ZESZYTY NAUKOWE POLITECHNIKI L SKIEJ 2007
Seria: BUDOWNICTWO z. 112 Nr kol. 1763

Jerzy SKRZYPCZYK*
Politechnika Śląska

SZTUCZNA INTELIGENCJA W ZAGADNIENIACH INŻYNIERSKICH

Streszczenie. Sztuczna inteligencja ma dwa podstawowe znaczenia: jest to hipotetyczna inteligencja realizowana w procesie inżynieryjnym, a nie naturalnym oraz nazwa technologii i dziedzina badań naukowych informatyki na styku neurologii, psychologii i ostatnio kognitywistyki, teorii systemów, a nawet współczesnej filozofii. W pracy przedstawiono specjalizowane systemy obliczeniowe jedno- i wieloczynnikowe. Bardziej szczegółowo natomiast omówiono algorytmy genetyczne oraz sieci neuronowe.

ARTIFICIAL INTELLIGENCE IN ENGINEERING PROBLEMS

Summary. Generally speaking artificial intelligence (AI) has two basic meanings: it's the hypothetical intelligence realized in engineering process, not in the natural one and the second meaning is built around technology and the combination area of information sciences, neurology, psychology, cognitivity, system theory and the modern philosophy. Specialized one-factor and multi-factor numerical systems are discussed. Genetic algorithms and neural sets are presented with more details.

1. Wstęp

Sztuczna inteligencja (ang. *Artificial Intelligence – AI*) ma dwa podstawowe znaczenia. Jest to:

- hipotetyczna inteligencja realizowana w procesie inżynieryjnym, a nie naturalnym,
- nazwa technologii i dziedzina badań naukowych informatyki na styku neurologii, psychologii, a nawet współczesnej filozofii.

Głównym zadaniem badań nad sztuczną inteligencją (w drugim znaczeniu) jest konstruowanie urządzeń i programów komputerowych zdolnych do realizacji wybranych funkcji umysłu i ludzkich zmysłów, które nie poddają się prostej, numerycznej algorytmizacji. Problemy takie bywają nazywane *NP-trudnymi* i zalicza się do nich między innymi:

- podejmowanie decyzji w warunkach braku wszystkich danych,

* Wykład wprowadzający, wygłoszony na VIII Konferencji Doktorantów Wydziału Budownictwa, Szczyrk, 8-9. 11. 2007 r.

- analizę i syntezę języków naturalnych,
- rozumowanie logiczne/racjonalne,
- dowodzenie twierdzeń,
- teorię gier logicznych, jak np. warcaby czy szachy,
- zarządzanie wiedzą, preferencjami i informacją w robotyce,
- systemy diagnostyczne i ekspertowe.

AI jako dział badań naukowych zaczęła się rozwijać w latach 50. XX w., kiedy to powstało pierwsze laboratorium AI na Uniwersytecie Carnegie Mellon, założone przez Allena Newella oraz Herberta Simona, i kilka lat później laboratorium w MIT (Massachusetts Institute of Technolog), założone przez Johna McCarthy'ego.

Termin *sztuczna inteligencja* po raz pierwszy został zaproponowany przez Johna McCarthy'ego, który w 1955 r. zdefiniował go w następujący sposób: „konstruowanie maszyn, o których działaniu dałoby się powiedzieć, że jest podobne do ludzkich przejawów inteligencji" [2, 3, 5, 6, 8, 10, 13, 20].

Istnieją dwa podstawowe podejścia do pracy nad AI:

- *podejście symboliczne,* polegające na tworzeniu modeli matematyczno-logicznych analizowanych problemów i implementowaniu ich w formie programów komputerowych, mających realizować konkretne funkcje powszechnie uważane za składowe inteligencji. W tej grupie są np. algorytmy genetyczne i metody logiki rozmytej;
- *podejscie subsymboliczne,* polegające na tworzeniu struktur i programów „samouczących się", bazujących na modelach sieci neuronowych, klasyfikatorów i sieciach asocjacyjnych, oraz opracowywaniu procedur „uczenia" takich programów.

Prace w dziedzinie AI przyniosły wiele konkretnych rezultatów, które znalazły już praktyczne i powszechne zastosowania, począwszy od domowej pralki, poprzez kompletne sterowanie helikopterami, aż do programów kosmicznych [2].

2. Specjalizowane systemy obliczeniowe

Mało kto zdaje sobie dzisiaj sprawę z tego, jak niezwykle skomplikowane stały się współczesne systemy obliczeniowe matematyki stosowanej. Obok klasycznych, znanych od dawna liczb naturalnych, całkowitych i rzeczywistych pojawiły się liczby wieloskładnikowe, składające się z kilku elementów, takich jak liczby: zespolone (1837), przedziałowe (1922), perturbacyjne I rzędu (2003) i II rzędu (2006).

Analiza systemów zawierających elementy niedeterministyczne (niepewne) spowodowała wprowadzenie liczb losowych (przypadkowych) (1867, 1933) i liczb rozmytych (1965).

Do najprostszych należą liczby jednoskładnikowe, np. naturalne, całkowite, wymierne i rzeczywiste [7, 12].

Zbiór *liczb naturalnych* jest zbiorem, który poznajemy w szkole podstawowej, a mianowicie:

$$N: = \{1,2,3,.....\}.$$

Jest on domknięty ze względu na dodawanie i mnożenie, tzn. suma dwóch liczb naturalnych jest znowu liczbą naturalną, podobnie iloczyn dwóch liczb naturalnych jest również taką liczbą. Jednak zbiór ten nie jest zamknięty ze względu na odejmowanie i dzielenie, ponieważ istnieją liczby naturalne, np. *3* i *7*, których różnica:

$$3 - 7 = -4$$

nie jest liczbą naturalną. To samo można powiedzieć o ilorazie liczb naturalnych.

Zbiór *liczb całkowitych* jest zbiorem:

$$Z: = \{....-3, -2, -1, 0, 1,2,3,....\},$$

który zawiera liczby naturalne jako podzbiór. Ten zbiór ma taką przewagę, że jest domknięty ze względu na dodawanie, mnożenie i odejmowanie. Zauważmy, że różnica dwóch liczb całkowitych jest ponownie liczbą całkowitą. Jednak liczby całkowite nie stanowią zbioru domkniętego ze względu na dzielenie.

Dla otrzymania czwartej własności potrzebny będzie zbiór *liczb wymiernych*, poza przypadkiem dzielenia przez zero. W taki sposób liczby wymierne stanowi zbiór:

$$\Theta: = \{p/q : p,q - \text{liczby całkowite, przy } q \neq 0\}.$$

Zawiera on jako podzbiory zarówno liczby naturalne, jak i całkowite i jest domknięty ze względu na wszystkie cztery operacje arytmetyczne. Zbiór ten wypełnia wszystkie własności algebraiczne, które przez matematyków przypisane są do tzw. pola.

Zbiór *R liczb rzeczywistych*, ze znanymi nam już zwykłymi operacjami dodawania i mnożenia, stanowi inny przykład pola algebraicznego. Znanych jest wiele definicji *liczb rzeczywistych*; jedna z najczęściej używanych nosi nazwę definicji Cantora (Georg Cantor 1845-1918). Punktem wyjścia jest pojęcie ciągu podstawowego liczb wymiernych. Przez ciąg podstawowy rozumiemy ciąg Cauchy'ego liczb wymiernych. Dwa ciągi podstawowe $\{a_n\}$ i $\{b_n\}$ liczb wymiernych nazwiemy równoważnymi, jeżeli ciąg $\{a_n - b_n\}$ jest zbieżny do zera. Ciągi podstawowe można identyfikować. W związku z tym w teorii Cantora *liczba rzeczywista* jest definiowana jako klasa równoważności ciągów podstawowych liczb wymiernych [12].

Rozwój teorii matematycznych zaowocował powstaniem liczb wieloskładnikowych; pierwsze, jak się wydaje, były *liczby zespolone*. Po raz pierwszy liczby zespolone pojawiły się w znanej pracy G. Cardano (1545) „Wielka sztuka, czyli o regułach algebraicznych", w której uznano je za nieużyteczne i niewygodne w zastosowaniu. Dla większości matematyków XVII wieku sensy algebraiczny i geometryczny wielkości urojonych pozostawały jednak niejasne, a nawet zagadkowe i mistyczne. Wiadomo na przykład, że I. Newton nie zaliczał wielkości urojonych do liczb, a G. Leibnitz wypowiedział nawet słowa: „Liczby urojone – to piękne i cudowne schronienie boskiego ducha, prawie połączenie bytu z niebytem".

Pojęcie *liczba zespolona* wprowadził C. Gauss w 1831 r. Czysto arytmetyczna teoria liczb zespolonych jako par liczb rzeczywistych była stworzona przez W. Hamiltona w 1837 r. Skonstruował on ścisłą algebrę liczb zespolonych, opartą na pojęciu liczby zespolonej jako pary liczb rzeczywistych. Jemu też zawdzięczamy ważne uogólnienie liczb zespolonych – *kwaterniony*, których algebra jest niestety nieprzemienna [12].

Uogólnieniem teorii kwaternionów zajął się William Kingdon Clifford (1845-1879). Wprowadził on tzw. *bikwaterniony*, czyli kwaterniony o współczynnikach będących liczbami zespolonymi $a+b\varepsilon$, gdzie ε^2 może przyjmować wartości +1, -1 lub 0 i którymi można się posługiwać przy badaniu ruchu w przestrzeniach nieeuklidesowych.

W 1922 r. Jahn wprowadził *liczby przedziałowe*, por. [9]. Przypomnijmy, że liczba przedziałowa $\bar{a} = [a^-, a^+]$ oznacza zbiór wszystkich liczb rzeczywistych a, takich że $a^- \leq a \leq a^+$. Może być ona traktowana jako uporządkowana para liczb rzeczywistych (a^+, a^-). Bywa też zapisywana w innej postaci $\bar{a} = [\breve{a} - \Delta a, \breve{a} + \Delta a] = (\breve{a}, \Delta a)_r$, gdzie $\breve{a}$ to środek przedziału, natomiast Δa oznacza jego promień, por. [9].

Dla liczb interwałowych $\bar{a}, \bar{b} \in I(R)$ definiujemy następujące operacje arytmetyczne: jeśli "*" jest jednym z symboli "+,-,·,/", to wtedy:

$$\bar{a} * \bar{b} := \{a * b : a \in \bar{a}, b \in \bar{b}\} \quad \text{z wyjątkiem } \bar{a} / \bar{b}, \text{ jeśli } 0 \in \bar{b}.$$

Działania te wykonujemy według następujących wzorów:

$$\bar{c} := \bar{a} + \bar{b}\ ,\ gdzie\ c^- = a^- + b^-\ i\ c^+ = a^+ + b^+$$

$$\bar{c} := \bar{a} \cdot \bar{b}\ ,\ gdzie\ c^- = min\left(a^- \cdot b^-, a^- \cdot b^+, a^+ \cdot b^-, a^+ \cdot b^+\right)\ c^+ = \max\left(a^- \cdot b^-, a^- \cdot b^+, a^+ \cdot b^-, a^+ \cdot b^+\right)$$

Liczby perturbacyjne stanowią najnowszą i najmniej znaną, jak się wydaje, kategorię liczb wieloskładnikowych. Po raz pierwszy pojawiły się w 2003 r. w pracach Skrzypczyka [14-17]. Liczbę zwaną dalej liczbą perturbacyjną definiujemy jako parę uporządkowaną liczb rzeczywistych $(x,y) \in R^2$. Zbiór liczb perturbacyjnych będziemy oznaczać jako R_ε.

Niech $z, z_1, z_2 \in R_\varepsilon$ oznaczają dowolne liczby perturbacyjne oraz $z=(x,y), z_1:=(x_1,y_1)$, $z_2:=(x_2,y_2)$, $x_i, y_i \in R$, $i=1,2$. Powiemy wówczas, że dwie liczby perturbacyjne są równe: $z_1 = z_2$ wtedy i tylko wtedy, gdy $x_1 = x_2$ oraz $y_1 = y_2$.

W zbiorze R_ε w następujący sposób wprowadzimy działania dodawania $(+_\varepsilon)$ i mnożenia $(\bullet_\varepsilon)$:

$$z_1 +_\varepsilon z_2 := (x_1 + x_2, y_1 + y_2)$$

$$z_1 \bullet_\varepsilon z_2 := (x_1 x_2, x_1 y_2 + x_2 y_1)$$

Zbiór R_ε z działaniami dodawania $(+_\varepsilon)$ i mnożenia $(\bullet_\varepsilon)$ oraz z wyróżnionymi elementami zerowym $0_\varepsilon := (0,0)$ oraz jedynkowym $1_\varepsilon := (1,0)$ jest ciałem.

Odnotować należy, że istnieje wiele modyfikacji wielkości perturbacyjnych, takich jak: liczby perturbacyjne II rzędu, liczby perturbacyjne wieloskalowe, przedziałowe liczby perturbacyjne, wieloskalowe przedziałowe liczby perturbacyjne itp., por. [18-19].

Liczby rozmyte pojawiły się w 1965 r. w pracach Zadeha, por. [5, 13].

Zbiorem rozmytym $\tilde{A}$, określonym na przestrzeni X, co zapisujemy $\tilde{A} \subseteq X$, jest zbiór par uporządkowanych: $\tilde{A} = \left\{(x, \mu(x;\tilde{A})) | x \in X\right\}$, gdzie $\mu(x;\tilde{A}) : X \to [0,1]$ oznacza tzw. funkcję przynależności zbioru rozmytego $\tilde{A}$. Funkcja ta każdemu elementowi $x \in X$ przypisuje jego stopień przynależności do zbioru rozmytego $\tilde{A}$, przy czym można wyróżnić trzy przypadki:

1) $\mu(x;\tilde{A}) = 1$ oznacza pełną przynależność elementu x do zbioru rozmytego $\tilde{A}$, tzn. $x \in \tilde{A}$,

2) $\mu(x;\tilde{A}) = 0$ oznacza brak przynależności elementu x do zbioru rozmytego $\tilde{A}$, tzn. $x \notin \tilde{A}$,

3) $0 < \mu(x;\tilde{A}) < 1$ oznacza częściową przynależność elementu x do zbioru rozmytego $\tilde{A}$.

Liczba rozmyta jest zdefiniowana jako zbiór rozmyty $\tilde{a}$ na prostej rzeczywistej R, który jest normalny (tj. $\sup_{r \in R} \mu(r;\tilde{a}) = 1$), ograniczony wypukle (tj. którego wszystkie α - przekroje są wypukłe i ograniczone), funkcja przynależności spełnia warunek ciągłości jednostronnej. Liczba rozmyta może być zilustrowana przez np. "około pięć", "trochę więcej niż 7", "mniej lub więcej pomiędzy 5 i 7" itp.

Podstawowe operacje arytmetyczne na liczbach i zbiorach rozmytych możemy określić jako:

dodawanie:

$$\mu\left(z;\tilde{a}+\tilde{b}\right) = \max_{z=x+y}\left(\mu(x;\tilde{a}) \wedge \mu\left(y;\tilde{b}\right)\right), \quad \forall\, x,y,z \in R$$

mnożenie:

$$\mu\left(z;\tilde{a}\cdot\tilde{b}\right) = \max_{z=xy}\left(\mu(x;\tilde{a}) \wedge \mu\left(y;\tilde{b}\right)\right), \quad \forall\, x,y,z \in R$$

Odnotować należy, że istnieje wiele modyfikacji wielkości rozmytych, takich jak: przedziałowe liczby rozmyte, losowe liczby rozmyte, liczby rozmyte wyższego rzędu itp.

Liczby losowe to od dawna najbardziej znana kategoria liczb opisujących zagadnienia niepewne. Wprowadzone przez Czebyszewa w 1867 r., dzisiejszą formę zawdzięczają Kołmogorowi, który w 1933 r. zaproponował teorię opartą na teorii miary, która funkcjonuje do dzisiaj. Niech trójka uporządkowana (Ω,Σ,P) oznacza abstrakcyjną przestrzeń

probabilistyczną, gdzie Ω – przestrzeń elementarnych zdarzeń losowych, Σ – σ-ciało podzbiorów borelowskich w Ω, P – miara probabilistyczna określona na Σ, por. [7, 12].

Zmienną losową rzeczywistą $x(\omega)$, $\omega \in \Omega$ definiujemy jako mierzalną funkcję określoną na Ω, o wartościach rzeczywistych x: $\Omega \rightarrow R$.

Operacje algebraiczne na zmiennych losowych $x(\omega)$, $y(\omega)$, $\omega \in \Omega$ określone są tak, jak dla funkcji rzeczywistych, czyli na ich wartościach:

$$(x+y)(\omega) := x(\omega) + y(\omega), \ \omega \in \Omega,$$
$$(xy)(\omega) := x(\omega)\, y(\omega), \ \omega \in \Omega.$$

Odnotować należy, że istnieje wiele modyfikacji wielkości losowych, takich jak: przedziałowe liczby losowe, rozmyte liczby losowe, wielkości: losowe zespolone, przedziałowe, wektorowe, macierzowe, o wartościach w przestrzeniach Hilberta, Banacha np. operatorowe czy wreszcie przestrzeniach topologicznych.

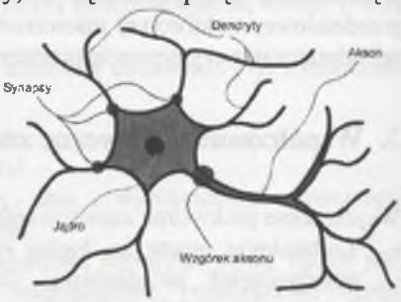
3. Współczesne praktyczne zastosowania AI

Współczesne praktyczne zastosowania *AI* obejmują takie zagadnienia, jak:

- Technologie oparte na *logice rozmytej* – powszechnie stosowane do np. sterowania pojedynczymi urządzeniami (lodówkami, pralkami, odkurzaczami itp.), skomplikowanymi systemami technicznymi (helikopterami), przebiegiem procesów technologgicznych w fabrykach, w warunkach „niepewności (braku) wszystkich danych".
- *Systemy ekspertowe* – czyli rozbudowane bazy danych z wszczepioną "sztuczną inteligencją", umożliwiającą zadawanie im pytań w języku naturalnym i uzyskiwanie w tym samym języku odpowiedzi. Systemy takie stosowane są już np. w farmacji, medycynie, diagnostyce samochodowej.
- *Maszynowe tłumaczenie tekstów* – systemy takie jak SYSTRANS, jakkolwiek wciąż bardzo ułomne, „robią szybkie postępy" i zaczynają się nadawać do tłumaczenia nawet tekstów naukowych i technicznych.
- *Sieci neuronowe* – używane z powodzeniem w wielu zastosowaniach, łącznie z programowaniem „inteligentnych przeciwników" w grach komputerowych.
- *Systemy uczące się* – dział sztucznej inteligencji zajmujący się algorytmami, które potrafią uczyć się podejmowania decyzji bądź nabywania wiedzy.
- *Eksploracja wiedzy* – omawia obszary, powiązanie z potrzebami informacyjnymi, pozyskiwaniem wiedzy, stosowaniem technik analizy, oczekiwanymi rezultatatami.
- *Rozpoznawanie obrazów* – stosowane są programy rozpoznające osoby na podstawie zdjęcia twarzy lub rozpoznające zadane obiekty na zdjęciach satelitarnych.
- *Rozpoznanie mowy i identyfikowanie rozmówców* – powszechnie już stosowane na skalę komercyjną.
- *Rozpoznawanie pisma (OCR)* – już masowo stosowane np. do automatycznego sortowania listów oraz w notatnikach elektronicznych.
- *Sztuczna twórczość artystyczna* – istnieją programy automatycznie generujące krótkie formy poetyckie, komponujące, aranżujące i interpretujące utwory muzyczne, które są w stanie skutecznie „zmylić" nawet profesjonalnych artystów, w takim sensie, że nie rozpoznają oni tych utworów jako sztucznie wygenerowanych.
- W ekonomii, powszechnie stosuje się *systemy automatycznie oceniające* m.in. zdolność kredytową, profil najlepszych klientów czy planujące kampanie reklamowe. Systemy te wcześniej poddawane są automatycznemu uczeniu, na podstawie danych (np. klientów banku, którzy regularnie spłacali kredyt i klientów, którzy mieli z tym problemy); patrz również [1-3, 5-6, 8,10, 13, 20].

4. Sieci neuronowe

W mowie potocznej często spotykamy się z określeniem mózgu jako „żywego komputera". Jest to błędne porównanie, gdyż mózg pracuje na zgoła odmiennych zasadach niż tradycyjny komputer. W dużym uproszczeniu: w zwykłym komputerze procesor wykonujący obliczenia jest oddzielony od pamięci. W mózgu zaś rolę pamięci spełnia sama jego budowa. Można powiedzieć, że mózg sam w sobie jest pamięcią. Na mózg człowieka składa się przede wszystkim bardzo duża liczba komórek mózgowych, jest ich około biliona, z czego około 100 miliardów stanowią aktywne neurony, w większości połączone ze sobą w formie skomplikowanej sieci. Ustalono, że średnio na jeden neuron przypada kilka tysięcy połączeń (nawet do 20000), ale dla poszczególnych komórek wartości te mogą się znacznie różnić.



Rzeczywiście klasyczny komputer w porównaniu z mózgiem jest szybszy, wydajniejszy i skuteczniejszy, tym niemniej są zadania, w których mózg góruje nad komputerem. Chodzi przede wszystkim o wszelkiego rodzaju rozpoznawanie i kojarzenie. Weźmy na przykład niemowlę, potrafi rozpoznać twarz matki w jednej chwili, bez względu na oświetlenie czy kąt patrzenia. Dla klasycznych algorytmów komputerowych nawet nieznaczna zmiana rozpoznawanego obrazu, jak mały obrót głowy czy inne oświetlenie, stanowią dużą trudność. Podobnie jest z rozpoznawaniem dźwięku, np. ludzkiego głosu.

Jądro – „centrum obliczeniowe" neuronu. To tutaj zachodzą procesy kluczowe dla funkcjonowania neuronu.

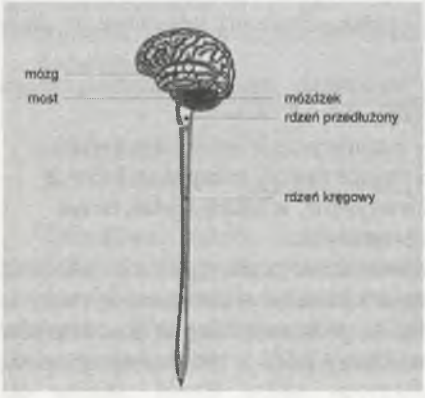
Akson – „wyjście" neuronu. Za jego pośrednictwem neuron powiadamia świat zewnętrzny o swojej reakcji na dane wejściowe. Neuron ma tylko jeden akson.

Wzgórek aksonu – stąd wysyłany jest sygnał wyjściowy, który wędruje dalej przez akson.

Dendryt – „wejście" neuronu. Tędy do jądra trafiają sygnały, które później mają być w nim poddane obróbce. Dendrytów może być wiele – biologiczne neurony mają ich tysiące.

Synapsa – jeśli dendryt jest „wejściem" neuronu, to synapsa jest jego „furtką". Może ona zmienić moc sygnału napływającego przez dendryt.

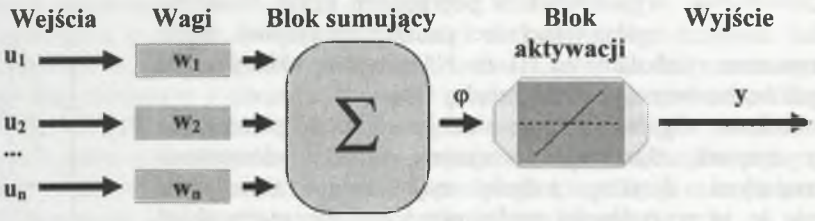
Miarą stopnia pobudzenia komórki biologicznej jest stopień depolaryzacji jej błony, zależny od sumarycznej ilości neuromediatora wydzielonego we wszystkich synapsach. W modelu sztucznej sieci neuronowej odpowiednie sygnały wejściowe muszą być pomnożone przez wagi (w modelu biologicznym jest to miara ilości neuromediatora).



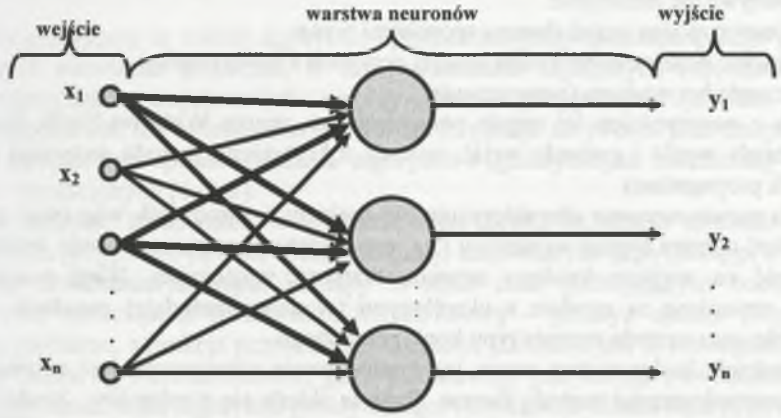
Komórki nerwowe organizują się w proste obwody i na ich podstawie można modelować sztuczne sieci neuronowe, patrz [2, 6, 10, 13, 20].

Sztuczne sieci neuronowe są modelami inspirowanymi przez strukturę i zachowanie prawdziwych neuronów. Podobnie jak mózg, mogą one rozpoznawać obrazy, przetwarzać dane i uczyć się. Analogicznie do budowy mózgu, którego podstawowym elementem jest komórka nerwowa (poniższy rysunek), dla sztucznych sieci neuronowych jest *sztuczny neuron*. Każdy neuron

otrzymuje wiele sygnałów wejściowych u_i i na ich podstawie wyznacza swoją „odpowiedź", tzn. jeden sygnał wyjściowy y związany jest z każdym oddzielnym wejściem neuronu, parametrem nazywanym wagą (weight) w_i. Określa on stopień ważności informacji docierających tym właśnie wejściem. Sygnały wejściowe (pomnożone przez odpowiednie wagi) są w neuronie sumowane, dając pewien pomocniczy sygnał wewnętrzny φ, zwany łącznym pobudzeniem neuronu (net value). Do tak utworzonej sumy sygnałów dodaje się niekiedy pewien dodatkowy składnik niezależny od sygnałów wejściowych, nazywany progiem (bias).

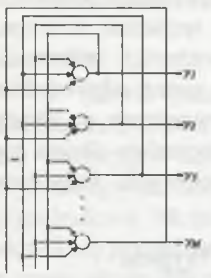


Właściwie to nawet jeden neuron można nazwać *siecią - jednoelementową*. Aby jednak móc użyć sieci w jakimś poważnym celu, trzeba użyć większej liczby neuronów. Jak widać z rysunków, neuron może mieć jeden lub więcej dendrytów (a dla każdego po jednej synapsie) i dokładnie jeden akson. Łączymy je w prosty sposób; po pierwsze układamy je warstwami. Można zbudować sieć o jednej, dwóch, lub trzech warstwach – tworzenie większej liczby warstw nie ma praktycznie sensu. Neurony należące do danej warstwy nie mają ze sobą żadnego kontaktu. Wygląda to mniej więcej tak (neurony są tu uproszczone w porównaniu z poprzednim rysunkiem – to czerwone kółko reprezentuje zarówno blok sumujący, jak i blok aktywacji):



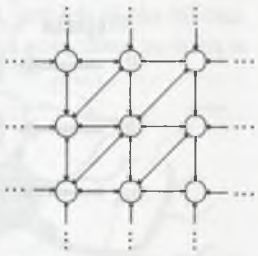
Przykładem *sieci jednokierunkowej wielowarstwowej* jest *perceptron wielowarstwowy*. Sieć tego typu ma warstwy wejściową, wyjściową oraz jedną lub więcej warstw ukrytych. Zadaniem elementów w warstwie wejściowej jest wstępne przetwarzanie obrazu wejściowego $x=[x_1, x_2, x_3, ..., x_N]$, które może obejmować normalizację lub skalowanie sygnałów.

Zasadnicze przetwarzanie neuronowe obrazu wejściowego odbywa się w *warstwach ukrytych* oraz w *warstwie wyjściowej*. Warstwy te zbudowane są z elementów przetwarzających, które stanowią modele sztucznych neuronów. Należy zauważyć, że połączenia pomiędzy poszczególnymi warstwami są zaprojektowane, tak że każdy element warstwy poprzedniej jest połączony z każdym elementem warstwy następnej. Połączeniom tym są przypisane odpowiednie współczynniki wag, które w zależności od zadania, jakie dana



sieć powinna rozwiązywać są wyznaczane dla każdego przypadku z osobna. Na wyjściu sieci generowany jest obraz wyjściowy $y=[y_1, y_2,..., y_M]$.

Sieci rekurencyjne. W tego typu sieciach występuje tzw. *sprzężenie zwrotne* między warstwą wyjściową a warstwą wejściową. Oznacza to, że sygnały na wyjściu podawane są na wejścia warstwy wejściowej, co powoduje pewną dynamikę w pracy sieci. Sygnały wejściowe w takiej sieci zależą zarówno od aktualnego stanu wejścia, jak i od sygnałów wyjściowych w poprzednim cyklu. Strukturę ogólną takiej sieci przedstawia rysunek, gdzie bloki oznaczone symbolami od N1 do N4 stanowią układy wielu sztucznych neuronów połączonych między sobą.



Sieci komórkowe. Ogólną strukturę sieci komórkowej przedstawia poniższy rysunek. Sprzężenia wzajemne między elementami przetwarzającymi dotyczą jedynie najbliższego sąsiedztwa. Połączenia te są w ogólności nieliniowe i opisane przez układ równań różniczkowych. Podstawową trudność w stosowaniu tego typu sieci stanowi opracowanie skutecznej, efektywnej i uniwersalnej metody projektowania. Typowym przykładem sieci komórkowej może być sieć typu *mapa Kohonena*.

Proces uczenia prowadzi do utrwalania określonych zachowań na bazie doświadczeń. Przez pojęcie uczenia sieci rozumie się wymuszenie na niej określonego reagowania na zadane sygnały wejściowe. Efekty uczenia mogą być różne, dlatego należy stosować proces weryfikacji „zdobytej" wiedzy. Dla sieci, podobnie jak dla organizmów żywych, stosuje się etapy uczenia, testowania i aplikacji. Oczywiście pierwsze dwa mogą być stosowane wielokrotnie. Najczęściej uczymy sieć wykonywać obliczenia przez dostrajanie wartości wag w_{ij}, które odbywa się iteracyjnie.

Sieci neuronowe można uczyć dwoma sposobami przez:

- *uczenie nadzorowane* zwane inaczej *uczeniem z nauczycielem*
- *uczenie bez nadzoru* (*samouczenie*)

Metody z nauczycielem to: reguła perceptronowa, reguła Widrowa-Hoffa, reguła delta, reguła gwiazdy wyjść i gwiazdy wejść, uczenie z krytykiem, metoda wstecznej propagacji błędu (back propagation).

Metoda samonauczania charakteryzuje się brakiem wzorcowych wag (sieć generuje je losowo). Sieć odbiera sygnał wejściowy i na jego podstawie wyznacza swoje wyjście, ocenia ona wartość na wyjściu każdego neuronu warstwy wyjściowej. Wagi poszczególnych neuronów zmieniane są zgodnie z określonymi (w danej metodzie) zasadami. Znane są: reguła Hebba oraz metoda uczenia typu konkurencyjnego.

W dziedzinie budownictwa znane jest zastosowanie sztucznych sieci neuronowych do badania mrozoodporności metodą Borasa. Baza ta składa się z rekordów. Struktura danych jest 6+2, co oznacza, że 6 atrybutów w każdym rekordzie dotyczy wejścia, a 2 wyjścia.

Atrybuty wejściowe – wyniki badań: gęstość betonu, zawartość powietrza w stwardniałym betonie, powierzchnia właściwa porów, współczynnik rozstawu porów, moduł sprężystości betonu oraz obecność pyłów krzemionkowych.

Atrybuty wyjściowe: 28-dniowa wytrzymałość betonu na ściskanie, mrozoodporność. Mrozoodporność została określona metodą Borasa, przy czym przyjęto, że beton mrozoodporny to ten, którego ubytek masy przy powierzchniowym łuszczeniu po 56 cyklach zamarzania nie przekracza 1 [kg/m]. Wartość drugiego atrybutu wyjściowego odpowiada betonowi, który nie jest mrozoodporny.

Inny przykład to analiza stopnia zużycia technicznego budynków. Wyniki badań uzyskano za pomocą radialnych sieci neuronowych (sieć RBF od ang. Radial Basic Function Neural Networks).

Wektor wejścia miał 9 cech, a wyjściem był skalar e, określający stopień zużycia budynku: $x=[x_1, x_2, x_3, ..., x_9]$, $y=e$, x_1 – rok budowy, x_2 – rodzaj zabudowy, x_3 – rodzaj pokrycia dachu, x_4 – sposób podpiwniczenia, x_5 – warunki gruntowe, x_6 – poziom wód gruntowych, x_7 – poziom natężenia ruchu ulicznego, x_8 – zdarzenia losowe (pożar, powódź), x_9 – sposób utrzymania obiektu (remonty). Porównywano wyniki uzyskane z pomiarów (ocen eksperckich) z wynikami otrzymanymi za pomocą sieci RBF II.

Sieci neuronowe stanowią (w jakimś zakresie) naśladownictwo działania ludzkiego umysłu. Stosowane są do problemów związanych z tworzeniem modeli matematycznych, w rozpoznawaniu dźwięków i obrazów (mowy i pisma), do prognozowania, klasyfikacji i rozpoznawania stanów obiektów ekonomicznych, w medycynie, w zastosowaniach inżynierskich – ocena stanu zużycia, do analizy, kojarzenia i optymalizacji w podsystemach doradczych w procesach zarządzania; wreszcie w predykcji, klasyfikacji oraz do sterowania robotami, do kompresji, kodowania oraz uzupełniania niekompletnych danych.

„Czym jest świadomość, tego nie wiemy. To, że coś tak godnego uwagi jak stany świadomości powstaje jako rezultat pobudzenia komórek nerwowych jest tak samo niewytłumaczalne, jak pojawienie się dżina wskutek pocierania lampki Aladyna". (Thomas Henry Huxley 1866).

5. Algorytmy genetyczne

Algorytm genetyczny to rodzaj algorytmu optymalizacyjnego przeszukującego przestrzeń alternatywnych rozwiązań problemu, w celu wyszukania najlepszych rozwiązań. Sposób działania algorytmów genetycznych nieprzypadkowo przypomina zjawisko ewolucji biologicznej, ponieważ ich twórca John H. Holland inspiracje do swoich prac czerpał właśnie z biologii. Obecnie ten sposób działania algorytmów genetycznych zalicza się do grupy algorytmów ewolucyjnych [1-4, 8].

Problem definiuje środowisko, w którym istnieje pewna *populacja osobników*. Każdy z osobników ma przypisany pewien zbiór informacji stanowiących jego *genotyp*, a będących podstawą do utworzenia *fenotypu*. Fenotyp to zbiór cech podlegających ocenie *funkcji przystosowania,* modelującej środowisko. Innymi słowy – genotyp opisuje proponowane rozwiązanie problemu, a funkcja przystosowania ocenia, jak dobre jest to rozwiązanie.

Genotyp składa się z *chromosomów*, gdzie zakodowany jest *fenotyp* i ewentualnie pewne informacje pomocnicze dla algorytmu genetycznego. Chromosom składa się z genów.

Wspólnymi cechami *algorytmów ewolucyjnych*, odróżniającymi je od innych, tradycyjnych *metod optymalizacji*, są:

1. używanie operatorów genetycznych, które dostosowane są do postaci rozwiązań,
2. przetwarzanie populacji rozwiązań, prowadzące do równoległego przeszukiwania przestrzeni rozwiązań z różnych punktów,
3. w celu ukierunkowania procesu przeszukiwania wystarczającą informacją jest jakość aktualnych rozwiązań,
4. celowe wprowadzenie elementów losowych.

Najczęściej działanie algorytmu przebiega następująco:

1. losowana jest pewna populacja początkowa;

2. populacja poddawana jest ocenie (*selekcja*); najlepiej przystosowane osobniki biorą udział w procesie *reprodukcji;*
3. genotypy wybranych osobników poddawane są operacjom ewolucyjnym: są ze sobą kojarzone przez złączanie genotypów rodziców (*krzyżowanie*), przeprowadzana jest mutacja, czyli wprowadzenie drobnych losowych zmian;
4. rodzi się drugie *(kolejne)* pokolenie i algorytm powraca do kroku drugiego, jeżeli nie znaleziono dostatecznie dobrego rozwiązania; w przeciwnym wypadku uzyskujemy wynik.

Kodowanie jest bardzo istotnym etapem projektowania algorytmu. Najczęściej stosowane są kodowania chromosomu:

- wektorem genów, z których każdy z nich może być jedno- lub wielobitową liczbą całkowitą bądź też liczbą rzeczywistą,
- za pomocą drzewiastych struktur danych.

Proces wyboru osobników poddanych ocenie jest wykonywany według określonych kryteriów. Zapisuje się je w postaci *funkcji oceny* albo inaczej *funkcji przystosowania.* Algorytm genetyczny dąży zwykle do jej minimalizacji (maksymalizacji). Jako kryterium często przyjmowana jest funkcja celu rozważanego problemu optymalizacji.

Funkcja oceny to *miara jakości* dowolnego osobnika (fenotypu, rozwiązania) w populacji. Dla każdego osobnika jest ona obliczana na podstawie pewnego modelu rozwiązywanego problemu. Załóżmy dla przykładu, że chcemy zaprojektować obwód elektryczny o pewnej charakterystyce. Funkcja oceny będzie premiowała rozwiązania najbardziej zbliżone do tej charakterystyki, zbudowane z najmniejszej liczby elementów. W procesie selekcji faworyzowane będą najlepiej przystosowane osobniki; staną się one"rodzicami" dla populacji.

Istnieje wiele *metod selekcji.* Dla przykładu można przedstawić tzw. metodę ruletki. Wirtualnie budujemy koło, którego wycinki odpowiadają poszczególnym osobnikom. Im lepszy osobnik, tym zajmuje większy wycinek koła. Rozmiar wycinków może zależeć od wartości funkcji oceny, jeśli wysoka wartość oceny oznacza wysokie przystosowanie. W takim układzie prawdopodobieństwo, że lepszy osobnik zostanie wybrany „jako rodzic", jest większe. Niestety przy takim algorytmie ewolucja z każdym krokiem zwalnia. Jeżeli osobniki są podobne, to każdy dostaje równy wycinek „koła fortuny" i presja selekcyjna spada. Algorytm słabiej rozróżnia osobniki dobre od słabszych.

Pozbawiona tej wady jest metoda rankingowa. Dla każdego osobnika obliczamy *funkcje oceny* i ustawiamy je w szeregu najlepszy-najgorszy. Pierwsi na liście dostają prawo do rozmnażania, a reszta trafia do historii. Wadą metody jest niewrażliwość na różnice pomiędzy kolejnymi osobnikami w kolejce. Może się okazać, że sąsiadujące na liście rozwiązania mają różne wartości funkcji oceny, ale dostają prawie taką samą liczbę potomstwa.

Istnieją także metody *selekcji wielokryterialnej.* Tworzymy kilka różnych funkcji oceny (oceniających pewne wybrane cechy osobników osobno). Dla przykładu osobniki mogą być ułożone nie w jednym, ale w kilku szeregach najlepszy-najgorszy, a proces selekcji jest bardziej złożony.

Jak widać, *selekcja* daje większe prawdopodobieństwo reprodukcji osobnikom o dużym przystosowaniu, więc kolejne pokolenia są coraz lepiej przystosowane. Spada jednak różnorodność genotypu populacji – zostaje ona z czasem zmonopolizowana przez nieznacznie różniące się (lub wręcz identyczne) odmiany tego samego osobnika. Objawia się to zbieżnością kolejnych, najlepszych rozwiązań do pewnej granicy. Czasami zbieżność jest przedwczesna, a ewolucja utyka i uzyskane rozwiązania przedstawiają pewne ekstrema lokalne. Mogą być one dalekie od oczekiwanych *rozwiązań globalnych,* czyli tych najlepszych w całej przeszukiwanej przestrzeni.

W każdym cyklu (każde pokolenie) poddawane są „obróbce" za pomocą *operatorów ewolucyjnych*. Celem tego etapu jest wygenerowanie nowego pokolenia, na podstawie poprzedniego, które być może będzie lepiej dopasowane do założonego środowiska.

Operator krzyżowania ma za zadanie łączyć w różnych kombinacjach cechy pochodzące z różnych osobników populacji, a *operator mutacji* ma za zadanie zwiększać różnorodność tych osobników. O przynależności dowolnego algorytmu do klasy algorytmów genetycznych głównie decyduje zastosowanie *operatora krzyżowania* i praca z całymi populacjami osobników (idea łączenia w przypadkowy sposób *genotypów* nieprzypadkowo wybranych osobników). Równie ważny jest *operator mutacji*. Jeśli krzyżowanie traktować jako sposób *eksploatacji* przestrzeni rozwiązań, to mutacja jest sposobem na jej *eksplorację*. Może się jednak zdarzyć, że dla niektórych zagadnień jej zastosowanie nie jest krytyczne.

Krzyżowanie polega na połączeniu niektórych (wybierane losowo) genotypów w jeden. Kojarzenie ma sprawić, że potomek dwóch osobników rodzicielskich ma zespół cech, który jest kombinacją ich cech (może się zdarzyć, że tych najlepszych).

Sposób krzyżowania jest zależny od kodowania chromosomów i specyfiki problemu. Jednak można wskazać kilka standardowych metod krzyżowania:

- rozcięcie dwóch chromosomów i stworzenie nowego, przez sklejenie lewej części jednego rodzica z prawą częścią drugiego rodzica (dla chromosomów z kodowaniem binarnym i całkowitoliczbowym),
- stosowanie operacji logicznych (kodowanie binarne),
- obliczenie wartości średniej genów (kodowanie liczbami rzeczywistymi).

Mutacja wprowadza do genotypu losowe zmiany. Jej zadaniem jest wprowadzanie różnorodności w populacji, czyli zapobieganie (przynajmniej częściowe) przedwczesnej zbieżności algorytmu. Mutacja zachodzi z pewnym przyjętym prawdopodobieństwem – zazwyczaj rzędu *1%*. Jest ono niskie, ponieważ zbyt silna mutacja przynosi efekt odwrotny do zamierzonego: zamiast subtelnie różnicować dobre rozwiązania – niszczy je. Stąd w procesie ewolucji mutacja ma znaczenie drugorzędne, szczególnie dla długich chromosomów.

Jeśli chodzi o chromosomy kodowane binarnie, losuje się zazwyczaj dwa geny i zamienia się je miejscami bądź np. neguje pewien wylosowany gen.

W przypadku genotypów zakodowanych liczbami całkowitymi stosuje się permutacje, natomiast przy genotypach zakodowanych liczbami rzeczywistymi wprowadza się do przypadkowych genów losowe zmiany o danym rozkładzie – najczęściej normalnym.

Algorytmy genetyczne znajdują zastosowanie tam, gdzie sposób rozwiązania problemu nie jest dobrze określony lub poznany, ale znany jest sposób oceny jakości rozwiązania. Przykładem jest np. problem komiwojażera, gdzie należy znaleźć najkrótszą drogę łączącą wszystkie miasta, tak by przez każde miasto przejść tylko raz. Ocena jakości proponowanej trasy jest błyskawiczna, natomiast znalezienie optymalnej trasy kwalifikuje się do klasy problemów NP-zupełnych. Przy zastosowaniu podejścia ewolucyjnego dobre rozwiązanie można znaleźć bardzo szybko, ale oczywiście pewni możemy być jedynie uzyskania rozwiązań suboptymalnych, co wynika z formalnie opisanej trudności problemów klasy NP. Algorytmy genetyczne „dobrze radzą" sobie także w znajdowaniu przybliżeń ekstremów funkcji, których nie da się obliczyć analitycznie.

Algorytmy genetyczne wykorzystywane są również do zarządzania populacją sieci neuronowych. Projektowanie maszyn bądź obwodów elektrycznych jest doskonałym polem, na którym mogą się wykazywać algorytmy genetyczne. Inżynierowi podczas tworzenia nowych pomysłów nie chodzi o znalezienie najlepszego, możliwego rozwiązania; wystarczy tylko przybliżone spełnienie granicznych warunków oraz optymalizacja projektu. Algorytmy genetyczne, w odróżnieniu od człowieka, nie działają schematycznie. Program nie zna wcześniejszych projektów i dlatego czasami „wykazuje się pewną inwencją". Co więcej,

człowiek często opiera się na bardzo przybliżonych modelach, które dają fałszywy obraz problemu. Algorytm genetyczny może przeanalizować złożony model/zagadnienie i znaleźć rozwiązanie, na które człowiek by nie wpadł.

Algorytmy genetyczne zapewniają skuteczne mechanizmy przeszukiwania dużych przestrzeni rozwiązań. Ponieważ *grupowanie* należy do tej kategorii zadań, to oczywiste jest, że algorytmy genetyczne stosowane są w grupowaniu. Algorytmy genetyczne są bardziej niezależne od wstępnej inicjalizacji oraz mniej skłonne do znajdowania lokalnych rozwiązań w miejsce optymalnych. Przykładem może być zagadnienie *grupowania,* w którym w miejscu klasycznych algorytmów z powodzeniem stosuje się algorytmy genetyczne.

System klasyfikatorów jest rodzajem genetycznego systemu uczącego się. Podstawy teoretyczne zostały opracowane przez Hollanda (1962), a pierwsza praktyczna implementacja miała miejsce w 1978 r. i polegała na treningu systemu w pokonywaniu labiryntu. Przedmiotem badań [11] było określenie możliwości zastosowania systemu klasyfikatorów SK (CS – classifier system) do *optymalizacji kratownic.* Jakkolwiek znane są przykłady zastosowania do tego celu metod analitycznych, algorytmów genetycznych, ewolucyjnych i innych. Przedstawiono zasadę działania systemu klasyfikatorów, użytego do optymalizacji kształtu płaskiej kratownicy. Funkcją celu była waga konstrukcji, zmiennymi decyzyjnymi przekroje elementów oraz współrzędne węzłów.

W celu przebadania *SK* autor [11] napisał obiektowo zorientowany program w języku C++, wykorzystując do prezentacji danych mechanizmy biblioteki *STL (Standard Template Library)* oraz język *VRML (Virtual Reality Modelling Language).* Algorytm jednej iteracji działa w sposób przedstawiony na rys 1. Najpierw następuje wygenerowanie wiadomości ze środowiska, zawierającej zakodowane dane dotyczące optymalizowanego węzła. Są w niej zawarte informacje na temat liczby dochodzących do węzła elementów, ich wzajemnego ułożenia, przekroju, działających sił i wytężenia. Następnie z bazy danych wybierana była lista klasyfikatorów pasujących do tej wiadomości.

Na rys. 2 przedstawiono postęp optymalizacji kratownicy. Im większa ilość iteracji, tym lepszy jest końcowy efekt.

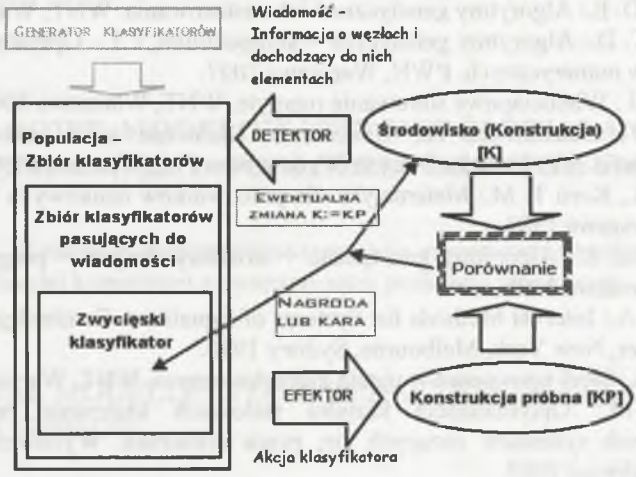
6. Wnioski

Czego mimo wielu wysiłków nie udało się dotąd osiągnąć:

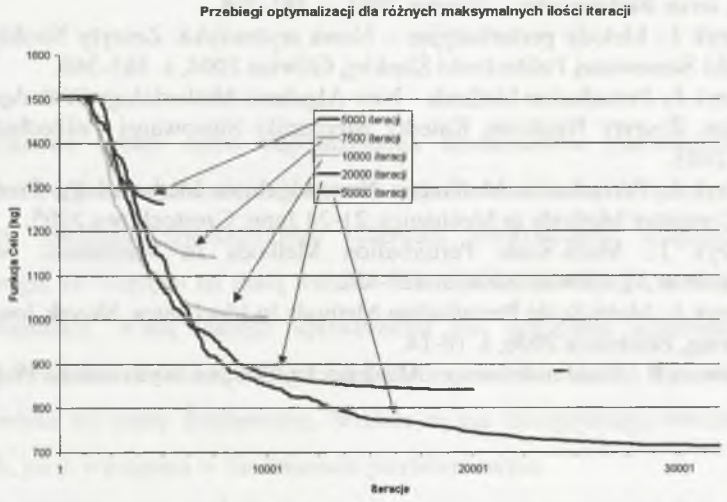
- *Programów skutecznie wygrywających w niektórych grach,* np. *w brydżu, warcabach.* Programy do gry w szachy, w które zainwestowano jak dotąd najwięcej wysiłku i czasu spośród wszystkich tego rodzaju programów, osiągnęły bardzo wysoki poziom, który pozwolił na wygranie nawet z mistrzem świata Garri Kasparowem (1997).
- *Programu, który umie idealnie naśladować człowieka, rozmawiając przy użyciu tekstu* i potrafiłby przejść test Turinga.
- *Programu, który potrafiłby skutecznie generować zysk, grając na giełdzie.* Problemem jest masa informacji, którą taki program musiałby przetworzyć i sposób ich kodowania przy wprowadzaniu do komputera.
- *Programu skutecznie tłumaczącego teksty literackie i mowę potoczną.* Istnieją programy do automatycznego tłumaczenia, ale sprawdzają się one tylko w bardzo

ograniczonym stopniu. Podstawową trudnością jest tu złożoność i niejasność języków naturalnych, a w szczególności brak zrozumienia (przez program) znaczenia tekstu.

- *Programów, które mogą dorównać człowiekowi w rozpoznawaniu obrazów i np. w prowadzeniu samochodu.*



Rys. 1. Schemat działania algorytmu SK w jednej iteracji [11]
Fig. 1. CS Algorithm scheme of single iteration [11]



Rys. 2. Wykres funkcji celu (waga najlepszej konfiguracji kratownicy) w danej iteracji [11]
Fig. 2. Diagram of fitness (weight of best structure) in a given iteration [11]

LITERATURA

1. Arabas J.: Wykłady z algorytmów ewolucyjnych. WNT, Warszawa 2001.
2. Diagnostyka procesów – modele, metody sztucznej inteligencji, zastosowania. Pod red. J. Korbicza, J. M. Kościelnego, Z. Kowalczuka, W. Cholewy. WNT, Warszawa 2002.
3. Goldberg D. E.: Algorytmy genetyczne i ich zastosowania. WNT, Warszawa 1995.
4. Gwiazda T. D.: Algorytmy genetyczne – kompendium, t. I – Operator krzyżowania dla problemów numerycznych. PWN, Warszawa 2007.
5. Kacprzyk J.: Wieloetapowe sterowanie rozmyte. WNT, Warszawa 2001.
6. Korbicz J., Obuchowicz A., Uciński D.: Sztuczne sieci neuronowe, podstawy i zastosowania. Akademicka Oficyna Wydawnicza PLJ, Warszawa 1994.
7. Korn G. A., Korn T. M.: Matematyka dla pracowników naukowych i inżynierów, cz.I. PWN, Warszawa 1983.
8. Michalewicz Z.: Algorytmy genetyczne + struktury danych = programy ewolucyjne. WNT, Warszawa 1996.
9. Neumaier A.: Interval Methods for Systems of Equations. Cambridge University Press, Port Chester, New York, Melbourne, Sydney 1990.
10. Osowski S.: Sieci neuronowe w ujęciu algorytmicznym. WNT, Warszawa 1996.
11. Rabijasz M.: Optymalizacja kształtu stalowych kratownic przy zastosowaniu genetycznych systemów uczących się, praca doktorska. Wydawnictwo Politechniki Śląskiej, Gliwice 2007.
12. Reinhardt F., Soeder H.: Atlas matematyki. Prószyński i Spółka, Warszawa 2006.
13. Rutkowska D., Piliński M., Rutkowski L.: Sieci neuronowe, algorytmy genetyczne i systemy rozmyte. PWN, Warszawa – Łódź 1997.
14. Skrzypczyk J.: Perturbation methods – New arithmetic. Zeszyty Naukowe Politechniki Śląskiej, seria: Budownictwo, Gliwice 2003, s. 391-398.
15. Skrzypczyk J.: Metody perturbacyjne – Nowa arytmetyka. Zeszyty Naukowe Katedry Mechaniki Stosowanej Politechniki Śląskiej, Gliwice 2004, s. 363-368.
16. Skrzypczyk J.: Perturbation Methods – New Algebraic Methodology With Applications In Mechanics. Zeszyty Naukowe Katedry Mechaniki Stosowanej Politechniki Śląskiej, Gliwice 2005.
17. Skrzypczyk J.: Perturbation Methods – New Algebraic Methodology. Proc. of CMM-2005 – Computer Methods in Mechanics, 21-24 June, Częstochowa 2005.
18. Skrzypczyk J.: Multi-Scale Perturbation Methods In Mechanics. Modelowanie Inżynierskie nr 32, Gliwice 2006, s. 427-432.
19. Skrzypczyk J.: Multi-Scale Perturbation Methods In Mechanics. Slovak Journal of Civil Engineering, Bratislava 2006, s. 10-14.
20. Tadeusiewicz R.: Sieci neuronowe. Akademicka Oficyna Wydawnicza PLJ, Warszawa 1995.