SIMULATED ANNEALING – 30 LAT PÓŹNIEJ

CZYLI JAK DALEKO NAM DO MYŚLĄCYCH MASZYN

Lewandowski Krzysztof
Jedynak Mirosław

1. Wstęp

W pieśniach starożytnego rzymianina Horacego możemy odnaleźć hasło, które dla wielu z nas jest aktualne aż do dzisiaj. „*Aurea mediocritas*" – złote umiarkowanie, czy inaczej zasada złotego środka. Poszukujemy sposobu, aby nasze życie było pełne spokoju, radości i szczęścia… Niestety ziemskie bytowanie jest ciągłym zmaganiem się w dążeniu do takiego stanu. Ponieważ nie jest to traktat filozoficzny zastanówmy się, jakiego „złotego środka" poszukujemy w informatyce. Niewątpliwie od czasu powstania pierwszych komputerów w latach pięćdziesiątych ubiegłego wieku czynione były usilne starania w celu stworzenia maszyny myślącej. Inaczej mówiąc ciągle dążymy do stworzenia sztucznej inteligencji.

Hasło „sztuczna inteligencja" zostało sformułowane w 1956 roku przez Johna McCarthy'ego. Samych definicji tego pojęcia jest całe multum, przytoczymy tylko dwie następujące:

- „*[Automatyzacja] czynności, które kojarzymy z myśleniem ludzkim, czynności takich, jak podejmowanie decyzji, rozwiązywanie problemów, uczenie się…*" (Bellman, 1978)
- „*Dziedzina informatyki nastawiona na automatyzację inteligentnych zachowań.*" (Luger, Stubblefield, 1993)

Trudności w sprecyzowaniu terminu spowodowały, że zamiast ścisłej definicji zaproponowano test. Taki najsłynniejszy to test Alana Turinga z 1950 roku. Mówiąc krótko test jest zaprojektowany tak, że osoba siedząca przed dalekopisem ma stwierdzić czy rozmawia z człowiekiem czy z maszyną.

Początkowo uważano, że maszynę myślącą uda się zbudować stosunkowo szybko. Mówiono, że sztuczna inteligencja osiągnie swoje cele w ciągu 3 - 8 lat. Rozpoczęto pracę nad metodami rozwiązywania problemów i ich zastosowaniem w programach ogólnego przeznaczenia, tzw. General Problem Solving. Świetlana przyszłość, jaką otwierała sztuczna inteligencja spowodowała, że szereg instytucji wspierał prowadzone badania. W 1982 roku rząd Japonii ogłosił rozpoczęcie programu nazwanego Systemy Komputerowe Piątej Generacji, który miał na celu wprowadzenie badań nad AI na następny poziom. Skala działań i ambicje wykonawców były bezprecedensowe i wzbudziły wśród naukowców z Zachodu obawy, że Japończycy zdominują przemysł komputerowy tak, jak już to uczynili w przypadku elektroniki i produkcji samochodów. Swoisty wyścig zbrojeń wywiązał się między USA i Japonią. Rząd USA z tego powodu zmienił prawo antymonopolowe, tak aby grupa amerykańskich korporacji (takich gigantów jak Kodak i Motorola) mogła połączyć siły przeciw Japończykom. Utworzono Korporację Mikroelektroniki i Technologii Komputerowych (MCC), której przewodniczył Doug Lenat, naukowiec z Uniwersytetu Stanforda. Spore pieniądze na badania łożył Pentagon poprzez swoją agencję DARPA. Jednakże frustrująco powolny postęp badań i spory wewnątrz grup badawczych

spowodowały, że nastał okres zwany „zimą” sztucznej inteligencji. Postanowiono szukać nowych kierunków badań. Zaczęto pracę nad sieciami neuronowymi, sztucznym życiem… Naukowcy skupili się nad konkretniejszymi zagadnieniami, takimi jak rozpoznawanie obrazu, mowy, tekstu, automatyczne planowanie, sterowanie robotami itp. Liczyli, że mając precyzyjniej określone cele, będą w stanie łatwiej je osiągnąć. Zaowocowało to rozbiciem nauki o sztucznej inteligencji na szereg dyscyplin. Jedną z szerszych i bardziej ogólnych, nad którą się tutaj zatrzymamy są metody heurystyczne.

1.1. Heurystyka

Słowo „heurystyka” pochodzi od greckiego czasownika „*heuriskein*”, oznaczającego „znaleźć” lub „odkryć”. W starożytnych podaniach mówi się, że Archimedes biegał nagi po ulicy wykrzykując „Heureka” (Znalazłem to) po tym, jak odkrył zasadę wyporu w swojej wannie. Kolejne pokolenia przetransformowały słowo „Heureka” w „Eureka” i tak zostało po dziś dzień. Techniczne znaczenie słowa heurystyka przeszło kilka zmian w ciągu lat rozwoju sztucznej inteligencji. W 1957 roku Georgie Polya w książce „How to Solve It” (Jak to rozwiązać) użył słowa „heurystyka” odnosząc się do metod odkrywania i wymyślania technik rozwiązywania problemów, a w szczególności problemu odrywania dowodów matematycznych. Inni używali słowa heurystyczny jako przeciwieństwo do algorytmiczny. Na przykład Newell, Shaw i Kimon w roku 1963 stwierdzili, że „*proces, który może rozwiązać dany problem, ale nie daje żadnych tego gwarancji jest nazywany heurystyką dla tego problemu*.” Jednak należy zauważyć, że nie oznacza to, że metody heurystyczne są losowe czy niedeterministyczne. Trzeba wyczuć różnice pomiędzy czymś niealgorytmicznym a czymś, co nie jest precyzyjnie scharakteryzowane. Początkowo heurystyka dominowała w zastosowaniach sztucznej inteligencji, pierwszy system ekspertowy tworzony przez naukowców z uniwersytetu w Stanford został nazwany projektem programowania heurystycznego (HPP – Heuristic Programming Chalange). Za heurystykę uważano praktyczną zasadę służąca do generowania dobrych rozwiązań bez ekstensywnego poszukiwania. Początkowo heurystyki były zapisywane bezpośrednio w strukturach tworzonych programów, ale takie podejście okazało się być zbyt mało elastyczne, kiedy potrzebna była duża liczba heurystyk. Stopniowo systemy były projektowane tak, aby mogły akceptować heurystyczne informacje wyrażone jako zasady i w ten sposób narodziły się tzw. „rule-based systems”, czyli systemy oparte na zasadach heurystycznych. W dzisiejszych czasach częściej używa się przymiotnika „heurystyczny” dla określenia technik, które poprawiają pośrednie rozwiązania w zadaniach rozwiązywania problemów. Co, na przykład, w dziedzinie algorytmów wyszukujących odnosi się do funkcji zapewniających szacunkowy koszt rozwiązania. Podsumowując, w wielkim skrócie heurystyka to wszelkie metody pozwalające algorytmowi poszukującemu rozwiązania pójść „na skróty”.

1.2. Od czego się zaczęło?

Zanim przejdziemy do nowoczesnych metod heurystycznych zastanówmy się, jakie źródła mają nowoczesne algorytmy rozwiązywania problemów. W literaturze możemy wyróżnić trzy główne mechanizmy stosowane w algorytmach wyszukujących:

- szukanie analityczne – sterowane za pomocą matematycznych funkcji, np. w optymalizacji algorytmy mogą być zależne od gradientu czy hesjanu, gwarantują odnalezienie istniejącego rozwiązania, ale zwykle w praktyce dostarczają rozwiązań nieglobalnych
- ślepe szukanie – inaczej nazywane poszukiwaniem niesterowalnym

- szukanie heurystyczne – sterowane szukanie, szeroko stosowane w praktyce, dostarcza zwykle satysfakcjonujących rozwiązań (choć nie ma gwarancji, że są to rozwiązania globalne)

Każdy proces poszukiwania można podzielić na trzy główne fazy – wybranie punktu startowego poszukiwań, inicjacja – generacja kolejnych rozwiązań – zakończenie po spełnienie przez rozwiązanie przyjętego kryterium. Przy projektowaniu algorytmów poszukujących należy rozważyć cały szereg spraw mających decydujący wpływ na postać algorytmu, stosowanych mechanizmów i metod. Począwszy od wybrania właściwego punktu startowego przez określenie skali naszego problemu, jego związków z rzeczywistością do zdefiniowania typu wiedzy, musimy zanalizować dany problem, tak, aby otrzymać satysfakcjonujące nas rozwiązanie. Dla przykładu tzw. wiedza pozytywna oznacza, że algorytm „nagradza" dobra rozwiązania a "karze" złe. Zwykle w praktycznych zastosowaniach mamy do czynienia z problemami wielkiej skali i właśnie dla takich problemów idealne wydaje się być stosowanie metod szukania heurystycznego. Nowoczesne algorytmy heurystyczne są inspirowane wiedzą z innych dziedzin takich, jak biologia, mechanika statystyczna, neurologia czy fizyka, żeby wymienić tylko kilka. Zwykle jądrem tych metod jest generowanie kolejnych rozwiązań z lokalnego sąsiedztwa wcześniejszych rozwiązań. Sąsiedztwo może być zdefiniowane za pomocą różnych norm, najpopularniejszą jest norma euklidesowa definiująca sąsiedztwo danego rozwiązania x jako zbiór rozwiązań będący w odległości nie większej niż б od x. W teorii problemów NP-zupełnych, do których głównie stosuje się metody heurystyczne, dla niektórych przypadków dowodzi się, że dokładne rozwiązania takich problemów mogą być aproksymowane przez rozwiązania, które otrzymujemy wykorzystując metody heurystyczne. Zwykle takie rozwiązania nie różnią się od rozwiązań optymalnych o więcej niż kilka zadanych procentów. Kwesta ta jest szerzej omówiona w książce „Computers and Intractability – A Guide to the theory of NP-Completness" autorstwa M. Garey'a i D. Johnsona.

2. Simulated annealing

Mimo, że metoda nie należy do metod najnowszych – została opisana przez Kirkpatrick'a w 1983 r. w Science – to może ona śmiało konkurować z dużo nowszymi metodami takimi jak algorytmy mrówkowe (początek lat 90.) czy algorytmami genetycznymi (1996). Symulowane wyżarzanie to algorytm z rodziny algorytmów „Generuj i Testuj" i stanowi modyfikację algorytmu wspinaczki (ang. *Simple Hill Climbing*), w którym dopuszcza się na początku przejścia ze stanu bieżącego także do stanów gorszych. Dzięki temu można uniezależnić się od punktu startowego i uzyskać możliwość przebadania znacznie większego obszaru przestrzeni rozwiązań.

Symulowane wyżarzanie wzorowane jest na fizycznym procesie nazywanym wyżarzaniem, w którym to pewne ciała, np. metal, są rozgrzewane a następnie stopniowo schładzane aż do osiągnięcia krystalizacji. Celem tego procesu jest osiągnięcie jak najmniejszego stanu energetycznego obrabianego ciała.

Energia stanu ciała odpowiada funkcji celu, a absolutne minimum tej energii - minimum globalnemu. W procesie powolnego wyżarzania, krystalizacji ciała towarzyszy globalne zmniejszanie energii, ale są również dopuszczalne stany, którym towarzyszy chwilowe jej zwiększenie. Dzięki dopuszczeniu chwilowego wzrostu stanu energetycznego możliwe jest opuszczenie minimum lokalnego, które może pojawić się w trakcie procesu. Dopiero zejście z temperaturą do zera absolutnego uniemożliwia jakiekolwiek podniesienie poziomu energetycznego. Wówczas to zmiany energetyczne mogą zachodzić już tylko w kierunku minimum.

2.1. Elementy algorytmu symulowanego wyżarzania

- Reprezentacja możliwych rozwiązań
- Generator losowych zmian w rozwiązaniu
- Funkcja realizująca ocenę rozwiązania – funkcja celu
- Schemat wyżarzania/ chłodzenia – temperatura początkowa i zasady jej obniżania w procesie poszukiwania minimum funkcji celu

2.2. Schemat schładzania

Podczas przebiegu algorytmy wyżarzania konieczne jest zastosowanie metody zmniejszającej prawdopodobieństwo przejścia ze stanu o gorszych parametrach. Takie zasady postępowania są nazywane *schematem schładzania* (*ang. cooling schedule).*

Aby zdefiniować schemat schładzania należy podać:

- Początkową temperaturę T_0
- Końcową temperaturę lub kryterium stopu
- Długość łańcuchów Markowa (zależna od ilości parametrów i złożoności rozwiązania)
- Regułę obniżania temperatury

W podstawowej wersji algorytmu temperatura jest obniżana proporcjonalnie- w każdym kroku jej wartość jest dzielona przez liczbę z przedziału (1, ∞). Możliwe jest jednak obniżanie temperatury w sposób bardziej skomplikowany, co pozwala na regulowanie tempa spadku w różnych fazach przebiegu algorytmu. Pierwsza metoda to:

$$T_{k-1} = \alpha \times T_k, \ 0 < \alpha < 1$$

Druga metoda to:

$$T_{k+1} = T_k e^{-\left(\frac{T_k \Delta(E)}{\sigma_{T_k}^2}\right)}$$
$$\Delta(E) = E_{T_k} - E_{T_{k-1}}$$

Kiedy $\sigma_{T_k}^2$ jest duże, co ma miejsce na początku działania algorytmu, procedura zachowuje się jak przy przeszukiwaniu losowym – zmiany temperatury będą bardzo powolne. Przeciwne zjawisko zachodzi, kiedy $\sigma_{T_k}^2$ jest małe, co oznacza, że prawdopodobnie jesteśmy blisko minimum.

Prawdopodobieństwo przejścia do wyższego stanu energetycznego określa wzór:

$$p = e^{\frac{\Delta E}{kT}}$$

gdzie :

ΔE - wartość o jaką zwiększyła się energia stanu

T – temperatura, przy jakiej następuje przejście do wyższego poziomu

k – stała Boltzmanna'a

Metoda symulowanego wyżarzania będąca odpowiednikiem fizycznego procesu wyżarzania, uznawana jest za jeden z nielicznych algorytmów umożliwiających praktyczne uzyskanie minimum globalnego funkcji wielu zmiennych. Algorytm samej metody przedstawia się następująco

2.3. Szkic algorytmu

```
Initialize the temperature to T
initialize the chain length to L
initialize the neighborhood length to ζ
x_0 ∈ Θ(x), f_0(x_0)
initialize optimal solution x_opt to be x_o and its objective value
f_opt = f_0
initialize current solution x to be x_o and its objective value f' = f_0
repeat
     for j = 0 to L
          i = i + 1
          x_i ∈ neighbourhood(x, ζ), f_i = f(x_i)
          Δ(f) = f_i − f'
          if  f_i < f_opt then  x_opt = x_i, f_opt = f_i
          if  f_i < f' then  x = x_i, f' = f_i
          else if exp(−Δ(f)/T) > random(0,1)
                then  x = x_i, f' = f_i
     update L and T
until loop condition is satisfied
return x_opt and f_opt
```

2.4. Charakterystyka – heurystyka

Symulowane wyżarzanie może rozwiązywać silnie nieliniowo zależne, nieuporządkowane modele z dużym poziomem szumów i z wieloma zależnościami. W związku z stosunkowo dobrym rozwiązywaniem problemów nieuporządkowanych chaotycznych w przypadku modeli finansowych tj. modelowanie rynku czy prognozowanie popytu, symulowane wyżarzanie często dużo szybciej zbiega do optymalnego rozwiązania niż inne heurystyczne metody- algorytmy genetyczne czy sieci neuronowe.

Oprócz łatwości znalezienia lokalnego minimum algorytm potrafi wydostać się z takiego położenia i znaleźć rozwiązanie globalne. Algorytm jest uniwersalny, ponieważ, nie jest powiązany ze szczególnymi właściwościami danego modelu – jest od niego silne niezależny.
Z powodu tej niezależności *jadro algorytmu* nie jest powiązane z danym algorytmem i może być stosunkowo łatwo wykorzystane do rozwiązania innego zadania. Ma to szczególne znaczenie

wtedy, kiedy korzystamy z kodu, przy którego pisaniu nastawiono się na optymalizacje – dużo trudniejsze staje się zrozumienie treści programu, za to korzyści z używania szybkiego algorytmy są dużo większe

Cecha programów symulowanego wyżarzania jest zależność miedzy jakością rozwiązania o czasem liczenia. Ma to analogie w fizycznym procesie – im wolniej obniżamy temperaturę tym mniej zaburzeń w strukturze krystalicznej materiału powstaje. Można to uznać również za zaletę – w zależności od oczekiwanej jakości rozwiązania (bliskości minimum znalezionego od globalnego) można regulować szybkość obniżania temperatury.

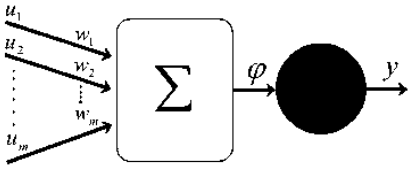
Czytając o symulowanym wyżarzaniu, może się narzucać myśl, *czemu to zostało wymyślone tak późno?*. Od wieków kowale stosowali metodę powolnego lub szybkiego schładzania żelaza w zależności o oczekiwanych własności. Trudności było w tym przypadku było początkowo, zauważenie i opisanie dokładnych zależności pomiędzy spadkiem temperatury a strukturą sieci krystalicznej materiału i związanej z tym minimalnej energii wewnętrznej. Kolejną przeszkodą był sposób dekrementacji temperatury na maszynach cyfrowych i określenie sposobu wydostania się z lokalnego minimum połączonego ze szybkością zbiegania do akceptowalnego rozwiązania globalnego

3. 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.

Klasyczny komputer w porównaniu z mózgiem jest szybki, wydajny, i skuteczny, tym nie mniej są zadania, w których mózg góruje nad komputerem bezdyskusyjnie. 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, stanowi dużą trudność. Podobnie jest z rozpoznawaniem dźwięku, jak choćby ludzkiego głosu.

3.1. Zasada działania neuronu



Na wejście przetwornika podawane są sygnały wejściowe, które następnie są mnożone przez odpowiednie współczynniki wag (odpowiadające "sile" połączeń synaptycznych w biologicznym neuronie). "Ważone" sygnały wejściowe są następnie sumowane i na tej podstawie wyznacza się aktywność neuronu. Oto schemat sztucznego neuronu:

W pewnym przybliżeniu blok sumowania odpowiada biologicznemu ciału komórki, w której realizowane jest algebraiczne sumowanie ważonych sygnałów wejściowych, oraz generowany jest sygnał wyjściowy j, który może być traktowany jako potencjał membranowy komórki.

Potencjał membranowy możemy wyliczyć ze wzoru:

$$\varphi = \sum_{i=1}^{m} w_i u_i = w^T u$$

gdzie w jest wektorem współczynników wag, u - wektorem sygnałów wejściowych, m - liczbą wejść neuronu.

W roku 1958 Rosenblatt opracował i zbudował sztuczną sieć neuronową nazwaną perceptronem. Ten częściowo elektromechaniczny, a częściowo elektroniczny układ przeznaczony był do rozpoznawania znaków alfanumerycznych z procesem uczenia jako metodą programowania systemu. Ważnym rezultatem Rosenblatta było ponadto udowodnienie tzw. twierdzenia o zbieżności perceptronu, które gwarantuje skończoną liczbę iteracji procesu uczenia, o ile dla zagadnienia modelowanego przy użyciu tego typu sieci optymalny układ wag istnieje. Pomimo iż działanie perceptronu nie było zadowalające z punktu widzenia zasadniczego celu (układ wykazywał dużą wrażliwość na zmianę skali rozpoznawanych obiektów oraz ich położenia w polu widzenia), był to ogromny sukces badań prowadzonych w tym zakresie. Przede wszystkim był to pierwszy fizycznie skonstruowany układ symulujący sieć nerwową, który wraz ze zdolnością do uczenia się wykazywał zdolność do poprawnego działania nawet po uszkodzeniu części jego elementów.
Idea perceptronu jest zawarta w następujących zasadach:

1. Elementem składowym perceptronu jest sztuczny neuron, którego model matematyczny może być opisany funkcją aktywacji unipolarną:
   $$y = \begin{cases} 1, & \varphi > 0 \\ 0, & \varphi \le 0 \end{cases} \quad \text{gdzie:} \quad \varphi = \sum_{i=1}^{m} w_i u_i - \theta$$
   przy czym w_i oznacza wagę *i*-tego połączenia wstępującego do elementu; u_i - wartość *i*-tego wejścia; q - wartość progową funkcji aktywacji.
2. Sieć perceptronową można podzielić jednoznacznie na ściśle uporządkowane i rozłączne klasy elementów zwane warstwami, wśród których wyróżnić można warstwę wejściową i wyjściową. Pozostałe noszą nazwę warstw ukrytych.
3. Perceptron nie zawiera połączeń pomiędzy elementami należącymi do tej samej warstwy.
4. Połączenia pomiędzy warstwami są asymetryczne i skierowane zgodnie z ich uporządkowaniem, tzn. od warstwy wejściowej do pierwszej warstwy ukrytej, następnie od pierwszej do drugiej warstwy ukrytej, itd. aż do warstwy wyjściowej. Nie ma połączeń zwrotnych.

3.2. Algorytm uczenia

1. Pokaż obraz wejściowy **u** i wyznacz wyjście *y*.
2. Podejmij jedną z poniższych decyzji:
   - jeśli obraz wyjścia jest poprawny, to wróć do punktu 1 i pokaż inny obraz wejściowy **u**;
   - jeśli sygnał wyjścia jest niepoprawny i równy 0, to dodaj wartość każdego wejścia u^i pomnożoną przez pewną liczbę h do wartości odpowiedniego współczynnika wag, w_i;
   - jeśli sygnał wyjścia jest niepoprawny i równy 1, to odejmij wartość każdego wejścia u^i pomnożoną przez pewną liczbę h od wartości odpowiedniego współczynnika wag, w_i.
3. Wróć do punktu pierwszego i pokaż inny obraz wejściowy **u**.

3.3. Samodzielne uczenie – już AI czy jeszcze nie?

Uczenie sieci może przebiegać zarówno *z nauczycielem*, jak i *bez nauczyciela*.

- W przypadku *uczenia z nauczycielem* znana jest odpowiedź prawidłowa na zadane sieci pytanie, możemy ją więc wykorzystać do skorygowania błędnej decyzji sieci. Przykładem tego typu problemu jest uczenie sieci klasyfikacji danych np. określania czy klient banku o podanej charakterystyce spłaci kredyt.
  Podstawowa zasada tej metody uczenia polega na tym, iż dla zadanych danych wejściowych, znana jest pożądana odpowiedź sieci. Wykonanie porównania z odpowiedzią udzieloną przez sieć pozwala ustalić błąd. Jego wielkość może zostać wykorzystana do korekty działania sieci. Nazwa metody oddaje jej charakter - kluczowym założeniem jest możliwość skorzystania z gotowych, poprawnych odpowiedzi - a więc wiedzy "nauczyciela".
- Ciekawym rodzajem uczenia jest *uczenie bez nauczyciela*. W takim przypadku nie znamy prawidłowej odpowiedzi, a zadaniem sieci jest jej ustalenie. Przykładem może być ustalanie granic pomiędzy różnymi klasami obrazów. W tym przypadku sieć musi posiadać mechanizm autoadaptacyjny.
  Taka metoda uczenia jest również nieobca człowiekowi. Każda próba samodzielnej analizy nowych zjawisk i ustalenia reguł nimi rządzących to uczenie bez nauczyciela - czyli bez znanych rozwiązań.

3.4. Sieci neuronowe krokiem ku sztucznej inteligencji

Otóż formalnie sztuczne sieci neuronowe nie wymagają programowania! Wystarczy stworzyć sieć, a ona – o ile jest dobrze zaprojektowana uczy się sama. Rola programisty ogranicza się do zaprojektowania takiej struktury sieci, która najlepiej będzie nadawała się do rozwiązania danego problemu, a następnie do umiejętnego pokierowania procesem uczenia sieci. Od razu narzuca się analogia do procesów poznawania świata przez człowieka. Aby być w jakiejś dziedzinie fachowcem, trzeba po pierwsze mieć wrodzone uzdolnienia w danej dziedzinie (odpowiednią strukturę własnej sieci neuronowej, jaką jest mózg), a po drugie - mieć dobrego nauczyciela.

W zwykłym programie błąd programisty może doprowadzić do niestabilności całego systemu. Sieć neuronowa natomiast nawet w przypadku poważnego uszkodzenia powinna działać dalej (do pewnego stopnia). A nalogia: nasz mózg działa nadal po wypadku, w którym odniesie niewielki uszczerbek, działa pomimo niszczenia neuronów przez alkohol lub proces starzenia, działa długo bez "awarii", choć jego struktura wciąż jest naruszana. Dopiero po przekroczeniu pewnego progu uszkodzeń mózg odmawia posłuszeństwa - choroba Alzheimera, dalsze stopnie choroby alkoholowej itd.

Sztuczne sieci neuronowe mają także zdolność do uogólniania zdobytej wiedzy. Znaczy to dokładnie tyle, że jeśli sieć nauczy się, rozpoznawać kolory: czerwony i żółty, to rozpozna również różowy i bladożółty, czyli kolory podobne do znanych.

Niedoskonałością sieci neuronowych jest niemożność wykorzystania ich tam gdzie potrzebne są precyzyjne wyniki - obsług kont bankowych, itp. Wynika to z faktu, iż sztuczne sieci neuronowe są odbiciem ludzkiego mózgu, ten zaś nie jest przystosowany do precyzyjnego operowania liczbami. Kiedy opisujemy kogoś, nie mówimy, że ma 188.34 cm wzrostu, tylko że jest wysoki. I to nie tyle dlatego, że dokładność tej informacji jest wystarczająca, lecz dlatego, że bez

odpowiedniej aparatury pomiarowej nasz mózg nie jest w stanie precyzyjnie ocenić pewnych wielkości. Sztuczne sieci neuronowe operują tzw. pojęciami rozmytymi. Nierzadko zdarza się że sieć neuronowa na pytanie nie udziela precyzyjnej odpowiedzi tylko „raczej tak" – jest to kolejna analogia do modelu sztucznej inteligencji. Nie jest to spowodowane błędnym zaprojektowaniem sieci, lecz wynika ze sposobu przechowywania danych w strukturach sieciowych.

3.5. Czego jeszcze brakuje do osiągnięcia AI

Istotną niedoskonałością sieci neuronowych, jeśli porównywać je do sztucznej inteligencji jest brak możliwości zastosowania wieloetapowego rozumowania. Struktury sieci nie są w stanie korzystać z rozwiązań pośrednich. Jest to odmienne niż działanie ludzkiego mózgu, któru na każdym etapie rozumowania korzysta z wyników poprzednich rozumowań, dochodząc w ten sposób do finalnego wniosku.
Jeżeli chcielibyśmy otrzymać taką strukturę, należałoby stworzyć nie pojedynczą sieć ale zespół sieć, które najczęściej nie byłyby połączone liniowo. Głównym problemem jest stworzenie takiego zestawu sieci, które odpowiadałyby różnorodności funkcji ludzkiego mózgu. Ograniczenia wynikają też z przyjęcia upraszczającego założenia, że korzystamy z sieci jednokierunkowych – informacja jest przekazywana z wejścia na wyjście i informacje o rozwiązaniach w kolejnych fazach nie mogą wpływać na rozwiązania we fazach wcześniejszych. Gdyby taki mechanizm był możliwy, sieci neuronowe potrafiłyby modyfikować na zasadzie sprzężenia zwrotnego wcześniejsze etapy, tak aby w końcu dojść do optymalnego rozwiązania.

3.6. Zastosowania SN

Ze względu na opisane powyżej specyficzne cechy i niepodważalne zalety, obszar zastosowań sieci neuronowych jest rozległy:
Rozpoznawanie wzorców (znaków, liter, kształtów, sygnałów mowy, sygnałów sonarowych), Klasyfikowanie obiektów, Prognozowanie i ocena ryzyka ekonomicznego, Prognozowanie zmian cen rynkowych (giełdy, waluty), Ocena zdolności kredytowej, Ocena wniosków ubezpieczeniowych, Rozpoznawanie wzorów podpisów, Prognozowanie zapotrzebowania na energię elektryczną i wiele innych

4. Algorytmy ewolucyjne

Algorytmy ewolucyjne są to procedury losowego przeszukiwania inspirowane przez mechanikę genetyczną oraz procesy naturalnej selekcji. Początki historii obliczeń ewolucyjnych sięgają lat pięćdziesiątych XX wieku, kiedy to paru naukowców studiowało systemy ewolucyjne z myślą wykorzystania ewolucji jako narzędzia optymalizacji w problemach inżynierskich. Przewodnią ideą we wszystkich tych systemach było generowanie rozwiązań-kandydatów dla danego problemu przy użyciu operatorów wykorzystywanych naturalnie przez zmiany genetyczne i procesy selekcji, a jej pomysłodawcą był John Koza. Jedna z pierwszych prac na ten temat jest to pochodząca z 1965 roku analiza Rechenberga, w której przedstawia on użycie strategii ewolucyjnych, jako metod do optymalizacji parametrów urządzeń, które korzystają z prawa Bernoulliego dla przepływów nielaminarnych, takich jak skrzydła samolotów czy kadłuby łodzi.

Idea ta była później rozmijana m.in. przez Schwefela (1975, 1977). Współcześnie algorytmy ewolucyjne odgrywają znaczącą rolę w zagadnieniach optymalizacji aplikacji odkrywania wiedzy („Data mining").

4.1. Rodzaje algorytmów

Główne rodzaje algorytmów ewolucyjnych:

- Algorytmy genetyczne (genetic algorithms – GAs)
- Programowanie genetyczne (genetic programming – GP)
- Strategie ewolucyjne (evolution strategies – ES)
- Programowanie ewolucyjne (evolutionary programming – EP)

Wszystkie algorytmy ewolucyjne dzielą te same podstawowe koncepty, ale różnicą się w sposobie kodowania rozwiązań i w rodzajach operatorów używanych do tworzenia kolejnych generacji.

Algorytmy ewolucyjne kontrolowane są za pomocą kilku parametrów takich, jak wielkość populacji, stopień kontroli częstości występowania mutacji i krzyżówek. Generalnie, jak wszystkie algorytmy heurystyczne nie ma gwarancji, że znalezione rozwiązanie będzie rozwiązaniem optymalnym, ale ostrożna manipulacja parametrami algorytmu oraz wybór reprezentacji adekwatnej do problemu zwiększa szanse sukcesu. Istnieje zarówno wiele sposobów kodowania potencjalnych rozwiązań jako chromosomów, jak i możliwości wyboru metod krzyżowania i mutacji. Algorytmy genetyczne tradycyjnie używają chromosomów złożonych z zer i jedynek, ale inne sposoby kodowania mogą być bardziej naturalne dla danego problemu. W programowaniu genetycznym koduje się rozwiązania jako programy komputerowe, z kolei ES i EP używają liczb zmiennoprzecinkowych, które mogą być przydatne w optymalizacji funkcji, w której parametry są liczbami rzeczywistymi, ale za to takie podejście może być problematyczne w rozwiązywaniu problemu komiwojażera. Wybór kodowania związany jest z operatorem wykorzystywanym do generowania nowych rozwiązań. Operatory szerzej będą opisane w części poświęconej algorytmom genetycznym.

4.2. Zastosowania

Algorytmy ewolucyjne mają szerokie zastosowanie, są przydatne m.in. w:

- Systemach odkrywania wiedzy
- Maszynowym uczeniu (w połączeniu z sieciami neuronowymi, drzewami decyzyjnymi oraz tzw. „rule-based systems")
- Programowaniu równoległym („cluster computing")

Potencjał algorytmów ewolucyjnych został dostrzeżony przez współczesnych badaczy, czego dowodem jest szereg organizowanych konferencji (np. konferencja EvoIASP organizowana przez Working Group on Evolutionary Algorithms in Image Analysis and Signal Processing (2001), Knowledge Discovery and Data Mining (KDD), International Conference on Machine Learning (ICML), and the Genetic and Evolutionary Computation Conference (GECCO)), wydawanych książek i biuletynów (np. *Evolutionary Computation, Genetic Programming and Evolvable Machines, IEEE Transactions on Systems, Man, and Cybernetics*, and the *IEEE Transactions on Evolutionary Computation*) traktujących o tych technikach.

4.3. Złożoność obliczeniowa

Niewątpliwie algorytmy ewolucyjne mogą być bardzo przydatne w wszelkich problemach optymalizacji, ale niestety zawsze musza istnieć pewne wady. Głównym czynnikiem deprecjonującym algorytmy ewolucyjne jest, jak wskazują niektórzy autorzy, czas konsumowany podczas obliczeń. Dla przykładu Poli (1996) komentuje, że olbrzymie wymagania obliczeniowe użycia programowania genetycznego w przetwarzaniu obrazów przeszkodziły naukowcom w badaniach nad zachowaniem tych algorytmów w rozwiązywaniu rzeczywistych problemów. Także Ebner i Zelż (1999) wyrażają podobny pogląd, gdyż zaobserwowali, że ewolucja operatora przetwarzania obrazów zajmuje na pojedynczym pececie kilka dni, co czyni bardzo trudnym wykorzystanie tego algorytmu w adapcyjnych systemach o zmiennych warunkach środowiska. Podjęto kilka prób ominięcia problemu nadmiernej złożoności obliczeń algorytmów ewolucyjnych. W przetwarzaniu obrazów zaproponowano (m.in. Poli,1996), aby zamiast używać wszystkich pikseli do ewaluacji operatora, można by wykorzystywać tylko pewną próbkę. Inne podejście (Bhanu, Lee, Ming, 1995) zakłada utrzymywanie „globalnej populacji zdrowych osobników", która może być zainicjalizowania algorytmu dla każdego obrazu, co czyni algorytm adopcyjnym, jak i skraca czas obliczeń. Proponuje się także naturalne zrównoleglenie algorytmów ewolucyjnych, tak, aby zredukować czas detekcji krawędzi podczas analizy obrazu. Z podobnymi problemami związanymi ze znaczną ekstensywnością obliczeń spotkano się także w innych zastosowaniach algorytmów ewolucyjnych, ale i tu analogiczne rozwiązania okazują się być pomocne. Jak już wcześniej wspomniano również wybór odpowiedniej reprezentacji problemu skutkuje projektowaniem algorytmów o lepszych właściwościach, tzn. mogących rozwiązywać większe problemy. Czasami bardziej wydajne rozwiązania otrzymuje się łącząc algorytmy ewolucyjne z innymi technikami heurystycznymi.

5. Algorytmy genetyczne

W ostatnich latach wzrosło zainteresowanie problematyką dotyczącą systemów, w których do rozwiązywania zadań stosuje się algorytmy ewolucyjne. Algorytmy te wykorzystywane są do przeszukiwania przestrzeni rozwiązań alternatywnych.

Algorytmy genetyczne swoje istnienie zawdzięczają obserwacji i próbie naśladowania naturalnych procesów zachodzących w świecie organizmów żywych, a dokładniej procesowi ewolucji, który zawiera naturalną selekcję osobników żywych występujących w naturalnym środowisku. W wyniku inspiracji światem naturalnym poszczególne składowe problemów rozwiązywanych z wykorzystaniem algorytmów genetycznych opisywane są przy użyciu terminologii, którą posługuje się genetyka i ewolucja. Jest to kolejny przykład, gdzie przeświadczenie o doskonałości natury wykorzystywany jest w informatyce.

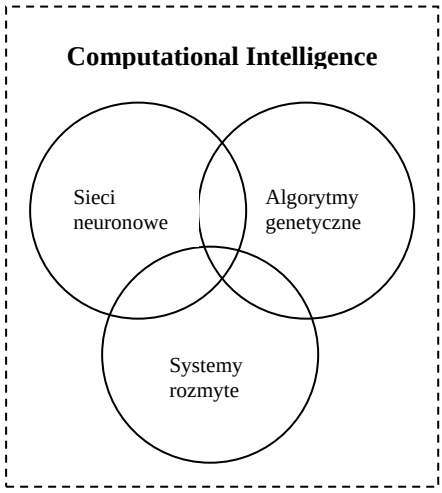
Ideę algorytmów genetycznych przedstawił Holland na przełomie lat sześćdziesiątych i siedemdziesiątych. Zainteresowały go cechy naturalnej ewolucji, w szczególności fakt, że ewolucja zachodzi na chromosomach, a nie na żywych istotach. Holland wierzył, że odpowiednio wprowadzony do komputera algorytm może dostarczyć techniki rozwiązywania trudnych problemów w sposób, w jaki czyni to natura – poprzez ewolucję.

Algorytmy ewolucyjne przetwarzają populację *osobników*, którzy reprezentują pewne rozwiązanie zadanego problemu. Populacja osobników działa w *środowisku*, które można zdefiniować na

podstawie rozwiązywanego problemu. Każdemu z osobników przyporządkowana jest wartość liczbowa określająca reprezentowane przez niego rozwiązanie, które odpowiada stopniowi jego *przystosowania* do otoczenia. Osobnik wyposażony jest w informację stanowiącą jego *genotyp*, na podstawie, którego można utworzyć *fenotyp* będący zestawem cech określanych przez genotyp, podlegających ocenie środowiska, wartość liczbowa tej oceny nazywa się przystosowaniem osobnika. Opisując środowisko działania osobników można posłużyć się *funkcją przystosowania*, która służy do wyznaczenia przystosowania na podstawie fenotypu osobnika. Genotyp osobnika składa się z *chromosomów* zbudowanych z jednostek elementarnych, zwanych *genami*.

5.1. Algorytm

Sposób działania algorytmu genetycznego można sprowadzić do pętli, w której wykonywane są po sobie: reprodukcja, operacje genetyczne, ocena i sukcesja.



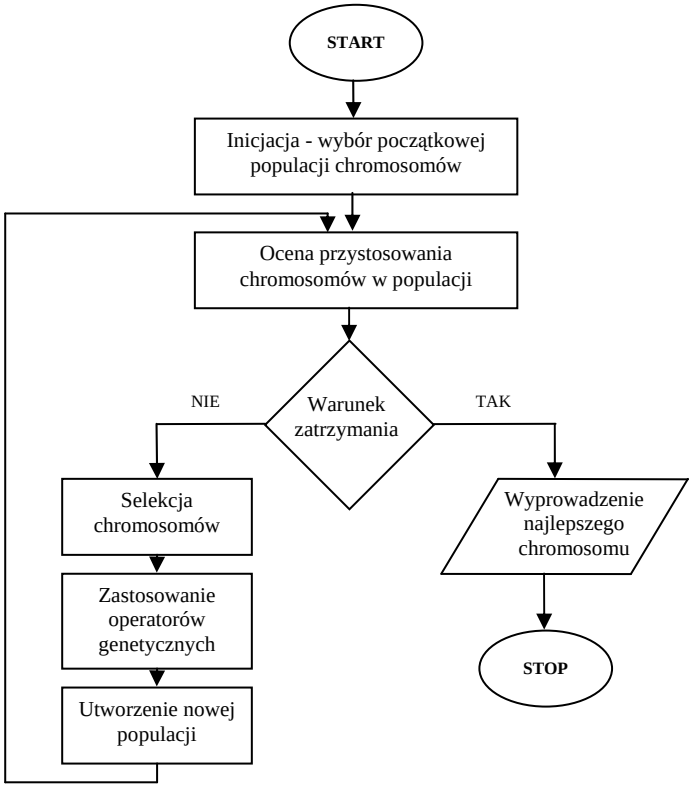
Wzajemne relacje między sieciami neuronowymi, algorytmami genetycznymi i systemami rozmytymi

Cechy wyróżniające algorytmy genetyczne:

- nie przetwarzają bezpośrednich parametrów zadania, lecz ich zakodowaną postać;
- prowadzą przeszukiwanie, wychodząc nie z pojedynczego punktu, lecz z pewnej ich populacji;
- korzystają tylko z funkcji celu, nie zaś z jej pochodnych lub innych pomocniczych informacji;
- stosują probabilistyczne, a nie deterministyczne reguły wyboru.

Schemat ogólnego algorytmu genetycznego

```
let G denote a generation, P a population of size M, and xl the l^th
chromosome in P
initialize the initial population P_G=0 = {x^1_G=0, … , x^M_G=0}
evaluate every xl Î PG=0, l = 1, …, M
k=1
while the stopping criteria is not satisfied do
      select P' (an intermediate population) from P_G=k-1
      P_G=k - crossover elements in P'
      mutate elements in P_G=k
      evaluate every x^l from P_G=0, l = 1, …, M
      k = k+1
end while
return the best encountered solution
```



Elementy algorytmu genetycznego:

- Populacja początkowe - Przy jej tworzeniu pożądane jest zachowanie różnorodności osobników, co sprzyja szybkiej zbieżności algorytmu do optymalnego rozwiązania – eksploracja możliwie dużego obszaru rozwiązań bez zwiększania liczebności populacji
- Ocena dostosowania – Najczęściej stosuje się porównanie bitowe chromosomów poszczególnych osobników
- Operatory genetyczne :mutacja, selekcja, krzyżowanie – opisane w dalszej części

5.2. Operatory genetyczne

- **Selekcja** – Zadaniem selekcji jest eliminowanie słabszych osobników, pozostawiając w dalszej *grze* tylko osobniki lepiej przystosowane. Istnieje wiele algorytmów selekcji – najprostszym z nich jest selekcja proporcjonalna, która polega na wyborze proporcjonalnym z funkcją wagową będąca miarą przystosowania osobnika:.

$$p(ch_i) = \frac{F(ch_i)}{\sum_{i=1}^{n} F(ch_i)}$$

  W czasie selekcji należy zwrócić szczególna uwagę na niebezpieczeństwo ujednolicenia populacji. Rodzaje metod selekcji można podzielić na dwa rodzaje ; deterministyczne i probabilistyczne. W pierwszej grupie znajdują się algorytmy, które nie używają żadnego czynnika losowego – wybierają zawsze najlepiej przystosowane osobniki –wzrasta wtedy zagrożenie ujednolicenia – zachowanie niekorzystnych cech w populacji
- **Krzyżowanie** – $\kappa : G \times G \to G \times G$ Idea krzyżowania zaczerpnięta jest z wzoru rozmnażania płciowego, gdzie osobnik potomek ma cechy obu osobników. Można stosować krzyżowanie jednopunktowe i wielopunktowe – kopiowania ciągów bitów od obu

osobników. W przypadku krzyżowania jednopunktowego pierwsza część łańcucha identyfikującego genotyp osobnika jest kopiowana od pierwszego rodzica a druga od drugiego. W przypadki krzyżowania wielopunktowego takich podła cuchów jest więcej, przy czym sumaryczna długość łańcucha osobnika potomnego jest tak jak u rodziców

- **Mutacja** - $\mu : G \to G$ Polega na losowej zmianie bitów w chromosomie pojedynczego osobnika. Przy zastosowaniu danego rozkładu prawdopodobieństwa jest wymieniany jeden (lub więcej) bitów na przeciwny. Po mutacji musi być zachowana zgodność nowego rozwiązania z ograniczeniami zadania

Operatory te są na tyle uniwersalne, że znalazły zastosowanie nie tylko w algorytmach genetycznych, ale także podobnych pod względem koncepcji algorytmach ewolucyjnych. Znowu pojawia się motyw uniwersalnych zastosowań mechanizmów używanych (i sprawdzonych) w naturze.

5.3. Algorytmy genetyczne a symulowane wyżarzanie

Opierając się na badaniach Manikas i Cain'a[16]. Porównywali oni szybkość zbieżności do optymalnego rozwiązania problemu podziału grafu (*Circuit Partitioning Problem*) dla algorytmu symulowanego wyżarzania i genetycznego. Badanymi parametrami były czas rozwiązania problemu, zajętość pamięci oraz jakość rozwiązania – odległość od rozwiązania optymalnego. Z tej rywalizacji w prawie wszystkich kryteriach zwycięsko wyszły algorytmy genetyczne. Jest to spowodowane tym, że algorytmy genetyczne przeglądają większe spektrum możliwych rozwiązań dzięki zastosowaniu różnorodnej populacji. Takie zróżnicowanie powoduje jednak większą ilość potrzebnej pamięci – tutaj lepiej wypada wyżarzanie. Inną zaletą algorytmów genetycznych jest możliwość zaimplementowania obliczeń na maszynach równoległych.

5.4. Algorytmy ewolucyjne jako krok w ewolucji maszyn cyfrowych ku sztucznej inteligencji

Algorytmy genetyczne są kolejnym przykładem wzorowania się na naturze. Porównując algorytmy genetyczne do symulowanego wyżarzania lub sieci neuronowych widzimy istotny problem poznania dokładnych zasad rządzących ewolucją. W naturalnych warunkach ewolucja przebiega bardzo wolno, a zmiany są mało widoczne. Chcąc zaimplementować takie struktury musimy pokonać barierę czasu – spowodować drastyczną zmianę genotypu całej populacji w czasie rzędu sekund lub minut. Trudnością nie są tutaj ograniczenia prędkości procesorów czy dostępu do pamięci, ale brak dokładniej wiedzy na temat ewolucji żywych organizmów – możemy tylko opierać się na przesłankach z przeszłości. Niemożliwa jest obserwacja tego procesu.

Na pierwszy rzut oka algorytmy genetyczne to nic innego jak przypadkowe rozwiązanie – mutacje – losowa zamiana bitów, krzyżowanie – losowe łączenie osobników. Jednak *w tym szaleństwie jest metoda*. Kluczowym elementem jest funkcja przystosowania – to ona determinuje, jakie rozwiązanie jest wystarczająco dobre by przetrwać. Nie można zapominać, że w tej funkcji też jest ukryty element losowy – słabsze osobniki też mają szansę przetrwać, aby zapewnić różnorodność populacji. W całym tym niedeterministycznym charakterze zjawiska jest ukryta jego siła. Podobne zjawisko można zaobserwować analizując ludzki mózg. Zasady jego pracy są rozmyte – dotychczas nie udało się dogłębnie przeanalizować pracy neuronów, które odpowiadają za proces myślowy.

Duża różnicę w pracy mózgu, a efektem pracy algorytmów genetycznych jest problem optymalnego rozwiązania – człowiek niejednokrotnie potrafi dowieść, że jest to najlepsze możliwe rozwiązanie. Maszyna potrafi tylko dać odpowiedź, jakie jest najlepsze dotychczasowe rozwiązanie (chociaż często potrafi znaleźć je dużo szybciej niż człowiek). Przykładem może być tutaj problem złożoności algorytmu sortowania opierającego się na porównaniach pomiędzy elementami – dowiedziono, że złożoność nie może być mniejsza niż *n log n.*

Ewidentną zaletą algorytmów genetycznych jest efekt. Widząc jak zmienił się człowiek i jakie bariery zdołał przełamać w ciągu tysiącleci ewoluując od człowieka prehistorycznego do *homo sapiens*, chyba większość czuje respekt dla takiej potęgi. Przeszkoda na drodze do maszyny myślącej (sztucznej inteligencji) jest paradoksalni – efekt niedokończonego procesu ewolucji – człowiek. Jego niedoskonałość nie pozwala na skopiowanie reguł rządzących w naturze.

6. Systemy immunologiczne

Natura od zawsze inspirowała człowieka. Wiele pomysłów powstało w wyniku obserwacji i naśladowania mechanizmów rządzących w naturze. Systemy immunologiczne – nowa dziedzina licząca niewiele ponad 10 lat, to kolejny przykład skutecznego wykorzystania metafor biologicznych w rozwiązaniu złożonych problemów. Interesują się nią nie tylko naukowcy, ale także firmy komercyjne jak Symantec czy IBM, Sun, co pokazuje jak praktyczne są algorytmy wzorowane na mechanizmach sprawdzających się w przyrodzie. Prace prowadzone w laboratoriach badawczych owych firm zmierzają w kierunku uniwersalnych systemów antywirusowych, systemów drążenia danych, czy jak w przypadku *Starlab* – systemów uczących się

Immunologia to nauka z pogranicza biologii i medycyny. Przedmiotem jej dociekać jest badanie odporności organizmów na zarazki, toksyny i niektóre substancje chemiczne. Za ojców immunologii uważa się Ludwika Pasteura i Roberta Kocha, którzy odkrywając złożony świat chorobotwórczych organizmów proponowali jednocześni środki zaradcze wspomagające zdolności obronne organizmu.

Od niedawna immunologia to nie tylko przedmiot zainteresowań lekarzy czy biologów. Chęć zrozumienia mikroskopowych własności układu odpornościowego – mechanizmów, jakie rządzą tym systemem – stymuluje badania w kierunku immunologii matematycznej. Chociaż immunologia to nauka, która liczy sobie blisko 100 lat to nasze widza na temat zasad działania ludzkiego systemu immunologicznego jest wciąż niezadowalająca.

Z tego, co wiemy, a raczej w większości przypuszczamy, z naturalnych systemów immunologicznych wyłania się niezwykle atrakcyjna perspektywa dla informatyków. Najważniejsze cechy, które mogą mieć zastosowanie przy tworzeniu algorytmów to:

- Rozproszona detekcja – nie ma scentralizowanego systemu zarządzania, każdy element wie, co ma robić. Z tego powodu przypomina system z rozproszonymi agentami. Stanowi o możliwości wykonywania równoległych obliczeń
- Progowa detekcja – proces wiązania antygenu i przeciw ciała opiera się na częściowym podobieństwie obu elementów. Mechanizm ten zwiększa ilość rozpoznawanych molekuł
- Detekcja anomalii – układ prawidłowo identyfikuje patogeny (ciała obce), z którymi zetknął się pierwszy raz
- Adaptacyjność – układ potrafi przystosować się do nowych warunków i wytworzyć nowe rozwiązania, które zdołają przeciwstawić się nowym, bardziej złożonym problemom

- Prostota reprezentacji – wewnętrzny obraz patogenów, z którymi zetknął się wcześniej system

6.1. Istota algorytmu

W uproszczonym sensie układ immunologiczny to klasyfikator binarny – musi on umieć rozpoznać obiekty „swój" – „obcy". Traktując obcą strukturę jako problem, który należy rozwiązać, a uruchomienia mechanizmu obronnego jako –rozpoczęcie przetwarzania algorytmu bardzo łatwo można zauważyć przydatność struktur immunologicznych w informatyce

Podstawowym mechanizmem utrzymania populacji limfocytów (struktury zawierające sposoby rozwiązania wcześniej napotkanego problemu) jest ich zmiana koncentracji poszczególnych rodzajów klonów. Równanie opisujące ten proces to:

$$\begin{array}{ccccc} \textit{stopień} & & \textit{dopłop} & \textit{śmierć} & \textit{reprodukcja} \\ \textit{różóżnorodści} = & \textit{nowych} & - \textit{nieefektywnych} + & \textit{pobudzonych} \\ \textit{populacji} & \textit{komórek} & \textit{komórek} & \textit{komórek} \end{array}$$

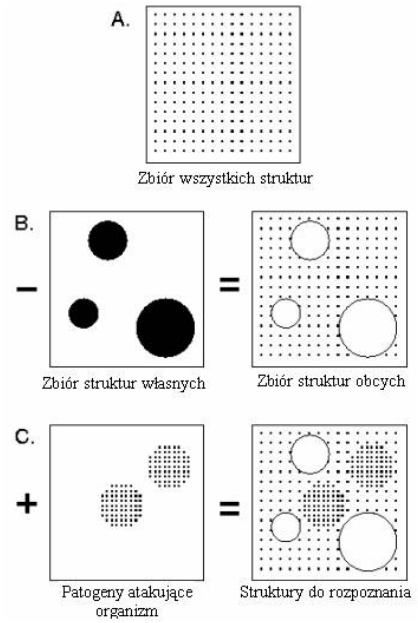
Istotą „immunologiczne" inżynierii jest wykorzystanie informacji zawartej w stawianym problemie w celu rozwiązania go. Generując przeciwciała dopasowane do antygenów będących zbiorem danych uczących się, system jest w stanie wypracować efektywną technikę rozwiązania problemu.

6.2. Dopasowanie

B-limfocyty są monoklonalnymi komórkami, na powierzchni, których znajduje się około 10^5 receptorów (przeciwciał) reagujących na substancje zagrażające funkcjonowaniu organizmu nazywane antygenami. Ze względu na to, że wszystkie receptory rozpoznają jeden typ antygenu (ściślej – niewielką klasę strukturalnie podobnych antygenów), w sztucznych systemach immunologicznych utożsamia się B-limfocyt z jego przeciwciałami. W przypadku zadań optymalizacji rolę antygenu pełni nieznane optimum, a przeciwciałem jest jego lokalizacja, czyli rozwiązanie tegoż zadania.

Mechanizm detekcji antygenów

Kwadrat A reprezentuje zbiór potencjalnych struktur, z którymi układ może się zetknąć. Z tego zbioru należy usunąć struktury odpowiadające komórkom własnym. Zaznaczono je na czarno w lewym kwadracie części B. Struktury, które musi rozpoznawać układ odpornościowy, przedstawiono w lewym kwadracie części B. Jeżeli organizm został zaatakowany przez konkretne patogeny, wskazane w lewym kwadracie części C, ich struktury zostają zapamiętane, co wskazuje prawy kwadrat części C.



W modelach systemów wykorzystywanych w informatyce najczęściej przyjmuję się upraszające założenie, że nie ma

komórek limfocytowych typu T a wszystkie informacje koncentrują się B-komórkach. Jako reprezentację epitopów i paratopów (struktury odpowiadające za dopasowanie do znanych rozwiązań) reprezentowane są jako ciągi binarne, co później ułatwia ocenienie stopnia dopasowania. Jest on liczony ze wzoru:

$$m_{ij} = \sum_k G\left(\sum_n \left| e_i(n+k) - p_j(n) \right| - s + 1 \right)$$

gdzie *s* to próg dopasowania, G to funkcja określające dopasowani aprzy danym ustawieniu paratopu i epitopu

	1	0	1	0	**1**	0	0	1	**1**	1	**0**	epitop
paratop					**1**	1	1	0	**1**	0	**1**	0	1	0	

Aby upewnić się, czy „limfocyty B" nie działają przeciwko operacjom wykonywanym prawidłowo, są w pierwszej kolejności porównywane przez „limfocyty T" z wyuczonymi przez siebie wzorcami właściwego funkcjonowania. Tak jak w żywych organizmach, nieprawidłowe „limfocyty" zmuszane są do samozniszczenia, zanim wywołają kolejne szkody. Gdy błąd zostaje zlokalizowany, zadanie przejmują specjalne procedury, których celem jest wykonanie takich działań, aby skorygować błąd

Niektóre używane obecnie systemy wykrywania błędów, np. te stosowane na statkach kosmicznych, zależą od tzw. procedury głosowania, podczas której kilka systemów oblicza to samo. Jeśli wszystko działa prawidłowo, ich wyniki powinny się zgadzać. Jeśli jednak kilka z nich zawiedzie, rezultat będzie zły. „Układ immunologiczny" jest w takim przypadku bardziej skuteczny.

6.3. Globalne rozwiązanie

W celu globalnego spojrzenia na procesy powstawania i eliminacji przeciwciał często zakłada się, że system jest dostatecznie duży, w tym sensie, że liczba iteracji koniecznych do przeprowadzania znaczącej zmiany w koncentracji jest dostatecznie duża. Przy taki założeniu można sformułować równanie dynamiki układu. Jak widać ta dynamika uwzględnia zmiany populacji przeciwciał zarówno w sensie ilościowym(parametr N), jak i jakościowy (odpowiedz na dany antygen)

$$\ddot{x}_i = c\left(\sum_{j=1}^{N} m_{ji} x_i x_j - k_1 \sum_{j=1}^{N} m_{ji} x_i x_j + \sum_{j=1}^{M} m_{ji} x_i x_j \right) - k_2 x_i$$

gdzie:
- i = 1, … ,N
- N to liczba różnych przeciwciał, M liczbę różnych antygenów,
- x_i oznacza koncentracje przeciwciał typy i, y_j – koncentracje antygenów typu j.
- c – stała lub parametr określający liczbę produkowanych limfocytów w reakcji odpowiedzi na dany antygen

Model pamięci immunologicznej to złożony system, którego zadaniem jest wykorzystanie wcześniej nabytej wiedzy w celu rozpoznawania i rozwiązania nowo napotkanego problemu. Z pamięcią immunologiczna związane jest „zapominanie immunologiczne" – z powodu ogromnych rozmiarów wiedzy, jaki musiałby przechowywać system istnieje proces usuwania przeciwciał, które zwalczają antygeny, z którymi przez dłuższy czas system nie miał kontaktu.

W systemie immunologicznym nie ma wyodrębnionego wyjścia. Celem systemu jest bowiem usuwanie łańcuchów. Problem braku wyraźnie wyodrębnionych sygnałów wyjściowych rozwiązania nieco sztucznie przyjmując, że rezultatem odpowiedzi immunologicznej jest klon o najwyższej skuteczności usuwania antygenów. Klon ten nazwano *zwycięzcą* odpowiedzi immunologicznej.
W bardziej rozwiniętych modelach sztucznego systemu immunologicznego (np. system wieloagentowy Epsteina i Axtella) zakłada dodatkowe reguły: ● zainfekowane jednostki mogą przenosić chorobę na sąsiadów ●choroby obniżają zdolność obronną systemu ●jeżeli system jest uodporniony na daną chorobę to nie ma ona wpływu na dalsze jego działania.

6.4. Systemy immunologiczne na maszynie cyfrowej

Podstawowe operacje na systemie immunologicznym to:

1. Ocena osobników
    a. dla każdego osobnika p z populacji P oblicz poziom aktywacji egzogenicznej (mierzonej przez znormalizowaną odległość Hamminga od aktualnego optimum)
    b. utwórz populację P_{ex} złożoną z n osobników o najwyższym poziomie pobudzenia
    c. dla osobników z $P\backslash P_{ex}$ oblicz poziom aktywacji endogenicznej reprezentującej stopień odmienności od wzorca
2. Selekcja klonalna i mutacja somatyczna
    a. utwórz populację Pex' wybierając osobniki z P_{ex} drogą selekcji turniejowej
    b. poddaj hipermutacji osobniki z populacji P_{ex}'
    c. utwórz populację Pen stosując selekcję turniejową do populacji osobników pobudzonych endogenicznie
3. Rekrutacja
    a. zastąp pobudzone endogenicznie osobniki z populacji P osobnikami z populacji Pen
    b. wprowadź osobniki z populacji P_{ex}' stosując selekcję turniejową między egzogenicznie pobudzonymi osobnikami p z P a osobnikami p' z P_{ex}'.

Ogólny schemat algorytmu immunologicznego:

```
let G denote a generation and P a population
P_{G=0} = {x^1_{G=0}, ..., x^M_{G=0}}
initialize the initial population of solutions
evaluate every x^l ∈ P_{G=0}, l = 1,...,M
compare_with_antigen_and_update_fitness(P_{G=0})
k=1
while the stopping criteria is not satisfied do
      select P' (an intermediate population) from P_{G=k-1}
      mutate element in P_{G=k}
      evaluate every x^l ∈ P_{G=k'} 1,...,M
compare_with_antigen_and_update_fitness (P_{G=k})
k=k+1
return x = arg max l f(x^l), x^l ∈ P_{G=k'}, the best encountered solution

procedure compare_with_antigen_and_update_fitness(P_{G=k})
      antigen=top y% in (P_{G=k})
      l=0
      while l<2xM
            antibodies ⊂ P_{G=k}
            randomly select y from P_{G=k}
            find x from similarity(y,x) = arg max_{x̄} similarity(y, x̄), x ∈ antibodies
            add similarity(y,x) to the fitness of x ∈ P_{G=k}
            l=l+1
end procedure
```

6.5. Zastosowania

Podstawowe zastosowania algorytmów immunologicznych

- Immunologiczne rozpoznawanie obrazów
- Eksploracyjna analiza danych
- Kompresja danych
- Immunologiczne algorytmy genetyczne
- Immunologiczna optymalizacja
- Detekcja anomalii

Pewnym mankamentem algorytmów immunologicznych (podobnie zresztą jak i algorytmów genetycznych) jest brak formalnego aparatu matematycznego umożliwiającego precyzyjną analizę ich własności. Stosowane najchętniej w takich sytuacjach metody oparte na teorii łańcuchów Markowa, koncentrują się w gruncie rzeczy na analizie strategii ewolucyjnych. Stosując analizę probabilistyczną do problemu można próbować stosować nieco uproszczone podejście.

Gałąź wiedzy, jaką są sztuczne systemy immunologiczne można uznać za część dziedziny nauki zwanej sztucznym życiem. Jej celem jest odkrycie uniwersalnych cech życia – nie tylko takiego, jakiego znamy, ale także takiego, jakie mogłoby funkcjonować w rzeczywistości. Zdaniem niektórych badaczy system immunologiczny jest najbardziej skomplikowanym „myślącym" systemem, jaki występuje na naszej planecie. Dotychczas rozpoznano tylko jego najbardziej fundamentalne właściwości, więc nadzieje o ekspansji systemów immunologicznych na inne gałęzie nauki nie jest bezpodstawna

7. Algorytmy mrówkowe (Ant colony optimization)

Teraz w naszej podróży przez metody heurystyczne wybierzemy się wprost to mrowiska. Jeśli zastanawiasz się czy się nam coś nie pomyliło od razu zapewniam, że wszystko jest w porządku – teraz poznamy, jak można wykorzystać zachowanie mrówek w algorytmach optymalizacji rozwiązań problemów. Algorytmy mrówkowe jest to nowo rozwijana forma sztucznej inteligencji w anglojęzycznej literaturze nazywanej „*swarm itelligence*", co możemy przetłumaczyć jako inteligencja roju lub kolonii. Na jej płaszczyźnie bada się kolektywną inteligencję, jaką wykazują grupy prostych stworzeń. Na przykład mrówki czy pszczoły żyją w koloniach, w których indywidualny osobnik wykonuje tylko proste zadania, podczas gdy wspólna praca całej koloni okazuje się być bardzo „przemyślana".

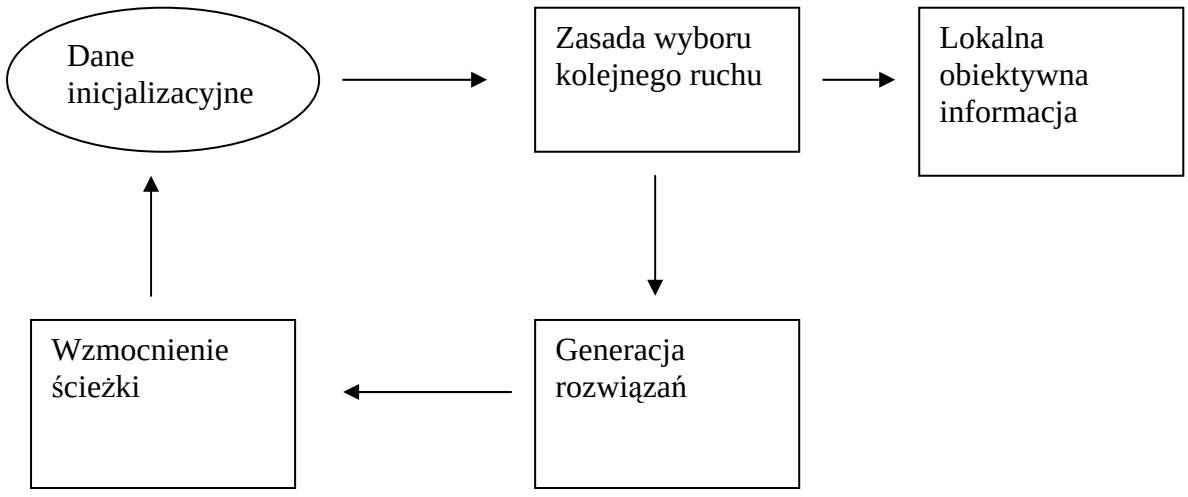
Tak naprawdę prawdziwe mrówki są ślepe, ale każda mrówka poruszając się pozostawia za sobą ślad. Jest to substancja chemiczna zwana feromonem. Ślady te zachęcają pozostałe mrówki do pozostawania blisko ścieżki, po której poruszały się ich poprzednice. Fermowy z biegiem czasu wyparowują. Zachowanie mrówek łatwo można zilustrować na przykładzie budowania ścieżki prowadzącej do pożywienia. Kiedy na drodze mrówek pojawia się przeszkoda, którą można ominąć na dwa sposoby, a jeden z nich jest zdecydowanie lepszy, tzn. bardziej optymalny. Początkowo taka sama liczba mrówek porusza się w obu możliwych kierunkach. Zakładając, że wszystkie mrówki poruszają się z taka sama prędkością te, które poruszają się krótszą ścieżką powrócą (z jedzonkiem) szybciej niż pozostałe. Z czasem ilość feromonów pozostawianych przez mrówki zwiększa się na krótszej ścieżce i w związku z tym więcej mrówek wybiera tą właśnie ścieżkę. Ten efekt nazywany jest autokatalizą, a różnicę między ścieżkami nazywa się efektem peryferyjnej ścieżki. Podsumowując, mrówki poruszające się po krótszej ścieżce składają więcej wizyt u źródła (np. mrowiska) niż te poruszające się do ścieżce dłuższej, a ponieważ fermowy z czasem wyparowują dłuższa ścieżka zanika z czasem. Ten prosty mechanizm pokazuje współpracę między osobnikami.

Pierwszy algorytm oparty na zachowaniu prawdziwych mrówek został stworzony przez Dorigo w 1991 roku pod nazwą *„Ant System"* (AS) do rozwiązywania problemów optymalizacji kombinatorycznej. Działał on dobrze dla małych problemów, ale miał problemy z zadaniami dużej skali. W celu poprawienia działania AS zostały wprowadzone dwie główne zmiany. Po pierwsze zostały dodane wyspecjalizowane techniki szukania lokalnego, a po drugie dozwolono mrówkom pozostawiać fermowy dodatkowo podczas budowania rozwiązania, kiedy w pierwotnej wersji algorytmu były one zostawiane tylko po ukończeniu rozwiązania. Poniżej znajduje się poprawiona wersja ogólnego algorytmu mrówkowego autorstwa Diogro i Cargo z 1999 roku:

7.1. Algorytm

```
procedure ACO_heuristic()
      initialize pheromone_table
      while (termination_criterion_not_satisfied)
            foreach ant k do
                  initialize _ant ();
                  M ← update _ant _memory ();
                  Ω ← a set of problem's constraints
                  while (current _state ≠ target _state)
                        A=read _local _ant – routing _table ();
                        P=compute _transition _probabilit ies (A, M, Ω)
                        next _state = apply _ant _decision _policy (P, Ω)
                        move _to _next _state (next _state);
                        if (online _step _by _step _pheronome _update)
                        then
                              deposit _phermone _on _the _visited _arc ();
                              update _ant _routing _table ();
                        M ← update _int ernal _state ();
            if (online_delayed_pheromone_update)
            then foreach visited_arc do
                  deposit _phermone _on _the _visited _arc ();
                  update _ant _routing _table ();
            die();
            update_thepheromone_table();
end procedure
```

Schemat blokowy działania algorytmu:



Tablica feromonów inicjalizowana jest z równymi feromonami, reprezentuje ona ilość fermonów pozostawionych przez mrówki pomiędzy dwoma stanami (np. węzłami w grafie). Dlatego tablica jest zwykle kwadratowa o rozmiarach zależnych od ilości stanów w danym problemie. Dopóki nie zostanie spełnione kryterium stopu tworzone są nowe mrówki, które inicjalizuje się w stanie początkowym. Mrówka zaczyna konstruować ścieżkę ze stanu początkowego do predefiniowanego stanu końcowego (generacja rozwiązań) wykorzystując probabilistyczna zasadę

wyboru kolejnych ruchów opartą na tablicy ruchów mrówki (*ant routing table)*. W zależności od zasady aktualizacji fermonów, mrówka aktualizuje tablicę ruchów (Wzmacnianie ścieżki). Ma to miejsce albo, gdy kolejna mrówka skonstruuje rozwiązanie (*online update* rule) albo po tym jak wszystkie mrówki zbudują rozwiązania (*delayed update rule)*.

7.2. Tablica ruchów mrówki – reguła wyboru kolejnego ruchu

Tablica ruchów mrówki jest to normalizacja tablicy feromonów, gdzie mrówka buduje ścieżkę na podstawie probabilistycznej zasady opartej na dostępnych feromonach w każdym możliwym kroku i jej [mrówki] pamięci. W literaturze możemy odnaleźć sporo sugestii, co do sposobu podejmowania decyzji o kolejnym kroku. (Element macierzy A [z powyższego algorytmu] z wiersza i i kolumny j reprezentuje prawdopodobieństwo, że mrówka w kolejnym ruchu przejdzie ze stanu i do stanu j.) Można wyróżnić następujące reguły:

1. Mrówka wykorzystuje informacje o feromonach tylko do podjęcia decyzji o następnym kroku – kolejny stan o największym prawdopodobieństwie to stan z największą ilością feromonów. Niestety taka, najprostsza strategia w łatwy sposób prowadzi do stagnacji.
2. Zasada zaproponowana w 1991 roku m.in. przez Dorigo wymaga określenia dwu parametrów:

$$\alpha_{ij} = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{l\in N_i}\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}$$

   Heurystyczna wartość η używana jest do intensyfikowania poszukiwań na zasadzie zachowania zachłannego. Na przykład, wartość heurystyczna może być natychmiastową zmianą obiektywnego rozwiązania pochodzącego od zwiększenia zmiennej o jedną jednostkę niezależnie od efektu zwiększenia ogólnego rozwiązania. Gdy $\beta=1, \alpha=0$ algorytm zachowuje się jakby szukał rozwiązania lokalnego, a gdyby $\beta=0, \alpha=1$ może dojść do stagnacji tak, jak w poprzednim przypadku. Z reguły wymaga się balansu pomiędzy α (wagą informacji feromonowej) a β (wagą rozwiązania lokalnego). Jednakże zasada ta jest droga, z obliczeniowego punktu widzenia przez występowanie eksponent.
3. Aby pokonać niedogodności poprzedniej strategii Diogro wraz z Gambardellą w 1997 roku zaproponowali nowa regułę:

$$\alpha_{ij} = \frac{\left[\tau_{ij}(t)\right]\left[\eta_{ij}\right]^{\beta}}{\sum_{l\in N_i}\left[\tau_{il}(t)\right]\left[\eta_{il}\right]^{\beta}}$$

   We wzorze występuje tylko parametr β.
4. Jeszcze jednym możliwym podejściem jest przełączanie się pomiędzy poprzednimi zasadami a zasadą wyboru przejścia o maksymalnym poziomie feromonów z pewnym prawdopodobieństwem (algorytm Metropolisa).

Dla osób zainteresowanych implementacją algorytmów polecamy jeszcze zapoznanie się ze sposobami aktualizacji feromonów podczas generowania kolejnych rozwiązań. Podsumowując, na początku działania algorytmu inicjalizowana jest tablica feromonów. Na każdym kroku algorytmu tablica ta jest normalizowana tak, aby skonstruować tablicę ruchów mrówki. Mrówki generują

kolejne rozwiązania (każda jedno) przemieszczając się ze stanu do stanu na podstawie zasady wyboru kolejnych ruchów. Każdy element tablicy feromonów jest wtedy aktualizowany używając kroku aktualizacji feromonów i algorytm dalej kontynuuje działanie.

7.3. Przykład zastosowania

Algorytmy mrówkowe stosuje się z powodzeniem do wielu problemów optymalizacji globalnej, w których napotyka się na barierę złożoności obliczeniowej, niemożliwą do pokonania przez algorytmy deterministyczne. Dla przykładu przedstawimy podejście do problemu TSP za pomocą algorytmu mrówkowego.

Dzięki wykorzystaniu właściwości feromonów można łatwo zabezpieczyć się przed osiąganiem lokalnych, a niezadowalających minimów (wyparowywanie). Stosuje się także jednoczesne poszukiwanie różnych rozwiązań, a poszczególne procesy-mrówki komunikują się ze sobą za pomocą feromonów. Odszukiwanie optymalnej ścieżki podobne jest do omijania przeszkód na drodze do pożywienia, z tym, że tutaj omijamy kosztowne ścieżki. Metodą prób i błędów mrówki uczą się, jak dojść do zadanego celu używając polityki wzmacniania odpowiednich ścieżek.

Innym przykładem zastosowania algorytmu mrówkowego może być rozwiązanie wielowymiarowego problemu Knapsack'a, który polega na odszukaniu podzbioru elementów, który maksymalizuje dna funkcję zachowując jednocześnie źródłowe ograniczenia.

8. Artificial life (Sztuczne życie)

Algorytmy mrówkowe możemy zaliczyć do szerszej klasy metod sztucznej inteligencji popularnie zwanej sztucznym życiem. Idea ta narodziła się w 1987 roku na konferencji naukowej w Nowym Meksyku w Stanach Zjednoczonych, ale jej początki możemy datować na prawie 20 lat wstecz, kiedy to w 1968 roku Aristin Lindenmaier podjął próbę stworzenia uniwersalnego języka genetycznego używanego przez rośliny i w rezultacie rozwinął algorytmy odtwarzające strukturę roślin. W wyniku tych prac powstał matematyczny opis wzrostu roślin, na cześć naukowca nazwany L-systemem. L-systemy znalazły profesjonalne zastosowania w studiach animacji komputerowych do tworzenia realistycznie wyglądających roślin. Ściślej mówiąc sztuczne życie to dyscyplina poświęcona zrozumieniu życia przez próbę wyciągania ogólnych teorii z podstawowych zjawisk biologicznych i przenoszenie ich na grunt innych fizycznych mediów, jak na przykład komputerów, umożliwiając w ten sposób stosowanie tych teorii w nowych rodzajach eksperymentów i testów naukowych. Badania łączą biologie z informatyką. Można powiedzieć, że sztuczne życie to alternatywna forma życia – dosłownie *„życie stworzone przez człowieka, nie przez naturę”*. W odróżnieniu od badań biologicznych, które są z założenia analityczne, sztuczne życie jest metodą syntetyczną, próbuje się z małych części zbudować system, który zachowywałby się jak żyjący organizm. Dodatkowo sztuczne życie pozwala na rozszerzenie zakresu studiów biologicznych o formy życia, które mogłyby istnieć.

Główną metoda badawczą sztucznego życia jest ewolucja, która oferuje możliwość adaptacji do dynamicznie zmieniającego się środowiska, pozwala ewoluować systemowi w przypadku nieprzewidzianej zmiany warunków. Jak już wcześniej wspomniano sztuczne życie jest metoda

syntetyczną, w literaturze angielskojęzycznej możemy się spotkać z określeniem *„emergence”*, które charakteryzuje powstanie czegoś więcej niż tylko sumy poszczególnych części. Systemy sztucznego życia składają się z dużej ilości prostych i podstawowych jednostek, których istotne właściwości pojawiają się dopiero na wyższym stopniu konsolidacji. Ciekawym przykładem eksperymentów ze sztucznym życiem jest algorytm Craiga Reynoldsa (1987), który pracował nad zachowaniem stad zwierząt. Algorytm został wykorzystany m.in. do stworzenia animacji nietoperzy w filmie *„Powrót Barmana”*.

9. Sztuczne życie a sztuczna inteligencja

Metoda budowania zaawansowanych robotów Brooksa (1991) demonstruje, że podejście do problemu od strony sztucznego życia różni się zasadniczo od podejścia od strony sztucznej inteligencji. Różnica jest podobna jak między analitycznymi badaniami biologicznymi a sztucznym życiem, tzn. w sztucznej inteligencji stosuje się metodykę *„top-down”*, złożone zachowania są identyfikowane tak, aby zbudować system spełniający wszystkie wymagania. Z kolei sztuczne życie operuje techniką *„bottom-up”*, poczynając od elementarnych jednostek stopniowo buduje system poprzez ich ewolucję, łączenie i rozwój. Ponadto, tradycyjnie sztuczna inteligencja koncentruje się na złożonych funkcjach ludzkich, jak np. gra w szachy, rozumienie tekstu czy stawianie diagnoz, podczas gdy sztuczne życie skupia się na podstawowych, naturalnych zachowaniach, podkreślając zdolność przetrwania w złożonych, dynamicznych środowiskach.

9.1. Jak to się dzieje?

Sztuczne życie jest sztuczne tylko w tym sensie, że zostało stworzone przez człowieka, tzn. środowiska życia zostało wytworzone wewnątrz komputera, ale zasady życia są naprawdę uniwersalne i stosują się nawet w naturalnym otoczeniu. Oczekiwanie na użyteczną do powstania życia kombinację atomów jest zastąpione zbiorem zasad do przestrzegania przez stworzenia i komórki. Jeśli te stworzenia maja zdolność reprodukcji i pewnej formy mutacji oraz są dostatecznie złożone mogą ewoluować w nowe formy życia. Rezultatem jest zagłada populacji albo odnalezienie rozwiązania, jak przetrwać w danym środowisku. Środowisko może być problemem programistycznym, problemem matematycznym do rozwiązania albo obszarem pamięci dzielonym z innymi podobnymi stworzeniami. Nie ma wymogu, aby środowisko odzwierciedlało rzeczywiste życie, w końcu kreując sztuczne życie poszukujemy również jakiś korzyści dla nas samych. Sztuczne życie może rozwiązać wiele problemów, które teraz wydają się być zbyt złożone, aby je rozwiązać od ręki. Sztuczne życie stosuje się w bardzo szerokim zakresie, najważniejsze przypadki wymienimy:

- Chemia syntetyczna – modelowanie nowych związków
- Projektowanie samolotów
- Maszynowe uczenie
- Rzeczywistość wirtualna
- Ekosystemy, wirtualny świat – modelowanie zachowania ekosystemów i ewolucji populacji, systemy te pozwalają na symulowanie dużej ilości interakcji oraz pomiar aktywności ewolucyjnej. Głównym celem badań jest poznanie wpływu prostych interakcji pomiędzy

pojedynczymi stworzeniami na całość populacji i środowiska. Jedną z przełomowych prac w dziedzinie sztucznego życia było stworzenie przez Thomasa Raya sztucznego ekosystemu Tierra (esp.'Ziemia'). Organizmy walczą w nim o przetrwanie zdobywając pożywienie i wydając swoje potomstwo. Autor celowo dopuścił dużą skalę różnorodności organizmów, aby uzyskać w miarę pełny obraz ewolucji. Współzawodnictwo polega na wypracowaniu przez organizmy strategii umożliwiającej im jak najdłuższe przetrwanie w ekosystemie. Symulacja Tierry odbywa się na komputerze, gdzie cyfrowe organizmy - zbiory instrukcji w asemblerze - walczą o miejsce w jego pamięci. Już pierwsze doświadczenia pokazały różnorodność form życia, jakie mogą wykształcić się w trakcie ewolucji. Dzięki tym doświadczeniom stwierdzono, że życie nie musi ograniczać się do organizmów zbudowanych ze związków węgla, ale można je stworzyć w środowisku komputerowym, wykorzystując instrukcje logiczne. Podobny projekt tworzony jest przez polskich naukowców w Instytucie Informatyki Politechniki Poznańskiej pod nazwą Framesticks, którego witryna znajduje się pod adresem: http://www.frams.alife.pl/pindex.html

- Robotyka/Psychologia – studium nad mechanizmami wyboru danych zachowań przez organizmy w określonych momentach. W tradycyjnej robotyce programiści starają się przewidzieć i z góry zaprogramować wszystkie możliwe zachowania, co w przypadku nieoczekiwanych zdarzeń prowadzi do błędów. Z kolei, zdecentralizowana, adapcyjna kontrola ruchów robota przez sztuczne życie jest osiągana przez zastosowanie sensorów i kontrolerów, które nieustannie się uczą i adaptują do zmieniających się warunków środowiska. Ponieważ inteligentny automat nie może zostać zbudowany, musi on ewoluować w procesie podobnym do ewolucji inteligencji w naturze: używając kombinacji ewolucji przez naturalną selekcję, adapcyjności i rozwoju.
- Kontrola ruchu
- Edukacja – symulatory sztucznego życia uczące biologii w szczególności dzieci.
- Wirusy komputerowe – wirusy komputerowe mogą być widziane, jako pewna forma sztucznego życia (Spafford, 1991). Są to programy próbujące zaspokoić swoje cele bez interwencji człowieka. Typowo, celem wirusa jest reprodukcja i „zarażenie" jak największej ilości komputerów. Często wirusy potrafią integrować swój kod bezpośrednio z kodem innych programów, dzięki czemu są uruchamiane przez użytkownika nieświadomie. Najczęściej efekty wirusów nie są pożądane, ale możemy sobie wyobrazić wirusy, które, jak w jogurcie Danone, mogłyby spełniać pożyteczne funkcje, jak wyszukiwanie i destrukcja anomalii w bazie danych zapewniająca integralność. Podobieństwo wirusów komputerowych do tych biologicznych powoduje szerokie zainteresowanie nimi w zakresie sztucznego życia, np. IBM „immune system" – system antywirusowy
- Do innych stworzonych form życia zalicza się: pętle (rozmnażające się litery), boidy (istoty podobne do ptaków), animki (kwadraty poszukujące pokarmu), bimorfy (rozmnażające się kształty), L-systemy (sztuczne kwiaty).

10. Podsumowanie

Ze względu na ograniczoną długość tego opracowania nie było możliwe opisanie wszystkich możliwości, jakie dają nam algorytmy heurystyczne oraz perspektyw, jakie stwarza przed programistami natura. Dlatego tu tylko krótko wymienimy inne idee, które dają możliwość badań nad sztuczna inteligencją.

- Tabu serach – metoda szukania mająca na celu uniknięcie lokalnych, niedozwolonych (tabu) rozwiązań
- Systemy agentowe

- Obliczenia molekularne
- Obliczenia kwantowe
- Membrane computing – okrywanie idei informatycznych na podstawie funkcjonowania i struktury żywych stworzeń
- Automaty komórkowe (cellular automata)

Jeszcze na zakończenie podajemy ciekawy przykład zastosowania algorytmów heurystycznych, jakim jest gra Mastermind. Mastermind jest to prosta gra logiczna polegająca na zgadnięciu kombinacji kolorów na podstawie dawanych podpowiedzi w ograniczonej liczbie prób. Funkcja celu ma w sobie dwa składniki. Jeden odpowiada za ilość prób (pytań), drugi zależny jest od ilości sprawdzonych kombinacji, które nie zostały wykorzystane w grze. „Ludzka" technika gry polega na stopniowym zgadywaniu kolejnych kolorów – ze względu na ograniczone możliwości analizowania kombinacji człowiek stara się tak układać kolory, aby móc dojść do wniosków dotyczących kolejnych pól, a nie rozwiązania jako całości (porównuje do ostatnich wyników wynik nowo uzyskany).

Algorytm heurystyczny opiera się na możliwości analizy wielu kombinacji, z których wybierane są te, które są najbardziej prawdopodobne mogące jednocześnie dostarczyć jak najwięcej informacji o następnych. Bardzo trafny jest algorytm, który dwie kolejne próby wykonuje losowo, następnie wyciąga z nich wniosek. Potem znowu dwie losowe próby (ale już oparte na wyniku pierwszych dwóch prób) i wniosek. z dwóch wniosków – wyciąganie kolejnego bardziej złożony wniosek. Jest to analogia do algorytmów *divie and conquer*, w których na podstawie wyników pośrednich tworzone jest całościowe rozwiązanie.

Jak widać na powyższym przykładzie próba stworzenia sztucznego gracza, pewna namiastka sztucznej inteligencji o ograniczonym zakresie rozwiązywania problemów, tworzona jest inaczej niż pierwowzór. Człowiek ze względu na ograniczone zdolności obliczeniowe musi bardziej położyć nacisk na stopniowe rozwiązywanie problemów. Maszyna cyfrowa z kolei może rozpatrywać szerszy zakres problemu i wyciągać informacje z większej ilości wniosków jednocześnie. Jest to istotna zaleta w problemach typowo kombinatorycznych. Niektórzy twierdzą już, że na tym polu *sztuczna inteligencja* pokonała człowieka. Jako przykład podają spektakularne (bo pierwsze w historii) zwycięstwo maszyny *IBM Deep Blue* nad ówczesnym mistrzem świata Garri Kasparowem roku 1997. Zainteresowanych tematem zastosowań algorytmów heurystycznych odsyłamy do literatury i raportów z badań naukowych, gdzie można odnaleźć wiele interesujących przykładów (np. zastosowanie algorytmów heurystycznych w prognozowaniu pogody [3]).

10.1. Jak daleko nam od myślących maszyn?

Należy pamiętać, że za każdym z opisanych powyżej zagadnień kryje się, zwykle bardzo skomplikowany aparat matematyczny, którego podstawy staraliśmy się wyjaśnić. Dzięki niemu komputery potrafią wnioskować i efektywnie przeszukiwać przestrzeń w poszukiwaniu rozwiązań wielu złożonych problemów. Obecnie rozwijanych jest szereg projektów, w których próbuje się symulować sztuczną inteligencję. Najciekawsze z nich to systemy oparte na architekturze SOAR (Laird, 1987) i THEO (Mitchell, 1990). Z nowszych mamy symulator Intellibuddy, program stworzony w 1995 roku przez Richarda Wallece'a, niezależnego informatyka z San Francisco. Program ten służy do prowadzenia czatów z maszyną ALICE (Artificial Linguistic Internet Computer Entity), jest on dostępny postaci Open Source. Niestety, nawet, jeśli początkowo

konwersacja z wirtualnym przyjacielem początkowo zapowiada się obiecująco, zwykle kończy się bezsensownymi odpowiedziami udzielanymi przez komputer. W dziedzinach takich gry logiczne, dowodzenie twierdzeń, wnioskowanie, planowanie czy diagnozowanie medyczne istnieją systemy oparte na rygorystycznych zasadach teoretycznych, które potrafią działać równie dobrze, a czasami nawet lepiej niż człowiek. W innych dziedzinach, jak na przykład uczenie się, robotyka czy rozumienie języka naturalnego ciągłe zachodzą szybkie ulepszenia, które zawdzięczamy lepszym metodą analitycznym i poprawie zrozumienia podstaw tych problemów. Jednym z głównych problemów jest tendencja projektantów systemów sztucznej inteligencji do nadmiernego skupiania się nad aspektami technicznymi, co czasami powoduje, że tracą z pola widzenia nadrzędny cel swoich badań. Rozważania nad istnieniem narzędzi, które pozwoliłyby stworzyć kompletny i uniwersalny inteligentny system maja również pomóc ujawnić luki we współczesnym rozumieniu sztucznej inteligencji.

Pewne nadzieje wiąże się najnowszym projektem znanym pod nazwą CYC (czyt. sajk), nad którym prace trwają już od 22 lat, a którego publiczna premiera jest planowana za kilka miesięcy. System ma docelowo być dostępny w Internecie, który ma być dla niego bazą i źródłem wiedzy. Twórca systemu, Doug Lenat z firmy Cycrop, podkreśla, że ta sztuczna inteligencja ma cechę brakującą poprzedniczkom - zdrowy rozsądek. - *Wierzę, że powoli zmierzamy w kierunku osobliwości technologicznej. Przekonamy się o tym w ciągu 10 lat* - dodaje.

10.2. Co jeśli nam się uda?

Prace nad stworzeniem sztucznej inteligencji należą do najbardziej kontrowersyjnych współczesnych dyscyplin. Obawy dotyczą przede wszystkim możliwości pojawienia się maszyn, które będą dorównywały poziomem intelektualnym człowiekowi. Sztuczna inteligencja jest bez wątpliwości dziedziną fascynującą – z pewnością inteligentne komputery byłyby dla nas bardziej użyteczne niż komputery niemyślące, wiec czy jest sens się martwić? Wymyślenie zastosowań dla sztucznej inteligencji w rzeczywistym życiu nie trudnym zadaniem. Już teraz pewne jej elementy funkcjonują w naszym otoczeniu. Stosowane są w programach obliczających zdolność kredytową, systemach sterujących sygnalizacją świetlną w miastach, automatycznych fotoradarach zapamiętujących numery rejestracyjne samochodów, w oprogramowaniu nawigującym statkami kosmicznymi. Można by snuć teoretyczne rozważania, kto powinien odpowiadać za błędy systemów AI, które na przykład diagnozują choroby, czy wspomagają operacje. Jak na razie w Stanach Zjednoczonych w podobnych sprawach systemy eksperckie są traktowana przez sądy tak samo, jak książki i poradniki medyczne. Podobne problemy stwarzają systemy zarządzania finansami. Nie wiadomo czy można traktować system komputerowy, jako osobę fizyczna posiadająca akcje i udziały oraz kto powinien odpowiadać za ewentualne długi sprokurowane przez błędne decyzje takich systemów. Takich przykładów można by jeszcze mnożyć, ale pojawiają się również bardziej skrajne teorie na temat pseudo użyteczności sztucznej inteligencji. Wszyscy dobrze znamy opowiadania Izaaka Asimova czy Philipa Dicka, którzy ze sztuczną inteligencja wiążą koniec naszej cywilizacji. Futuryści tacy, jak Edward Fredkin i Hans Moravec sugerują, że jeśli rasa ludzka wypełni swoje przeznaczenie powołania do życia stworzeń o wyższej, a być może nieskończonej inteligencji jej przetrwanie stanie się mniej ważne…

Jednak, jeśli spojrzymy na jaśniejszą stronę sukcesu, jaki niesie ze sobą sztuczna inteligencja możemy łatwo wskazać szereg korzyści, które mogą z niego wyniknąć. Poprawa poziomu życia, łatwiejsza i przyjemniejsza praca, bardziej relaksujący odpoczynek – to tylko takie bardziej przyziemne jego elementy. Łatwo sobie wyobrazić świat, w którym dostęp do inteligentnych nauczycieli mamy bez wychodzenia z domu, świat zarządzany przez elektroniczne mózgi, w

którym nie ma miejsca na biedę, wojny. Autorzy science fiction nad taki utopijny świat preferują bardziej katastroficzne wizje, ale może tylko dlatego, że oferują one ciekawszą fabułę. Przyszłość jednak może nie zmierza w tak definitywnie pesymistycznym kierunku. Inna sprawą jest czy chcemy żyć w rzeczywistości doskonałej, w której nie ma miejsca na pomyłkę, a wszystko jest już zależy wyłącznie od wirtualnych zarządców.

11. Bibliografia i referencje

1. "Data Mining: A Heuristic Approach" - Hussein A. Abbass, Ruhul A. Sarker, Charles S. Newton
2. "A heuristic hill climbing algorithm for Mastermind" – A. Temporel, T. Kovacs
3. "Heuristic models of fuzzy time series for forecasting" - Kunhuang Huarng
4. "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems" - Kenneth Rose
5. "A efficient limited-memory heuristic tree search algorithm" – S. Ghosh, A. Mahanti, D. S. Nau
6. "Computer and Intractability – A Guide to the theory of NP-completeness" – M. R. Garey, D. s. Johnson
7. "Hybrid constrained simulated annealing and genetic algorithms for nonlinear constrained optimization" – B. W. Wah, y. Chen
8. "Optimization by Simulated Annealing" - S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi
9. "An Introduction to Genetic Algorithms" - Mitchell Melanie
10. "Real-time lane recognition by simulated annealing algorithm" – S. T. Park, S. Y. Yang, J. H. Yung
11. "Artificial Intelligence A Modern Approach" - Stuart J. Russell, Peter Norvig
12. "Genetic algorithms vs. simulated annealing" – T. W. Manikas, J. T. Cain
13. „Simulated annealing overview" - Franco Busetti
14. "Computer Viruses as Artificial Life" - Eugene H. Spafford
15. "Heuristic and learning approaches for solving the traveling salesman problem" – G. Erban, C. Pintea
16. "Ant Algorithm For The Multidimensional Knapsack Problem" - In`es Alaya
17. "Sztuczne systemy immunologiczne" - Sławomir T. Wierzchom
18. „Numerical Recipies In C" - Numerical Recipes Software.
19. "The Traveling Salesman Problem: A Case Study in Local Optimization" - David S. Johnson, Lyle A. McGeoch
20. "Wstęp do obliczeń ewolucyjnych i neuronowych" - Kazimierz Rygiel
21. „Sieci neuronowe" – Ryszard. Tadeusiewicz
22. PCkurier 3/1999 : „*Sztuczna inteligencja*" - *Cezary Głowiński*
23. www.doc.ic.ac.uk/~nd/surprise_96/journal/vol2/tcw2/article2.html
24. www.stewdean.com/alife/intro.html
25. http://pl.wikipedia.org