Ewa FIGIELSKA[1]

ALGORYTMY EWOLUCYJNE I ICH ZASTOSOWANIA

Streszczenie:

Pojęcie algorytmy ewolucyjne obejmuje metodologie inspirowane darwinowską zasadą doboru naturalnego stosowane do rozwiązywania trudnych zagadnień. W artykule przedstawione są podstawowe cztery typy algorytmów ewolucyjnych: algorytmy genetyczne, programowanie genetyczne, strategie ewolucyjne i programowanie ewolucyjne, omówiona jest i zilustrowana przykładem zasada działania algorytmu ewolucyjnego oraz przedstawione są przykłady zastosowań algorytmów ewolucyjnych w praktyce.

The term evolutionary algorithm encompasses methodologies inspired by the principles of genetics and Darwinian natural selection that are used for solving hard problems. In this paper four types of evolutionary algorithms are described: genetic algorithms, evolution strategies, genetic programming and evolutionary programming. An example illustrating how an evolutionary algorithm works is shown. Some real-life applications of evolutionary algorithms are presented.

1. WPROWADZENIE

Algorytm ewolucyjny jest to termin określający przybliżony algorytm optymalizacyjny, w którym stosowane są mechanizmy selekcji, reprodukcji i mutacji inspirowane przez biologiczny proces ewolucji. W biologicznym procesie ewolucji na daną populację osobników działa presja środowiska powodując naturalną selekcję. Tylko najlepiej przystosowane osobniki mają szansę na przetrwanie i zapoczątkowanie nowych coraz to lepszych populacji. W algorytmie ewolucyjnym problem, który mamy rozwiązać, gra rolę środowiska, w którym żyje populacja osobników. Każdy osobnik reprezentuje potencjalne (możliwe) rozwiązanie problemu [13]. Podobnie jak w procesie biologicznym, algorytm ewolucyjny tworzy stopniowo coraz to lepsze rozwiązania. Stąd wynika, że algorytm ewolucyjny może służyć do rozwiązywania problemów optymalizacyjnych.

[1] Dr inż. Ewa Figielska jest wykładowcą w Warszawskiej Wyższej Szkole Informatyki oraz w Politechnice Warszawskiej.

Proces optymalizacji polega na przeszukiwaniu przestrzeni potencjalnych rozwiązań danego problemu w celu znalezienia najlepszego rozwiązania. Przykładem problemu optymalizacji może być harmonogramowanie pracy załóg samolotów. Mając dany harmonogram lotów dla pewnego typu samolotów jednym z zadań jest zaprojektowanie tygodniowych harmonogramów pracy załóg. Każdego dnia załodze musi być przydzielony czas pracy składający się z jednego lub więcej powiązanych ze sobą lotów i spełniający ograniczenia dotyczące na przykład maksymalnego całkowitego czasu spędzonego w powietrzu, minimalnego czasu wypoczynku między lotami itp. Następnie, biorąc pod uwagę harmonogramy jednodniowe, konstruowane są harmonogramy tygodniowe, które z kolei muszą spełniać dalsze ograniczenia dotyczące na przykład nocnego wypoczynku, czy też powrotu załogi do macierzystego portu. Celem jest minimalizacja wypłacanej załogom kwoty, która jest funkcją czasu spędzonego w powietrzu, długości czasu pracy, gwarantowanego minimum na godziny lotu itp. [16].

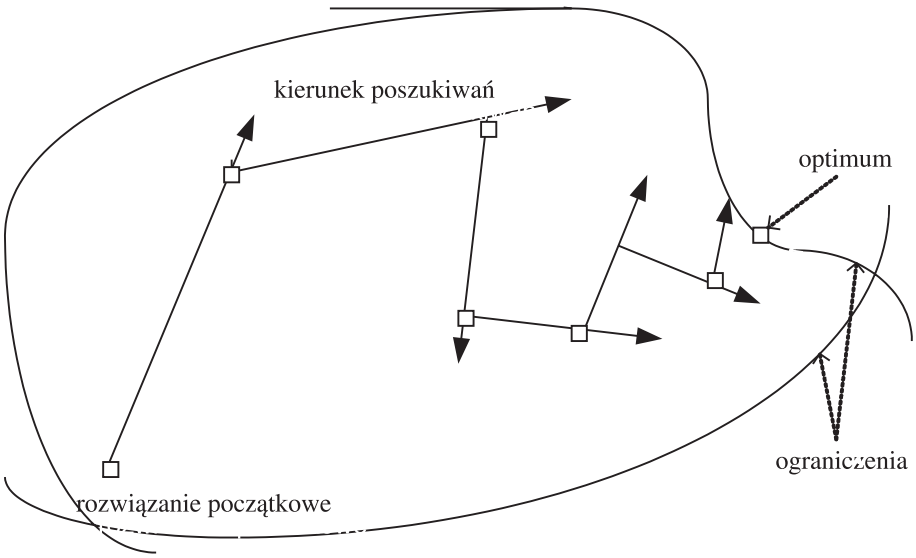
W rzeczywistych zastosowaniach przestrzeń potencjalnych rozwiązań jest zwykle tak duża, że nie jest możliwe w dostatecznie krótkim (realnym) czasie sprawdzenie wszystkich możliwych potencjalnych rozwiązań w celu wybrania najlepszego rozwiązania. Dlatego też, w takich przypadkach uzasadnione staje się stosowanie technik probabilistycznych, które używają wyboru losowego jako narzędzia do ukierunkowania procesu poszukiwań. Algorytmy ewolucyjne są właśnie jedną z takich technik, stosowaną z powodzeniem w praktyce inżynieryjnej.

2. KLASYCZNE PODEJŚCIE DO OPTYMALIZACJI A ALGORYTMY EWOLUCYJNE

Problem optymalizacji może być krótko zapisany jako znalezienie wartości zmiennej x, zawartej w danym zbiorze X, przy której dana funkcja f zmiennej x przyjmuje najkorzystniejszą wartość. Funkcja f, nazywana funkcją celu bądź wskaźnikiem lub kryterium jakości, mierzy cel, jaki ma być osiągnięty. Na przykład celem może być minimalizacja kosztu lub maksymalizacja zysku. W praktyce często występuje wiele rozbieżnych celów. Zbiór X jest wyznaczony przez zbiór ograniczeń problemu, na przykład na dostępność zasobów czy wielkość zamówień.

Większość klasycznych algorytmów optymalizacyjnych stosuje deterministyczną procedurę, która punkt po punkcie zbliża się do rozwiązania optymalnego. Procedura taka zwykle zaczyna poszukiwanie rozwiązania optymalnego startując z pewnego wybranego rozwiązania, po czym, na podstawie informacji lokalnych, określany jest kierunek poszukiwań. Następnie przeprowadzane jest jednokierunkowe poszukiwanie najlepszego rozwiązania. To najlepsze rozwiązanie staje się nowym rozwiązaniem i powyższa procedura jest powtarzana określoną liczbę razy. Algorytmy klasyczne różnią się między sobą głównie sposobem określania kierunku poszukiwań.

Wśród algorytmów tych można wyróżnić metody bezpośrednie i gradientowe. Metody bezpośrednie przy wyznaczaniu kierunku poszukiwań wykorzystują tylko wartości funkcji celu i ograniczeń. Metody gradientowe posługują się pojęciem pierwszej i drugiej pochodnej funkcji celu lub ograniczeń [11]. Przykład działania klasycznego algorytmu optymalizacji jest przedstawiony na Rys. 1.



Rys.1. Działanie klasycznego algorytmu optymalizacji (opracowanie na podstawie [2])

Podstawowe trudności przy stosowaniu metod klasycznych są następujące [11]:

- Zbieżność algorytmu do rozwiązania optymalnego zależy od wyboru rozwiązania początkowego.
- Większość algorytmów ma tendencję do utknięcia w rozwiązaniu suboptymalnym.
- Algorytm, który może być efektywny przy rozwiązywaniu danego problemu optymalizacji, może nie być efektywny przy rozwiązywaniu innego problemu.
- Algorytmy nie są efektywne w zastosowaniu do problemów z dyskretną przestrzenią poszukiwań.
- Algorytmy nie mogą zostać w sposób efektywny wykorzystane na równoległych maszynach.

Algorytmy ewolucyjne radzą sobie z większością trudności, jakie napotykane są przy stosowaniu algorytmów klasycznych. Algorytmy ewolucyjne mają wyjątkową zdolność łatwej adaptacji i mogą być stosowane przy rozwiązywaniu złożonych

nieliniowych i wielowymiarowych problemów inżynieryjnych. Jakość ich działania nie zależy od problemu, nie ma znaczenia jego struktura ani różniczkowalność.

Podstawowe cechy algorytmów ewolucyjnych odróżniające je od innych metod są następujące [6]:

- Nie przetwarzają one bezpośrednio parametrów zadania, lecz ich zakodowaną postać.
- Prowadzą poszukiwania, 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.

3. ALGORYTMY EWOLUCYJNE

Działanie algorytmu ewolucyjnego można opisać następująco: algorytm ewolucyjny rozpoczyna proces przeszukiwania od utworzenia populacji potencjalnych rozwiązań nazywanych osobnikami, które są reprezentowane przez chromosomy zawierające genetyczną informację o osobnikach. W każdym ewolucyjnym kroku, nazywanym generacją, chromosomy są dekodowane i ocenianie zgodnie z pewnym z góry przyjętym kryterium jakości nazywanym przystosowaniem (funkcją przystosowania może być na przykład funkcja celu), a następnie przeprowadzana jest selekcja w celu eliminacji osobników ocenionych jako najgorsze. Osobniki wykazujące wysokie przystosowanie podlegają mutacji oraz rekombinacji przeprowadzanej przy pomocy operatora krzyżowania. Sama selekcja nie wprowadza żadnego nowego osobnika do populacji, tj. nie znajduje nowych punktów w przestrzeni poszukiwań, natomiast takie punkty wprowadzane są przez krzyżowanie i mutację. Dzięki krzyżowaniu ewolucyjny proces może się przesuwać w kierunku obiecujących obszarów w przestrzeni poszukiwań. Mutacja zapobiega zbieżności do lokalnego optimum. W wyniku działania operatora krzyżowania i mutacji tworzone są nowe rozwiązania, z których następnie budowana jest populacja następnej generacji. Warunkiem zakończenia algorytmu może być na przykład pewna określona liczba generacji albo osiągnięcie zadawalającego poziomu przystosowania.

Niech *P(t)* oznacza populację w generacji *t*. Ogólny schemat działania algorytmu ewolucyjnego jest następujący:

1. $t = 0$
2. Wygeneruj i oceń początkową populację *P(t)*
3. Dopóki warunek stopu nie jest spełniony wykonuj:
   3.1. $t = t + 1$

3.2. Wybierz $P(t)$ z $P(t - 1)$
3.3. Zmień $P(t)$ stosując operator krzyżowania i mutacji
3.4. Oceń $P(t)$

4. WYŚWIETL NAJLEPSZE ZNALEZIONE ROZWIĄZANIE

Idea zastosowania darwinowskiej zasady doboru naturalnego do automatycznego rozwiązywania problemów pojawiła się w latach pięćdziesiątych, na długo przed „erą komputerów" [5]. W latach sześćdziesiątych trzy różne sposoby zastosowania tej idei pojawiły się w trzech różnych miejscach. W Stanach Zjednoczonych Fogel wprowadził ewolucyjne programowanie [4], podczas gdy Holland [8,6] nazwał swoją metodę algorytmem genetycznym. W Niemczech Rechenberg i Schwetel [14,15] zaproponowali strategie ewolucyjne. Przez około 15 lat metody te rozwijały się niezależnie. Dopiero od początku lat dziewięćdziesiątych są one traktowane jako różne formy jednej techniki - algorytmów ewolucyjnych. Obecnie wyróżnia się następujące podstawowe cztery typy algorytmów ewolucyjnych: algorytmy genetyczne, programowanie genetyczne, programowanie ewolucyjne oraz strategie ewolucyjne.

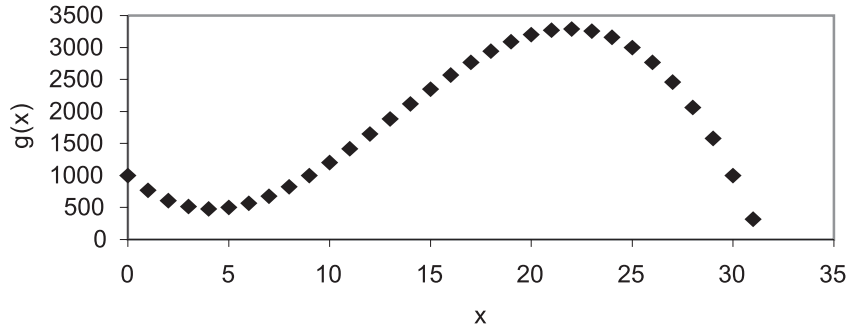
4.1. Algorytmy genetyczne

Algorytmy genetyczne stanowią najlepiej znaną klasę algorytmów ewolucyjnych. Algorytmy te zostały zapoczątkowane przez J. Hollanda w l960 r. Tradycyjnie chromosomy w algorytmach genetycznych są ciągami binarnymi o ustalonej długości (chromosomy mogą być także kodowane za pomocą ciągów liczb całkowitych lub rzeczywistych). Chromosomy są oceniane w każdej generacji. Rozmiar populacji jest stały. Parametrami, które muszą być określone są: wielkość populacji, prawdopodobieństwo mutacji oraz warunek zakończenia działania algorytmu. Z góry musi być również określona funkcja przystosowania.

W celu prześledzenia działania algorytmu genetycznego rozważymy następujący przykład.

Przykład. Dążymy do maksymalizacji funkcji $f(x) = -x^3 + 39x^2 - 270x + 1000$ określonej na zbiorze liczb całkowitych z przedziału [0, 31]. Funkcja ta jest przedstawiona na Rys. 2. Zakodujemy najpierw zmienną x za pomocą ciągu znaków (chromosomu). Prostym sposobem kodowania będą tutaj ciągi złożone z pięciu znaków, będących zerem lub jedynką. Każdy ciąg będzie reprezentował wartość zmiennej x w systemie binarnym. Na przykład ciąg 00000 reprezentuje wartość 0, ciąg 00001 – wartość 1 a ciąg 11111 – wartość 31. Kod pięciobitowy umożliwia operowanie na liczbach całkowitych od 0 do 31. Algorytm genetyczny startuje od po-

czątkowej populacji chromosomów. Populację taką można otrzymać drogą losową dokonując kolejnych rzutów monetą (np. orzeł = 1, reszka= 0). Załóżmy, że w ten sposób została wygenerowana populacja początkowa o rozmiarze $n = 4$ (składająca się z 4 chromosomów). Dla każdego chromosomu określamy jego przystosowanie przez obliczenie odpowiadającej mu wartości funkcji celu (Tablica 1).



Rys. 2. Funkcja $g(x) = -x^3 + 39x^2 - 270x + 1000$ *na zbiorze liczb całkowitych z przedziału [0, 31].*

Tablica 1. Symulacja działania algorytmu genetycznego (opracowanie na podstawie [6])

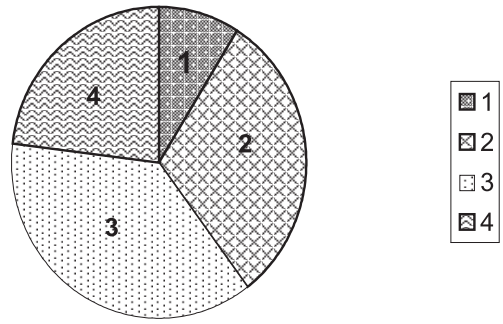
Nr chromosomu	Populacja początkowa	Wartość x	Przystosowanie $f(x)$	$\frac{f_i}{\sum_i f_i}$	Oczekiwana liczba kopii $f_i / \bar{f}$	Liczba wylosowanych kopii
1	00010	2	608	0,09	0,35	0
2	01110	14	2120	0,31	1,23	1
3	10000	16	2568	0,37	1,49	2
4	11101	29	1580	0,23	0,92	1
Suma			**6876**	**1,00**	**4,00**	
Wartość średnia			1719	0,25	1,00	
Wartość maksymalna			2568	0,37	1,49	

f_i- przystosowanie chromosomu i

Następnie z populacji początkowej tworzone są nowe, coraz to lepsze populacje, przy pomocy trzech podstawowych operacji: selekcji, krzyżowania i mutacji.

Selekcja jest procesem, w którym chromosomy zostają powielone w stosunku zależnym od wartości, jakie przybiera dla nich funkcja przystosowania f (która w omawianym przykładzie jest równoważna funkcji celu). Selekcja polega na tym, że chromosomy o wyższym przystosowaniu mają większe prawdopodobieństwo wprowadzenia potomków do następnego pokolenia. Najczęściej używana selekcja jest oparta na zasadzie proporcjonalności

do przystosowania, która zapewnia, że liczba określająca ile razy dany osobnik jest wybrany, jest w przybliżeniu proporcjonalna do jego względnego przystosowania. W najprostszy sposób operację selekcji można zrealizować przez symulację odpowiednio wykalibrowanej tarczy obrotowej (ruletki), gdzie każdemu chromosomowi z populacji odpowiada sektor o rozmiarze proporcjonalnym do wartości względnego przystosowania. Selekcja taka jest nazywana selekcją ruletkową. Rys. 3 przedstawia tarczę z sektorami o rozmiarach proporcjonalnych do wartości względnych przystosowań $f_i / \sum_i f_i$ określonych w Tablicy 1. Na przykład dla chromosomu o numerze 4 przystosowanie wynosi 1580, co stanowi 23% sumy przystosowań, a więc na chromosom ten przypada 0.23 obwodu koła. Selekcji dokonujemy uruchamiając ruletkę czterokrotnie. Dla każdego wylosowanego chromosomu tworzymy jego replikę, która zostaje włączona do nowego pokolenia pośredniego stanowiącego pulę rodzicielską przy dalszych operacjach genetycznych. Pula rodzicielska otrzymana z symulacji procesu selekcji przy pomocy koła ruletki jest pokazana w Tablicy 2.



Rys. 2. Wybieranie chromosomów za pomocą ruletki, której rozmiary sektorów są proporcjonalne do wartości względnych przystosowań (tarcza jest wykalibrowana według danych z Tablicy 1)

Krzyżowanie jest operacją dokonywaną na dwóch losowo wybranych z puli rodzicielskiej chromosomach, zwanych rodzicami. Dla pary rodziców wybieramy w sposób losowy *punkt krzyżowania k*, po czym zamieniamy miejscami wszystkie znaki od pozycji *k + 1* do pozycji *m* włącznie (gdzie *m* jest długością chromosomu (liczbą genów w chromosomie) w obu chromosomach pary rodzicielskiej. W ten sposób powstają dwa nowe chromosomy (chromosomy potomne). Operacja krzyżowania jest zwykle przeprowadzana dla pewnej liczby par rodzicielskich. Populacja po operacji krzyżowania przedstawiona jest w Tablicy 2.

Mutacja jest operacją przeprowadzaną losowo dla każdego genu z osobna. Jeżeli w omawianym eksperymencie przyjmiemy prawdopodobieństwo mutacji równe 0.05, to można oczekiwać, że mutacja dotknie 1 genu w całej populacji. W naszym doświadczeniu uległ zmianie jeden gen, tak jak to pokazano w Tablicy 2.

Tablica 2 Symulacja działania algorytmu genetycznego, cd. (opracowanie na podstawie [6])

Pula rodzicielska	Punkt krzyżowania	Nowa populacja po krzyżowaniu	Wartość x	Przystosowanie $f(x)$	Nowa populacja po mutacji	Wartość x	Przystosowanie $f(x)$
011|10	3	01100	12	1648	01100	12	1648
100|00	3	10010	18	2944	11010	26	2768
11|101	2	11000	24	3160	11000	24	3160
10|000	2	10101	21	3268	10101	21	3268
Suma				**11020**			**10844**
Wartość średnia				2755			2711
Wartość maksymalna				3268			3268

Po przeprowadzaniu selekcji, krzyżowania i mutacji nowe pokolenie jest poddawane ocenie, tzn. obliczane jest przystosowanie poszczególnych chromosomów z tego pokolenia w celu utworzenia puli rodzicielskiej dla następnego pokolenia. Z porównania Tablicy 1 i 2 widać, że wartość średnia przystosowania (funkcji celu) zmieniła się z 1719 na 2711 (po operacji krzyżowania i mutacji) a wartość maksymalna z 2568 na 3268. Jak widać na Rys. 2 druga wartość jest bliska wartości optymalnej. Wynik ten można wytłumaczyć faktem, że najlepszy chromosom z pierwszego pokolenia (10000) otrzymał dwie kopie, dzięki wysokiej wartości przystosowania. Kiedy jedna z nich została skojarzona z trzecim pod względem przystosowania chromosomem (01101), jeden z otrzymanych w ten sposób potomków (10101) okazał się być naprawdę dobrym rozwiązaniem.

Jakość wyników otrzymywanych za pomocą algorytmów genetycznych zależy od wielkości populacji, czasu przeznaczonego na poszukiwanie rozwiązania, doboru sposobu selekcji, zastosowanych operatorów krzyżowania i mutacji oraz prawdopodobieństwa, z jakim te operacje są przeprowadzane. W algorytmach genetycznych stosowane są także inne operatory krzyżowania i mutacji niż przytoczone powyżej. Na przykład dość często stosowane jest krzyżowanie dwupunktowe, w którym dwa punkty są wybierane losowo, po czym następuje wymiana fragmentów chromosomów zawartych między tymi punktami. Operacja mutacji może na przykład polegać na wymianie wartości dwóch losowo wybranych genów. Również inne rodzaje selekcji niż selekcja ruletkowa, na przykład selekcja turniejowa, gdzie losowane są dwa chromosomy, z których lepszy staje się chromosomem rodzicielskim, mogą okazać się lepsze w działaniu dla pewnych problemów. Poza tym algorytmy genetyczne mogą działać nie tylko na ciągach liczb binarnych, ale również na ciągach liczb całkowitych lub rzeczywistych.

4.2. Programowanie genetyczne

Programowanie genetyczne wykorzystuje zasady genetyki i darwinowskiej naturalnej selekcji do tworzenia programów komputerowych [10,11]. Programowanie genetyczne jest w dużym stopniu podobne do algorytmów genetycznych. Podstawowa różnicą między programowaniem genetycznym i algorytmem genetycznym leży w reprezentacji rozwiązania. Podczas gdy algorytm genetyczny tworzy ciąg liczb, który reprezentuje rozwiązanie problemu, w programowaniu genetycznym osobniki są programami o strukturze drzew, a operatory genetyczne są stosowane do gałęzi i węzłów w tych drzewach [9, 11].

4.3. Programowanie ewolucyjne

Programowanie ewolucyjne skupia się głównie na problemach optymalizacji z ciągłymi parametrami. W programowaniu ewolucyjnym każdy osobnik rodzicielski w populacji generuje potomka na drodze mutacji. Prawdopodobieństwo mutacji ma na ogół rozkład równomierny. Po ocenieniu potomków pewien wariant stochastycznej selekcji turniejowej wybiera pewną ilość najlepszych osobników spośród zespołu rodziców i potomków. Najlepszy osobnik jest zawsze przechowywany, co zapewnia, że jeżeli optimum zostaje osiągnięte, nie może on zostać zgubiony. Programowanie ewolucyjne jest algorytmem stosującym wyłącznie mechanizmy selekcji i mutacji (bez krzyżowania) [11] .

4.4. Strategie ewolucyjne

Istnieją dwie główne strategie ewolucyjne (μ, β) ES i $(\mu + \beta)$ ES, gdzie μ rodziców produkuje β potomków wykorzystując operator krzyżowania i mutacji. W (μ, β) ES najlepszych β potomków przeżywa i zastępuje rodziców. W rezultacie rodzice nie są obecni w następnej populacji. Przeciwnie, $(\mu + \beta)$ ES dopuszcza przeżycie zarówno potomków jak i rodziców. W $(\mu + \beta)$ ES stosowana jest strategia elitarna (najlepszy osobnik zawsze jest kopiowany do następnej populacji). W obu wymienionych strategiach przeprowadzana jest zarówno operacja krzyżowania jak i mutacji [11].

5. PRZYKŁADY ZASTOSOWAŃ ALGORYTMÓW EWOLUCYJNYCH

Występujące w realnym świecie problemy charakteryzują się zwykle bardzo dużą liczbą zmiennych (dyskretnych lub ciągłych), dużą złożonością przestrzeni poszukiwań (wiele ograniczeń i celów, które na dodatek mogą być ze sobą sprzeczne).

Problemy te mogą się zmieniać w czasie (być dynamiczne), co pociąga za sobą konieczność szybkiego otrzymywania dobrych rozwiązań.

Ważnym obszarem zastosowań algorytmów ewolucyjnych jest harmonogramowanie i planowanie procesów przemysłowych. Problemy harmonogramowania, gdzie należy przydzielić zasoby do kolekcji zadań, są zwykle trudne do rozwiązania z powodu występowania różnorodnych ograniczeń oraz złożonych struktur produktów. Stosowane tutaj techniki programowania matematycznego pozwalają zwykle na uzyskanie rozwiązań tylko dla problemów o małych rozmiarach. W większości zastosowań algorytmów genetycznych do problemów harmonogramowania, cały harmonogram jest kodowany za pomocą chromosomu (chromosom przechowuje zakodowany harmonogram, a więc określa kolejność wykonywania poszczególnych czynności), co wymaga następnie stosowania odpowiednio zaprojektowanych operatorów genetycznych tak, aby tworzone w kolejnych populacjach chromosomy reprezentowały rozwiązania dopuszczalne (np. aby wykonanie tej samej czynności nie powtarzało się).

Kolejnym obszarem zastosowań algorytmów ewolucyjnych jest przemysł chemiczny. Zakłady chemiczne zawierają urządzenia takie jak pompy, destylatory i reaktory chemiczne, które tworzą złożoną sieć wzajemnie powiązanych strumieni materiału, ogrzewania i informacji, przeznaczonych do przeprowadzania procesu chemicznego, podczas którego surowy materiał jest przekształcany w wymagany produkt. Optymalizacja w zakładach chemicznych polega na modyfikacji struktury parametrów operacyjnych tak, aby znaleźć globalne optimum w celu wykonania pewnego zadania chemicznego [11]. Klasyczne podejścia do rozwiązywania problemów chemicznych nie pozwalają na otrzymanie rozwiązań w zadowalającym czasie. Techniki gradientowe zbiegają do lokalnych optimów i nie są odpowiednie dla występujących tu problemów nieróżniczkowalnych. Stąd też wynika zainteresowanie stosowaniem technik opartych na algorytmach ewolucyjnych w tym klasycznych algorytmów genetycznych. Sukces w stosowaniu algorytmów genetycznych w tak skomplikowanym środowisku wynika z faktu, że na ogół przy pomocy chromosomu kodowana jest tylko niezbędna cześć problemu, co pociąga za sobą niewielki rozmiar chromosomu, co z kolei zwiększa szansę na otrzymanie dobrego dopuszczalnego rozwiązania [7].

Innym przykładem związanym z zastosowaniem przemysłowym może być wykorzystanie algorytmu genetycznego do kontroli fermentacji piwa [1]. Algorytm genetyczny jest tutaj użyty do dopasowania profilu temperatury mieszaniny w ustalonym okresie czasu.

Algorytmy ewolucyjne znalazły również dość szerokie zastosowanie w medycynie. Większość medycznych decyzji może być sformułowana jako poszukiwanie w pewnej odpowiedniej przestrzeni. Na przykład radiolog planując serię naświetlań poszukuje najlepszego sposobu leczenia w przestrzeni wszystkich możliwych sposobów [17]. Przestrzenie poszukiwań w medycynie są zwykle bardzo duże i złożone. Decyzje są oparte na testach klinicznych, które dostarczają ogromnych ilości danych. W oparciu o te dane trzeba ostatecznie podjąć jedną decyzję (np. patolog musi stwierdzić czy nowotwór jest łagodny czy złośliwy na podstawie cech komórek, czyli wśród wszystkich możliwych cech komórek poszukuje się zbioru cech, który pozwala postawić właściwą diagnozę [12]). Algorytmy ewolucyjne są stosowane w medycynie do wykonywania zadań, które mogą być podzielone na trzy grupy: (a) eksploracja danych, głównie w celu diagnozowania i prognozowania, (b) obrazowanie i przetwarzanie sygnałów, (c) planowanie i harmonogramowanie [13]. (a) Eksploracja danych (odkrywanie wiedzy) jest procesem znajdowania wzorów, trendów i regularności poprzez „przesiewanie" dużych ilości danych [3]. W medycznej eksploracji danych algorytmy ewolucyjne są zwykle wykorzystywane w celu znajdowania wartości parametrów ustawianych przez projektanta tak, aby przeszukiwane dane były interpretowane w sposób optymalny (np. znajdowanie wag w sieciach neuronowych). (b) Wiele danych medycznych jest wyrażanych za pomocą obrazów lub innych sygnałów. Algorytmy ewolucyjne znajdują tu zastosowanie do poprawiania działania algorytmów przetwarzających sygnały (np. filtrów lub kompresorów) przez znalezienie ich optymalnych parametrów. Mogą też zostać wykorzystane do bezpośredniego wyprowadzenia użytecznej informacji z dostarczonych danych. Algorytmy ewolucyjne szczególnie dobrze nadają się do rozwiązywania problemów planowania i harmonogramowania (c). Przykładem może tu być problem harmonogramowania dla pacjenta, poddawanego różnym medycznym procedurom i potrzebującego konsultacji różnych specjalistów, w celu optymalizacji zarówno czasu oczekiwania pacjenta jak również wykorzystania aparatury [13].

6. ZAKOŃCZENIE

Algorytmy ewolucyjne są obecnie z powodzeniem stosowane w rozwiązywaniu wielu rzeczywistych problemów w różnych dziedzinach. Wydaje się, że w przyszłości znaczenie technik opartych na algorytmach ewolucyjnych będzie wzrastać, jako że pozwalają one na pokonanie wielu trudności pojawiających się przy stosowaniu klasycznych metod. Dlatego też należy oczekiwać, że aplikacje bazujące na zasadach ewolucyjnych osiągać będą coraz większą popularność.

Literatura

1. G. E. Carrillo-Ureta, P. D. Robrts, V. M. Becerra, Genetic algorithm for optimal control of beer fermentation, Proc. of IEEE Intern. Symp. On Intelligent Control, Mexico City, 391-396, 2001.
2. K. Deb, Multi-objective Optimization Using Evolutionary Algorithms, Wiley, 2001.
3. U.M. Foyyad, G. Piatetzky-Shapiro, P. Smyth, R. Uthurusamy, Advances in Knowledge Discovery and Data Minning, Cambridge, MA: AIII Press/MIT Press, 1996.
4. L. J. Fogel, A. J. Owens, M. J. Walsh, Artificial Intelligence through Simulated Evolution, John Wiley, New York, 1966.
5. D. B. Fogel, Evolutionary Computing: The Fossile Record, IEEE Press, Piscataway, NJ, 1998.
6. D. E. Goldberg, Algorytmy genetyczne i ich zastosowania, WNT 1995.
7. S. D.Harris, L.Elliot, D. B. Ingram, M. Pourkashanian, C. W. Wilson, The optimization of reaction rate parameters for chemical kinetic modeling of combustion using genetic algorithms, Comput. Meth. Appl. Mech. Eng. 190, 1065-1090, 2000.
8. J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
9. K. E. Kinnear, Advances in Genetic Programming, MIT Press, Cambridge, Mam 1994.
10. J. R. Koza, Genetic Programming, MIT Press, 1992.
11. V. Oduguwa, A. Tiwari, R. Roy, Evolutionary computing in manufacturing industry: an overview of recent applications, Applied Soft Computing 5, 281-299, 2005.
12. C. A. Peňa-Reyes, M. Sipper, A fuzzy-genetic approach to breast cancer diagnosis, Artif. Itell. Med. 17, 131-135, 1999.
13. C. A. Peňa-Reyes, M. Sipper, Evolutionary computation in medicine: an overview, Artif. Itell. Med. 19, 1-23, 2000.
14. R echenberg, Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution, Fromman-Hozlboog Verlag, Stuttgart, 1973.
15. H. P. Schwefel, Numerical Optimization of Computer Models, John Wiley & Sons, New York, 1981.
16. L. A. Wolsey, Integer Programming, John Wiley & Sons Inc. 1998.
17. Y. Yu, M. C. Schell, J. B. Hang, Decision theoretic steering and genetic algorithm optimization: application to stereotactic radiosurgery treatment planning, Med. Phys 24, 1742-1750, 1997.