'єднання нейрона d, $i = 1, \ldots, n$. Зауважимо, що виділення елементів d_{ij} серед усіх елементів $d \in \tau^+(d)$ легко реалізується шляхом введення додаткової інформації в $\underline{Інф}(d)$. З метою скорочення виклад деталей опускаємо.

Враховуючи домовленості щодо тактування нейронної мережі, задамо кілька локальних операторів $O_i(\mu)$ з різними областями визначення. Очевидно, що послідовне виконання цих операторів (де послідовність визначається згідно з домовленістю про порядок спрацювання нейронів у нейронній мережі, що розглядається) визначає локальний алгоритм, який задає роботу нейронної мережі, що розглядається.

Теорему доведено.

Приклад 1. Як приклад розглянемо нейронну мережу, що моделює відношення XOR. Граф міжнейронних з'єднань цієї мережі подано на рис. 2. Нейрони 1 та 2 тут є вхідними. Для решти нейронів використовується лінійне правило комбінування вхідних сигналів (1) та функція активації вигляду $F(net) = \begin{cases} 1, & якщо \quad net \ge 0; \\ 0, & якщо \quad net < 0. \end{cases}$

Побудуємо локальний алгоритм таким чином. Покладемо $D = \{d_1, d_2, d_3, d_4, d_5\}$, де елемент d_i зіставляється з i-м нейроном. Покладемо $\tau(d_1) = \tau(d_2) = \tau(d_5) = \{d_3, d_4\}$, $\tau(d_3) = \tau(d_4) = \{d_1, d_2, d_5\}$, $\tau^+(d_1) = \tau^+(d_2) = \varnothing$, $\tau^+(d_3) = \tau^+(d_4) = \{d_1, d_2\}$, $\tau^+(d_5) = \{d_3, d_4\}$. Як бачимо, $r = 2$.

На початку роботи алгоритму $\underline{Інф}(d_1) = (0, 0, y_1)$, $\underline{Інф}(d_2) = (0, 0, y_2)$, де y_1 та y_2 – вхідні значення; $\underline{Інф}(d_3) = \underline{Інф}(d_4) = (-1, -1, -\infty)$; $\underline{Інф}(d_5) = (1, -1, -\infty)$.

Розписавши функції активації нейронів згідно з (3), отримаємо:

$$f_{d_i}(\underline{Інф}(\tau(d_i)))\Big|_3 = \underline{Інф}(d_i)\Big|_3, \ i = 1, 2,$$

$$f_{d_3}(\underline{Інф}(\tau(d_3)))\Big|_3 =$$

$$= \begin{cases} 1, & якщо \quad 1{,}5 - \underline{Інф}(d_1)\Big|_3 - \underline{Інф}(d_2)\Big|_3 \ge 0; \\ 0, & інакше, \end{cases}$$

$$f_{d_4}(\underline{Інф}(\tau(d_4)))\Big|_3 =$$

$$= \begin{cases} 1, & якщо \quad 0{,}5 - \underline{Інф}(d_1)\Big|_3 - \underline{Інф}(d_2)\Big|_3 \ge 0; \\ 0, & інакше, \end{cases}$$

$$f_{d_5}(\underline{Інф}(\tau(d_5)))\Big|_3 =$$

$$= \begin{cases} 1, & якщо \quad \underline{Інф}(d_3)\Big|_3 - \underline{Інф}(d_4)\Big|_3 - 0.5 \ge 0; \\ 0, & інакше. \end{cases}$$

Нехай локальні оператори $O_1(\mu)$ та $O_2(\mu)$ мають області визначення $\{d_3, d_4\}$ та $\{d_5\}$, відповідно. Легко бачити, що локальний алгоритм

1 begin

2 $O_1(\mu)$;

3 $O_2(\mu)$;

4 end.

вираховує функцію XOR: в кінці роботи алгоритму $\underline{Інф}(d_5)\Big|_3 = \text{XOR}(\underline{Інф}(d_1)\Big|_3, \ \underline{Інф}(d_2)\Big|_3)$.

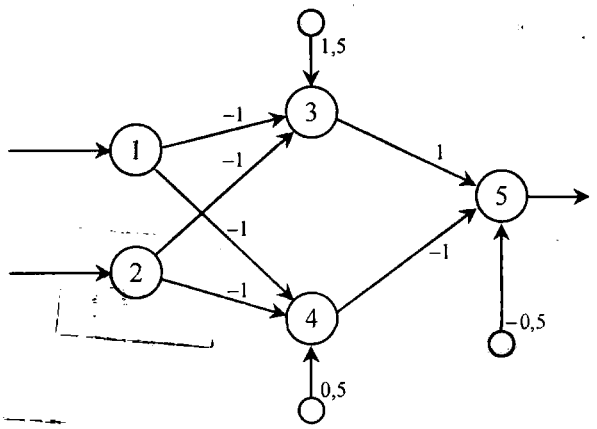

Рис. 2. Нейронна мережа, що моделює відношення XOR

Як уже зазначалось вище, властивості штучних нейронних мереж стосовно апроксимації досліджувались багатьма авторами. Основою цих досліджень найчастіше ставали відома теорема Колмогорова–Арнольда [14, 15] або теорема Стоуна [16]. Наслідком з останньої є, зокрема, такий результат: для довільної неперервної нелінійної функції активації φ нейрона, що є ∑-елементом, за допомогою штучних нейронних мереж можна як завгодно точно наблизити будь-яку неперервну функцію багатьох змінних на довільній замкненій обмеженій множині [2].

У роботах [7, 8] йдеться про рекурсивні нейронні мережі, обчислювальні можливості яких не менші за обчислювальні можливості машини Тьюрінга. З результатів цих робіт та з тези Чорча випливає, що для довільної алгоритмічно обчислюваної функції існує нейронна мережа, яка її обчислює. Дамо конструктивний спосіб побудови нейронної мережі, що обчислює задану алгоритмічно обчислювану функцію.

Згідно з тезою Чорча клас алгоритмічно обчислюваних функцій збігається з класом частково рекурсивних функцій. Для побудови нейронної мережі, що обчислює задану частково рекурсивну функцію, скористаємось теоремою про представлення примітивної рекурсії [17,18]. Згідно з наслідком з цієї теореми, клас частково рекурсивних функцій збігається з класом функцій, одержаних з базових функцій $O(x) = 0$, $S(x) = x + 1$, $I_m^n(x_1, x_2, ..., x_n) = x_m$, $1 \le m \le n$ та функцій $+(x_1, x_2)$, $*(x_1, x_2)$, $\div(x_1, x_2)$ за допомогою скінченної кількості застосувань операцій суперпозиції S^{n+1} та мінімізації μ. Зауважимо, що ми розглядаємо частково рекурсивні функції, які діють на множині натуральних чисел. Про те, як легко поширити поняття обчислюваності на інші області, можна прочитати, наприклад, у [19].

Базова функція $O(x)$ реалізується ∑-елементом з тотожною функцією активації $f(x) = x$ та єдиним з'єднанням вагою 0.

Базова функція $S(x)$ реалізується ∑-елементом з тотожною функцією активації $f(x) = x$, єдиним з'єднанням вагою 1 та величиною порога 1.

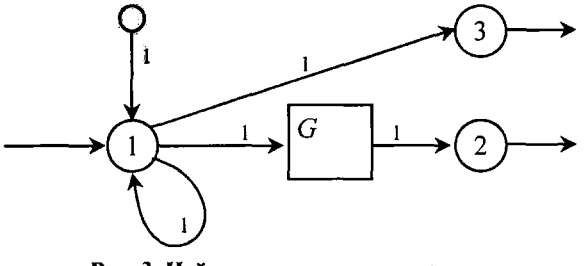

Рис. 3. Нейронна мережа, що моделює операцію мінімізації

Базова функція $I_m^n(x_1, x_2, ..., x_n) = x_m$, $1 \le m \le n$ також реалізується ∑-елементом з тотожною функцією активації $f(x) = x$; цей елемент має n з'єднань, причому m-те з'єднання має вагу 1, а решта – 0.

Функція $+(x_1, x_2)$, очевидно, реалізується ∑-елементом з тотожною функцією активації $f(x) = x$ та вагою 1 кожного з'єднання.

Функція $*(x_1, x_2)$ реалізується ∑-∏-елементом з тотожною функцією активації $f(x) = x$ та вагою 1 єдиного з'єднання.

Функція $\div(x_1, x_2)$ реалізується ∑-елементом з вагою 1 на з'єднанні, що відповідає першому параметру, та вагою –1 на з'єднанні, що відповідає другому параметру. Функцією активації при цьому є функція $f(x) = \begin{cases} x, & якщо \quad x \ge 0; \\ 0, & якщо \quad x < 0. \end{cases}$

Операція суперпозиції в очевидний спосіб реалізується поданням виходу одного нейрона на вхід іншого.

Покажемо, як можна реалізувати операцію мінімізації. Нагадаємо, що

$$\mu_y(g(x_1, x_2, ..., x_n, y) = 0) =$$

$$= \begin{cases} найменше \ з \ таких \ значень \ y, \ що \\ g(x_1, x_2, ..., x_n, y) = 0 \ \ та \ \ \forall z \le y \ визначено \\ g(x_1, x_2, ..., x_n, z) \ не \ визначене, \ інакше, \end{cases}$$

де g – частково рекурсивна функція.

Припустимо, що операторний терм функції g не містить операцій мінімізації (якщо це не так, то будемо реалізовувати ту з операцій мінімізації операторного терма функції g, яка не тягне за собою інших операцій мінімізації). Тоді, як показано вище, можна побудувати нейронну мережу G, що обчислює функцію $g(x_1, x_2, ..., x_n, y)$. Нейронну мережу, що реалізує операцію мінімізації, будемо конструювати з використанням мережі G; назвемо мережу G макронейроном і позначатимемо її квадратом в графі міжнейронних з'єднань.

Граф міжнейронних з'єднань нейронної мережі, що задає обчислення $\mu_y(g(x_1, x_2, ..., x_n, y) = 0)$, подано на рис. 3. Усі нейронні елементи цієї мережі є ∑-елементами. Функцією активації для нейронів 1 та 3 є тотожна функція $f(x) = x$, а для нейрона 2 – функція $f(x) = \begin{cases} 1, & якщо \quad x = 0; \\ 0, & інакше. \end{cases}$

Порядок спрацювання нейронів визначається так. На першому такті спрацьовує нейрон 1, далі за k тактів спрацьовує макронейрон G, потім одночасно спрацьовують нейрони 2 та 3. Якщо нейрон 2 має активність 1, то результат знімається з нейрона 3, в іншому випадку знову спрацьо-

вує нейрон 1, і обчислення повторюються. Як бачимо, побудована мережа є рекурсивною та видає результат за скінченну кількість кроків, якщо такий результат існує.

З усього вищесказаного випливає, що, маючи спосіб реалізації функцій O, S, I_m^n, $+$, $*$, $\div$, операцій S^{n+1} та μ і маючи операторний терм заданої частково рекурсивної функції, легко побудувати нейронну мережу, що обчислює вказану функцію. Зауважимо, що ця нейронна мережа складається з Σ- та Σ-Π-елементів та як функції активації використовує функції $f(x) = x$,

$$f(x)=\begin{cases} x, & \textit{якщо} \quad x \geq 0; \\ 0, & \textit{якщо} \quad x < 0 \end{cases} \text{ та } f(x)=\begin{cases} 1, & \textit{якщо} \quad x = 0; \\ 0, & \textit{інакше}. \end{cases}$$

Теорема 2. Для довільного локального алгоритму існує нейронна мережа, що його реалізує.

Доведення. Нехай задано локальний алгоритм з множиною елементів D та з визначеними на елементах $d \in D$ функціями $f_d(\underline{Інф}(\tau(d)))$ і зображуючими предикатами $p_d(\underline{Інф}(\tau(d)))$. Нехай F_d – нейронні мережі, що обчислюють функції f_d, а Q_d – нейронні мережі, що обчислюють зображуючі предикати p_d. Існування таких мереж випливає з викладеного вище матеріалу та з того факту, що предикати можна розглядати як функції з областю значень потужності 2. Вказані мережі називатимемо макронейронами та позначатимемо квадратами в графах міжнейронних з'єднань. Ставлячи у відповідність елементам $d \in D$ макронейрони F_d та з'єднуючи їх згідно з відношенням $\tau(d)$, отримаємо підграф графа міжнейронних з'єднань.

Локальний оператор O_k реалізується шляхом одночасного спрацювання таких макронейронів F_d, для яких елементи d належать області визначення локального оператора O_k. Під «одночасним спрацюванням макронейронів» тут розуміємо таке їх спрацювання, при якому всі макронейрони починають свою роботу одночасно, а кінцем обчислень вважається виконання останнього такту тим з макронейронів, робота якого вимагає найбільше тактів.

Для задання локальної умови розширимо побудовану нейронну мережу макронейронами Q_d так, що вихід з кожного макронейрона F_d зв'язується із входом відповідного макронейрона Q_d. Для кожного локального предиката $P_k(\mu)$ розширимо мережу макронейронами AND_k (які, очевидно, обчислюють булеву функцію AND), подаючи на вхід цих макронейронів виходи з тих Q_d, для яких елементи d належать області визначення локального предиката $P_k(\mu)$. Локальний предикат P_k реалізується шляхом одночасного спрацювання таких макронейронів Q_d, для яких елементи d належать області визначення локального предиката P_k та спрацювання відповідного макронейрона AND_k.

Прийнявши домовленості щодо тактування побудованої нейронної мережі згідно з послідовністю виконання локальних операторів та предикатів, визначеною локальним алгоритмом, отримаємо нейронну мережу, що реалізує заданий локальний алгоритм. Про способи подачі вхідних сигналів у нейронну мережу (що відповідає процесу задання початкових значень $\underline{Інф}(d)$), можна прочитати, наприклад, в [1].

Теорему доведено.

Зауважимо, що у випадку, коли функції f_d є неперервними і визначеними на компакті, та за відсутності локальних предикатів у поданні локального алгоритму, можна побудувати гомогенну нейронну мережу. Цей факт випливає з доведення теореми та з того, що для довільної неперервної нелінійної функції активації φ нейрона, що є Σ-елементом, за допомогою нейронних мереж можна як завгодно точно наблизити будь-яку неперервну функцію багатьох змінних на довільній замкненій обмеженій множині [2].

Приклад 2. Як приклад розглянемо локальний алгоритм перевірки зв'язності орграфа, що був запропонований А. В. Анісімовим у [10].

Нехай задано орієнтований граф $G = (V, E)$, де V – множина вершин, E – множина дуг, $|V| = n$, $|E| = m$. Позначимо через $R(v)$ множину всіх вершин, зв'язаних з вершиною v дугами. Тоді елементами множини D обчислювального середовища локального алгоритму є вершини графа; околом елемента $d \in D$, що відповідає вершині v графа, є $\tau(d) = R(v) \cup \{d\}$.

Серед усіх вершин графа G виділимо довільну вершину v_0 та покладемо

$$\underline{Інф}(v)=\begin{cases} 1, & \textit{якщо} \quad v = v_0; \\ +\infty, & \textit{інакше}. \end{cases}$$

Локальний оператор $O(\mu)$ визначається як одночасне обчислення всіх функцій f_d на елементах $d \in D$, де $f_d(\underline{Інф}(\tau(d))) = \min\limits_{v \in \tau(d)} (\underline{Інф}(v))$.

Локальний предикат $P(\mu) = \forall d\, p(\underline{Інф}(\tau(d)))$ визначається як одночасне обчислення всіх зображуючих предикатів p, де

$$p(\underline{Інф}(\tau(d)))=\begin{cases} 1, & \textit{якщо} \quad \underline{Інф}(d) = 1; \\ 0, & \textit{інакше}. \end{cases}$$

Локальний алгоритм розпізнавання зв'язності орграфа запишеться таким чином:

```
1 begin
2   for i = 1 to n do
3     O(μ);
4   if P(μ) then
5     return true
6   else
7     return false;
8 end.
```

Побудуємо спочатку нейронну мережу, що обчислює функцію $\min(x_1, x_2)$. Нагадаємо, що $\min(x_1, x_2) = x_1 \div (x_1 \div x_2)$. Граф міжнейронних з'єднань нейронної мережі, що обчислює функцію $\min(x_1, x_2)$, подано на рис. 4. Усі нейронні елементи цієї мережі є $\sum$-елементами. Нейрони 1 та 2 є вхідними (перший та другий параметри функції відповідно). Функцією активації нейронів 3 та 4 є функція $f(x) = \begin{cases} x, & якщо \quad x \geq 0; \\ 0, & якщо \quad x < 0. \end{cases}$ На першому такті функціонування мережі спрацьовує нейрон 3, на другому – нейрон 4; результат знімається з нейрона 4. Нейронна мережа, що обчислює функцію мінімуму від n параметрів, $n > 2$, легко будується з урахуванням такого співвідношення:

$$\min(x_1, x_2, \ldots, x_n) =$$
$$= \min\left(x_1, \min\left(x_2, \min\left(\ldots, \min(x_n)\right)\right)\right).$$

Через F_d позначимо нейронну мережу, що обчислює функцію $\min(x_1, x_2, \ldots, x_n)$, де $x_i \in \tau(d)$, $i = 1, 2, \ldots, |\tau(d)|$. У графі, що досліджується на зв'язність, замінимо вершини d на макронейрони F_d, зберігаючи усі зв'язки між вершинами та додавши ще зворотні зв'язки макронейронів на себе. Зображуючий предикат у даному випадку, очевидно, може бути обчислений нейроном, що є $\sum$-елементом та має функцію активації $f(x) = \begin{cases} 1, & якщо \quad x = 1; \\ 0, & інакше, \end{cases}$ позначимо ці нейрони Q_d. З'єднаємо макронейрони F_d з нейронами Q_d, а останні – з макронейроном обчислення функції AND.

Приклад отриманого графа міжнейронних з'єднань для орграфа з п'ятьма вершинами подано на рис. 5.

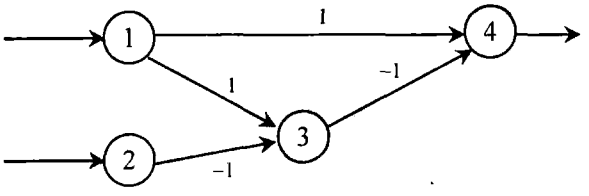

Рис. 4. Нейронна мережа, що моделює обчислення функції $\min(x_1, x_2)$

Тактування отриманої нейронної мережі задається так: n раз одночасно спрацьовують усі макронейрони F_d, потім один раз спрацьовують нейрони Q_d, і, нарешті, спрацьовує макронейрон обчислення функції AND. Результат знімається з макронейрона AND.

Останніми роками досить широко вживається сполучення «нейронні мережі – генетичні алгоритми». І не випадково. Адже, як зазначено в [20], генетичне програмування, що є гілкою генетичних алгоритмів, та штучні нейронні мережі можуть розглядатись як альтернативні техніки розв'язання одних і тих самих задач, зокрема проблем класифікації та апроксимації. Досить інтригуючим, як зазначено в [21], є й саме поєднання генетичних алгоритмів із штучними нейронними мережами, оскільки обидва напрямки належать до еволюційного моделювання, яке будується на концепції еволюції, що відбувається в живій природі. Генетичні алгоритми показали високу ефективність при розв'язуванні задач глобальної оптимізації [22, 23, 24]. Широке застосування вони знайшли в задачах пошуку оптимальної структури штучних нейронних мереж, призначених для вирішення конкретних проблем [21, 25].

Нагадаємо, що для задання генетичного алгоритму необхідно:

- представити оптимізаційні параметри об'єкта у вигляді хромосом;
- згенерувати початкову популяцію;
- вказати схему репродукції та правило обчислення ступеня пристосованості індивіда;
- задати генетичні оператори;
- задати критерій зупинки еволюційного процесу.

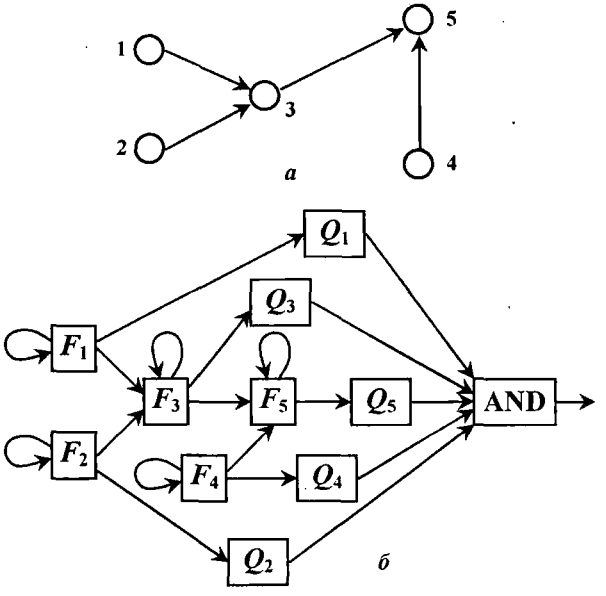

Рис. 5. Орграф (*а*) та нейронна мережа (*б*) розв'язку задачі його зв'язності

Серед генетичних операторів найважливішими є мутація та кросовер. Мутація інтерпретується як випадкова заміна існуючого стану гена в хромосомі на протилежний (напр., 10100111⇒10000111). Кросовер генерує нову хромосому, об'єднуючи генетичний матеріал двох батьківських хромосом. Найпростішим варіантом кросовера є одноточковий, при якому дві хромосоми перерізаються у випадково обраній точці і результуюча хромосома одержується з початку однієї та кінця іншої батьківської хромосоми (напр., **1**1011101, **10100111** ⇒ **10100111**). Приклади та аналіз різних схем репродукції та генетичних операторів можна знайти в [21, 26, 27].

Легко бачити, що генетичні алгоритми можуть бути реалізовані в паралельних архітектурах. Існує два основних підходи до паралельної реалізації генетичних алгоритмів: на основі підпопуляцій та на основі індивідів [23, 28]. У підході на основі індивідів (diffusion model) кожному індивіду приписується певне місце у вузлі решітки невеликої розмірності. Популяція розглядається як система активних індивідів, що взаємодіють лише зі своїми сусідами. Зауважимо, що найчастіше використовується двовимірна решітка з чотирма (окіл фон Неймана) або вісьмома (окіл Мура) сусідами. Оператори репродукції та генетичні оператори виконуються локально (тобто з урахуванням інформації лише від сусідів). У роботі [28] показано спосіб реалізації таких алгоритмів на архітектурах типу SIMD, а в роботі [23] – за допомогою клітинних автоматів.

Теорема 3. *Генетичні алгоритми на основі індивідів є локальними алгоритмами.*

Доведення. Розглядатимемо генетичний алгоритм на основі індивідів з функцією обчислення пристосованості індивідів φ, функцією ψ визначення вибору індивіда для збереження, генетичними операторами мутації та одноточкового кросовера, що застосовуються по черзі, функцією g вибору сусіда для виконання операції кросовера, що повертає його номер (всі індивіди вважаються перенумерованими), функцією h_1 вибору точки мутації та функцією h_2 вибору точки кросовера.

Задамо локальний алгоритм у такий спосіб.

Нехай D – множина індивідів, що утворюють популяцію; тоді елемент $d \in D$ є індивідом з певним генотипом. Визначимо окіл $\tau(d)$ як самого індивіда d та його сусідів; нехай $|\tau(d)| = m$. Задамо $\underline{Інф}(d)$ як вектор значень $(x_0, x_1, x_2, \ldots, x_m, x_{m+1}, x_{m+2})$, де x_0 – номер індивіда, $x_1, x_2, \ldots, x_m$ – послідовність генів хромосоми індивіда d, x_{m+1} – значення обчисленої пристосованості індивіда d, а $x_{m+2} = \begin{cases} 1, & \textit{якщо слід застосувати операцію мутації}; \\ 2, & \textit{якщо слід застосувати операцію кросовера.} \end{cases}$

Тоді локальний оператор $O(\mu)$ полягає в обчисленні функцій $f_d(\underline{Інф}(\tau(d)))$, що задаються в такий спосіб:

$$f_d(\underline{Інф}(\tau(d))) =$$

$$= \begin{cases} (x_0, x_1, \ldots, \overline{x}_{h_1(\ldots)}, \ldots, x_m, \varphi(x_1, x_2, \ldots, \overline{x}_{h_1(\ldots)}, \ldots, x_m), 2), \\ \quad \textit{якщо } x_{m+2} = 1 \textit{ та } \psi(\varphi(x_1, \ldots, \overline{x}_{h_1(\ldots)}, \ldots, x_m), x_{m+1}) > 0; \\ (x_0, x_1, \ldots, x_{h_1(\ldots)}, \ldots, x_m, x_{m+1}, 2), \textit{ якщо } x_{m+2} = 1 \\ \quad \textit{та } \psi(\varphi(x_1, \ldots, \overline{x}_{h_1(\ldots)}, \ldots, x_m), x_{m+1}) \le 0; \\ (x_0, x_1, \ldots, x_{h_2(\ldots)}^{g(\ldots)}, \ldots, x_m^{g(\ldots)}, \varphi(x_1, x_2, \ldots, x_{h_2(\ldots)}^{g(\ldots)}, \ldots, x_m^{g(\ldots)}), 1), \\ \quad \textit{якщо } x_{m+2} = 2 \textit{ та } \psi(\varphi(x_1, \ldots, x_{h_2(\ldots)}^{g(\ldots)}, \ldots, x_m^{g(\ldots)}), x_{m+1}) > 0; \\ (x_0, x_1, \ldots, x_{h_2(\ldots)}, \ldots, x_m, x_{m+1}, 1), \textit{ якщо } x_{m+2} = 2 \\ \quad \textit{та } \psi(\varphi(x_1, \ldots, x_{h_2(\ldots)}^{g(\ldots)}, \ldots, x_m^{g(\ldots)}), x_{m+1}) \le 0, \end{cases}$$

де $\overline{x}$ – значення, протилежне x, d_u – сусід індивіда d з номером u, x_k^u – k-та координата вектора $\underline{Інф}(d_u)$.

Зауважимо, що випадок інших генетичних операторів та інших схем їх застосування легко отримується відповідними змінами в наведених вище міркуваннях.

Теорему доведено.

1. *Горбань А. Н., Россиев Д. А.* Нейронные сети на персональных компьютерах.– Новосибирск: Наука. Сибирская издательская фирма РАН, 1996.– 276 с.
2. *Горбань А. Н.* Нейроинформатика: кто мы, куда идем, как путь наш измерить? // Информационные технологии.– 2000.– № 4.– С. 2–6.
3. *Галушкин А. И.* Некоторые исторические аспекты развития элементной базы вычислительных систем с массовым параллелизмом (80–90-е годы) // Информационные технологии.– 2000.– № 8.– С. 2–10.
4. *Корнеев В. В.* Параллельные вычислительные системы.– М.: Нолидж, 1999.– 320 с., ил.
5. *Cybenko G.* Approximation by Superposition of a Sigmoidal Function // Mathematics of Control, Signals and Systems. – 1989.– Vol. 2.– P. 303–314.
6. *Sandberg I. W.* Approximation for Nonlinear Functionals // IEEE Transactions on Circuits and Systems.– 1: Fundamental Theory and Applications.– Vol. 39, № 1.– January 1992.– P. 65–67.
7. *Siegelmann H. T., Sontag E. D.* On the Computational Power of Neural Nets // Journal of Computer and System Sciences.– 1995. – Vol. 50 (1). – P. 132–150.
8. *Siegelmann H. T.* Computation Beyond the Turing Limit // Science. – Vol. 268.– 29 April 1995.– P. 545–548.
9. *Журавлев Ю. И.* Локальные алгоритмы вычисления информации. 1 // Кибернетика.– 1965.– № 1.– С. 12–19.
10. *Анисимов А. В.* О локальных вычислениях // Модели и системы обработки информации.– К.: Вища школа, 1987.– Вып. 6.– С. 3–7.
11. *Чернов А. Ю.* Об одном классе локальных алгоритмов // Докл. АН СССР.– 1989.– Т. 308.– № 4.– С. 795–798.
12. *Евстигнеев В. А.* Граф-модели систем и основные принципы их исследования // автореф. дис. ... докт. физ.-мат. наук.– Новосибирск, 1991.
13. *Дюк В., Самойленко А.* Data mining: учебный курс.– СПб: Питер, 2001.– 368 с.: ил.
14. *Колмогоров А. Н.* О представлении непрерывных функций нескольких переменных в виде суперпозиции непрерыв-

ных функций одного переменного и сложения // Докл. АН СССР.– 1957.– Т. 114.– № 5.– С. 953–956.
15. *Арнольд В. И.* О представлении функций нескольких переменных в виде суперпозиции функций меньшего числа переменных // Математическое просвещение.– 1958.– Вып. 3.– С. 41–61.
16. *Stone M. N.* The Generalized Weierstrass Approximation Theorem // Math. Mad.– 1948.– V. 21.– P. 167–183, 237–254.
17. *Мальцев А. И.* Алгоритмы и рекурсивные функции.– М.: Наука. Гл. ред. физ.-мат. лит., 1986.– 368 с.
18. *Лісовик Л. П., Шкільняк С. С.* Основи теорії алгоритмів: Навчальний посібник.– К.: НМЦ КУ «Інтелектуальні системи», 1993.– 94 с.
19. *Катленд Н.* Вычислимость. Введение в теорию рекурсивных функций.– М.: Мир, 1983.– 256 с.
20. *Brameier M., Banzhaf W.* A Comparison of Linear Genetic Programming and Neural Networks in Medical Data Mining // IEEE Transactions on Evolutionary Computation.– 2001.– Vol. 5.– № 1.– P. 17–26.
21. *Вороновский Г. А., Махотило К. В., Петрашев С. Н., Сергеев С. А.* Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности.– Харьков: Основа, 1997.– 112 с.
22. *Coello C. A., Rudnick M., Christiansen A. D.* Using Genetic Algorithms for Optimal Design of Trusses // Proc. 6th International Conference on Tools with Artificial Intelligence, Nov. 6–9, 1994, New Orleans, Louisiana.– P. 88–94.
23. *Folino G., Pizzuti C., Spezzano G.* Parallel Hybrid Method for SAT That Couples Genetic Algorithms and Local Search // IEEE Transactions on Evolutionary Computation.– 2001.– Vol. 5.– № 4.– P. 323–334.
24. *Комарцова Л. Г.* Поисковое проектирование распределенных вычислительных систем с использованием генетического алгоритма // Информационные технологии.– 2000.– № 5.– С. 25–30.
25. *Moon Sang-Woo, Kong Seong-Gon.* Block-Based Neural Networks // IEEE Transactions on Neural Networks.– 2001.– Vol. 12.– № 2.– P. 307–317.
26. *Кисляков А. В.* Генетические алгоритмы: математический анализ некоторых схем репродукции // Информационные технологии.– 2000.– № 12.– С. 9–14.
27. *Кисляков А. В.* Генетические алгоритмы: операторы скрещивания и мутации // Информационные технологии.– 2001.– № 1.– С. 29–34.
28. *Куприянов М. С., Матвиенко Н. И.* Генетические алгоритмы и их реализации в системах реального времени // Информационные технологии.– 2001.– № 1.– С. 17–21.

N. M. Gulayeva

LOCAL ALGORITHMS: CONNECTION WITH ARTIFICIAL NEURAL NETWORKS AND GENETIC ALGORITHMS

It is shown, that for every local algorithm (by Anisimov) there exists an artificial neural network implementing this algorithm, and vice versa. Examples of construction of such algorithms and neural networks are given. It is also shown that genetic algorithms implemented via diffusion model are local algorithms (by Anisimov).