SOKÓŁ SESJA OKOLICZNOŚCIOWA
PSKN zeszyt nr 33/2014
Gliwice 2014
INSTYTUT MATERIAŁÓW INŻYNIERSKICH I BIOMEDYCZNYCH
POLITECHNIKI ŚLĄSKIEJ W GLIWICACH
PRACE STUDENCKICH KÓŁ NAUKOWYCH

Aproksymacja funkcji wpływu węgla na twardość stali za pomocą metody programowania genetycznego

P. Papliński[a], W. Sitek[b]

[a] Student Politechniki Śląskiej, Wydział Mechaniczny Technologiczny
email: paplinski@outlook.com
[b] Politechnika Śląska, Wydział Mechaniczny Technologiczny, Instytut Materiałów Inżynierskich i Biomedycznych, Zakład Inżynierii Materiałów Biomedycznych
email: wojciech.sitek@polsl.pl

Streszczenie: W artykule, używając programowania genetycznego, przedstawiono, jak metody sztucznej inteligencji wspomagają rozwiązywanie problemów inżynierii materiałowej. Na przykładzie modelowania funkcji pokazującej wpływ zawartości węgla na twardość stali pokazano jak nowoczesne metody sztucznej inteligencji mogą być w łatwy sposób zaadaptowane do rozwiązywania problemów nowoczesnej nauki.

Abstract: In the paper it was presented how, using genetic programming, artificial intelligence methods can be used to solve the problems of materials science. On the example of modelling a function showing how carbon concentration in steel changes its hardness it was shown how modern artificial intelligence methods can easily be adapted for solving problems of modern science.

Słowa kluczowe: programowanie genetyczne, sztuczna inteligencja, inżynieria materiałowa

1. WPROWADZENIE

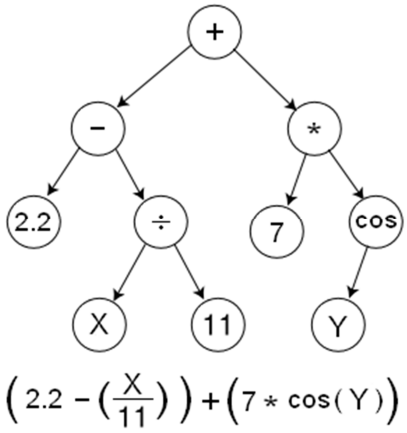
Węgiel jest pierwiastkiem mającym największy wpływ na twardość stali, dlatego został wybrany jako przykład do modelowania funkcji jej twardości.

Programowanie genetyczne jest metodą sztucznej inteligencji, która ewoluowała z algorytmów genetycznych, zaproponowanych przez J. Holland'a i potem rozwijanych przez R. Koze, D.E. Goldberg'a i innych [1,3]. Metoda ta naśladuje procesy występujące w naturalnej ewolucji. Algorytm metody jest bardzo nietypowy w porównaniu z klasycznymi technikami programowania [1]. Zwykle, programista tworzy program, który działając według pewnego spersonalizowanego algorytmu dochodzi do rozwiązania. W programowaniu genetycznym, program główny tworzy populację mniejszych programów, które ewoluują i mutują, dążąc do osiągnięcia jak najlepszego przystosowania do szukanej funkcji, z tego też względu nie jest potrzebny specjalny algorytm.

Algorytmy genetyczne są głównie metodą optymalizacji, programowanie genetyczne idzie dalej, pozwalając znaleźć funkcję jak najbardziej zbliżoną do wczytanego zestawu danych. Metoda ta jest również wolna od większości ograniczeń, które są największym problemem

innych metod aproksymacji. W przypadku programowania genetycznego możemy mówić o tylko jednym wymogu – wczytany zestaw danych musi odzwierciedlać wszystkie znaczące zmiany w szukanym fragmencie wykresu szukanej funkcji, w przeciwnym wypadku możemy nigdy nie otrzymać satysfakcjonujących rezultatów. Metoda sama w sobie zależy również w pewnym stopniu od prawdopodobieństwa, więc w przypadku złego przejścia algorytmu, może on nigdy nie wrócić na trop poszukiwania optymalnego rozwiązania i musi zostać przerwany [3]. Jest to bardzo rzadkie, lecz zawsze pozostaje pewne ryzyko, dlatego pojedynczy przebieg algorytmu nie może dać nam pewnej odpowiedzi. Jeżeli błąd nie jest równy 0, zawsze jest szansa na odnalezienie lepszej odpowiedzi.

Podstawowym elementem metody jest, tak jak i w algorytmie genetycznym, populacja. W przypadku programowania genetycznego najczęściej stosowaną reprezentacją pojedynczego osobnika jest struktura drzewiasta (rys. 1). Jest ona często stosowana głównie ze względu na łatwość jej krzyżowania i mutacji. Wymusza ona również kolejność przechodzenia przez węzły i liście. Przykładowy program może wyglądać następująco:



Rysunek 1. Funkcja w reprezentacji drzewiastej
Figure 1. Function represented as a tree structure

Jest to reprezentacja o zmiennej długości, ponieważ osobnikami mogą tutaj być zarówno programy sumujące dwie liczby, jak i programy obliczające pierwiastki równania kwadratowego. Wynikiem działania każdego programu jest pewna wartość. Aby wyznaczyć wartość programu reprezentowanego w strukturze drzewiastej, należy przed wykonaniem zadania zapisanego w węźle najpierw obliczyć wyrażenia „zaczepione" w jego potomkach.

Algorytm programowania genetycznego rozpoczyna się od wygenerowania populacji losowych osobników. Pierwszym krokiem jest reprodukcja, odbywa się ona poprzez wybór osobników z populacji rodzicielskiej za pomocą jednej z metod selekcji a następnie wybranie losowo położonego punktu krzyżowania i wymianę kodu pomiędzy osobnikami. Sama procedura selekcji wygląda następująco:

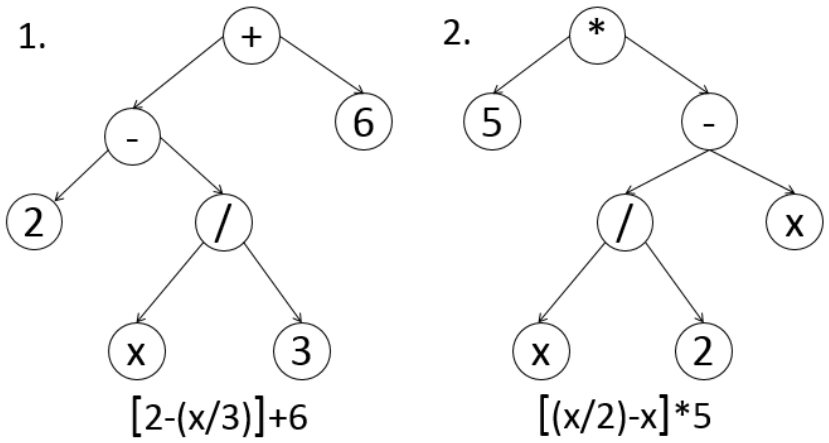
1. funkcja przystosowania (fenotyp) jest obliczana dla każdego osobnika. Następnie jest ona normalizowana, co oznacza podzielenie jej przez sumę wartości przystosowania pozostałych osobników. Suma znormalizowanych funkcji przystosowania powinna wynosić 1;
2. populacja jest sortowana malejąco według funkcji przystosowania;
3. dla każdego osobnika przygotowana zostaje zakumulowana wartość przystosowania, oznaczająca sumę jego własnego przystosowania i funkcji przystosowania wszystkich poprzednich osobników. Zakumulowana funkcja przystosowania ostatniego osobnika powinna wynosić 1;
4. wybrany jest losowy numer od 0 do 1;

5. wybrany zostaje pierwszy osobnik, którego zakumulowana wartość przystosowania jest większa niż wylosowana wartość.

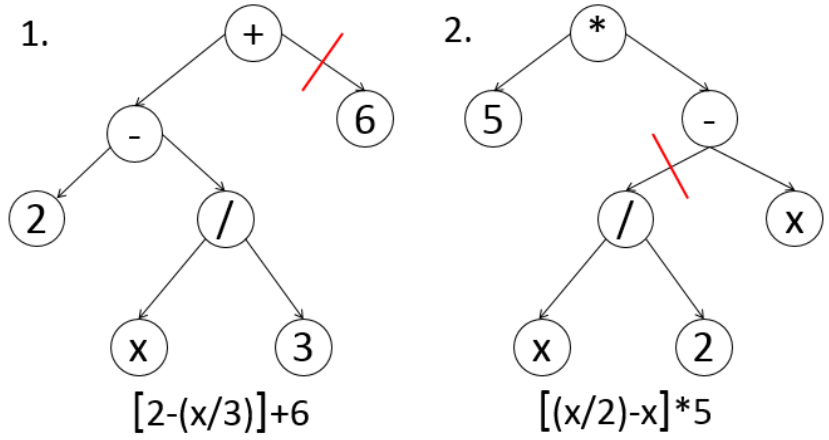
Jeżeli ta procedura jest powtarzana, aż do zapełnienia nowej populacji, mamy do czynienia z selekcją proporcjonalną według przystosowania albo selekcją metodą „Ruletki". Niestety ewolucja przy takim algorytmie z każdym krokiem zwalnia. Jeżeli osobniki są podobne, to każdy dostaje równy wycinek koła fortuny i presja selekcyjna spada. Algorytm słabiej rozróżnia osobniki dobre od słabszych. Pozbawiona tej wady jest metoda rankingowa. Obliczamy dla każdego osobnika funkcję oceny i ustawiamy je w szeregu najlepszy-najgorszy. Pierwsi na liście dostają prawo do rozmnażania, a reszta usuwana jest z populacji. Wadą metody jest niewrażliwość na różnice pomiędzy kolejnymi osobnikami w kolejce. Może się okazać, że sąsiadujące na liście rozwiązania mają różne wartości funkcji oceny, ale dostają prawie taką samą ilość potomstwa.

Operacje genetyczne rozpoczyna się od krzyżowania. Aby tego dokonać, należy losowo wybrać dwóch osobników z nowej populacji (rys. 2).

Następnie, na gałęziach struktury należy wybrać losowe punkty krzyżowania (rys. 3).

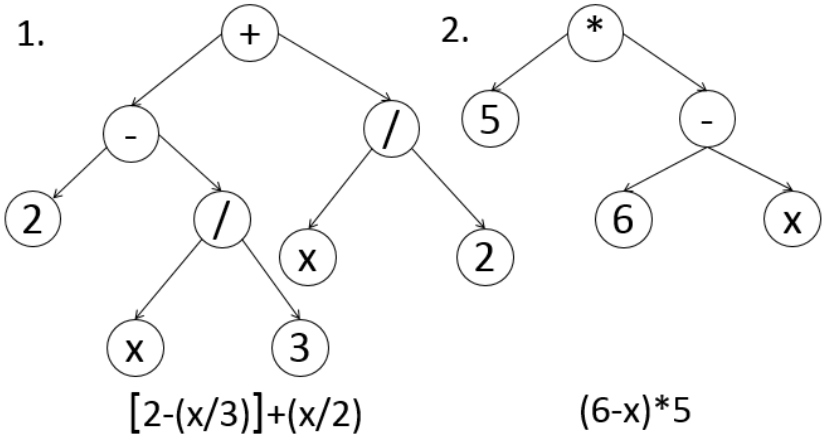


Rysunek 2. Osobniki wybrane do krzyżowania
Figure 2. Individuals chosen for crossover



Rysunek 3. Punkty krzyżowania wybrane na osobnikach
Figure 3. Crossover points chosen on the individuals

Wszystkie zmienne i operatory znajdujące się poniżej punktów krzyżowania zostają zamienione pomiędzy osobnikami (rys. 4).



Rysunek 4. Osobniki po krzyżowaniu
Figure 4. Individuals after crossover

Po krzyżowaniu może, lecz nie musi, nastąpić mutacja, która polega na losowej zamianie wartości umieszczonej na węźle lub liściu drzewa na inna.

Na koniec każdej sekcji operacji genetycznych należy sprawdzić przystosowanie wszystkich osobników. Jeżeli znajdziemy osobnika o przystosowaniu przynajmniej tak dobrym, jakie zakładano, algorytm zatrzymuje swoje działanie, w przeciwnym wypadku populacja potomna zamienia się w populację rodzicielską i algorytm jest powtarzany.

2. APROKSYMACJA FUNKCJI WPŁYWU WĘGLA NA TWARDOŚĆ STALI ZA POMOCĄ PROGRAMOWANIA GENETYCZNEGO

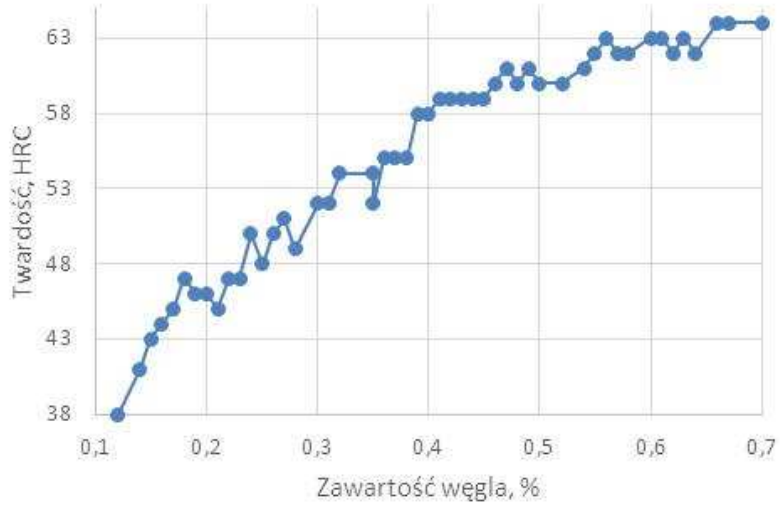
Problem został rozwiązany poprzez zaimplementowanie metody programowania genetycznego w programie komputerowym przetwarzającym zestaw danych, zawierający twardości 50 próbek utwardzonej stali [2]. Twardość próbek mieści się w przedziale od 38 do 64 HRC natomiast zawartość procentowa węgla w próbkach wynosi od 0,12% do 0,7%. Prawdopodobieństwo reprodukcji zostało ustalone na poziomie 75%, prawdopodobieństwo wystąpienia mutacji na 10%, populacja składała się ze 100 programów, liczba iteracji algorytmu została ograniczona do 1000. Użyto dwóch zestawów instrukcji, podstawowego – zawierającego tylko podstawowe operatory takie jak +, -, *, /, oraz rozszerzonego – dodatkowo używającego ln, sin, cos, tg, ctg. Wykorzystano również trzy metody selekcji – Najlepszy, Ranga i Ruletka.

Algorytm był testowany na przygotowanym zestawie danych (rys. 5).

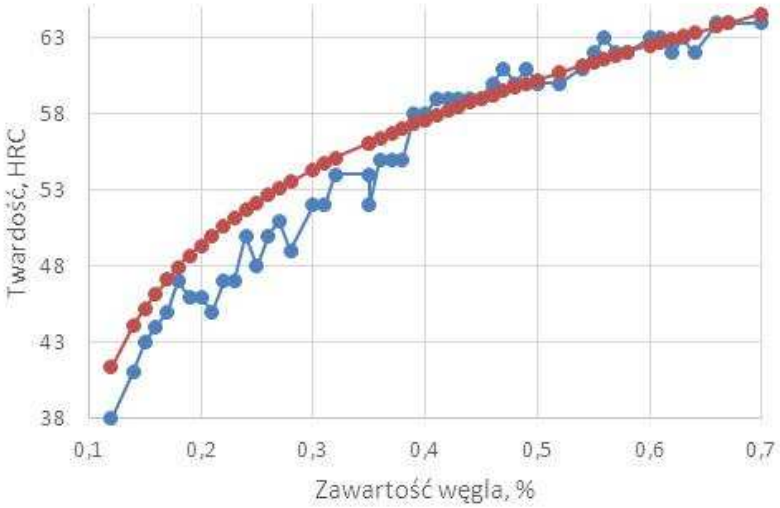
Pierwszy przebieg programu został wykonany dla metody selekcji “Najlepszy” i podstawowego zestawu operatorów. Przystosowanie odnalezionej funkcji (wzór 1) jest bardzo obiecujące (rys. 6). Średni błąd funkcji w porównaniu z danymi źródłowymi wynosi jedynie 1,0122 HRC.

$$f(x) = \left(\left((3+x)\times 2\right)\times 5 - \left(\frac{4}{x\times 2} - \left(x\times 5 - (x-5)\right)\right)\right) - \left(x\left(-\frac{4}{2} - (1-x)\right) - 5\right) + 16$$

Wzór 1. Wzór pierwszej znalezionej funkcji
Equation 1. Equation of first found function



Rysunek 5. Zestaw danych testowych
Figure 5. Set of test data

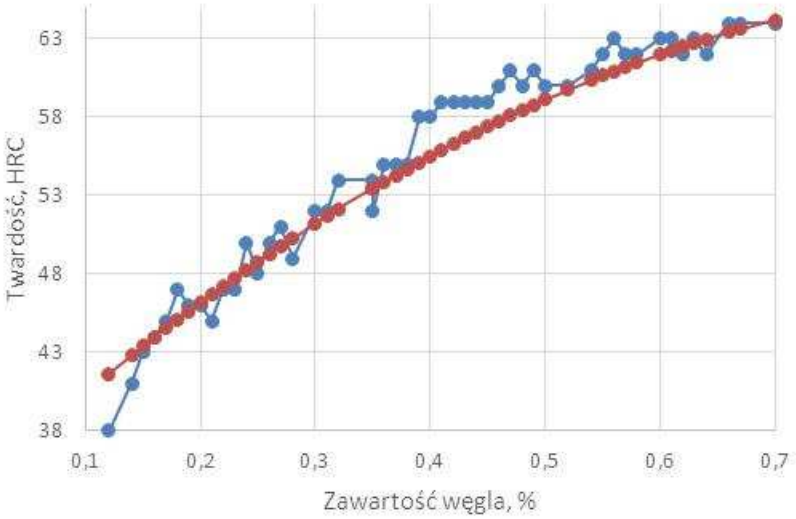


Rysunek 6. Przystosowanie pierwszej znalezionej funkcji
Figure 6. Fitness of first found function

Drugi przebieg programu został wykonany dla metody selekcji “Ranga” i podstawowego zestawu operatorów. Przystosowanie odnalezionej funkcji (wzór 2) wskazuje, że metoda selekcji „Najlepszy” była optymalna dla tego zestawu danych (rys. 7). Średni błąd funkcji w porównaniu z danymi źródłowymi wynosi 1,0788 HRC, jednak wykres wskazuje na to, że w szerszym przedziale funkcja będzie coraz gorzej przystosowana.

$$f(x) = -5x^3 + \left(\frac{7}{2} - x\right)(16 - x)x + 2(x - 3)(2x - 2)x + 34$$

Wzór 2. Wzór drugiej znalezionej funkcji
Equation 2. Equation of second found function

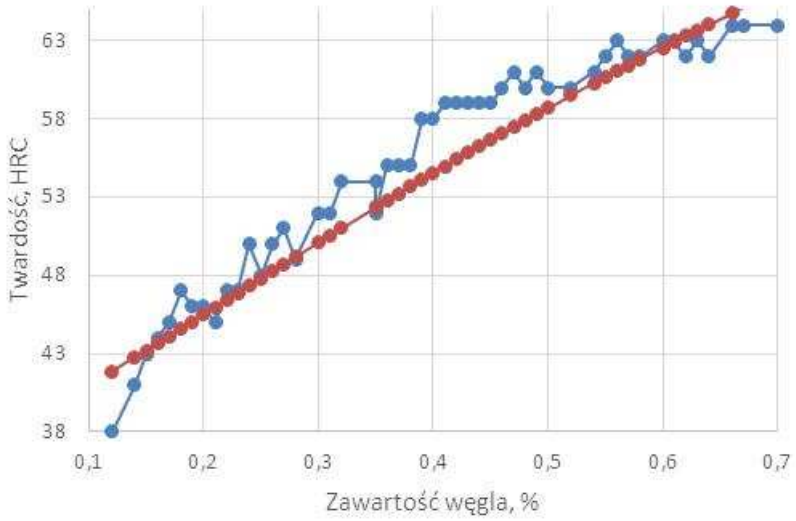


Rysunek 7. Przystosowanie drugiej znalezionej funkcji
Figure 7. Fitness of second found function

Trzeci przebieg programu został wykonany dla metody selekcji “Ranga” i rozszerzonego zestawu operatorów. Przystosowanie odnalezionej funkcji (wzór 3) pokazuje, że metoda selekcji „Ranga”, daje rezultaty, które nie mogą być zaakceptowane (rys. 8). Średni błąd funkcji w porównaniu z danymi źródłowymi wynosi 1,30106 HRC, a wykres wskazuje na to, że w szerszym przedziale funkcja będzie coraz gorzej przystosowana.

$$f(x) = (x+2)(x+4) + \frac{5}{4}(5x + \sin(4)) + (x+2)(\sin(x) + 25) + \sin(6x+4) - 20$$

Wzór 3. Wzór trzeciej znalezionej funkcji
Equation 3. Equation of third found function

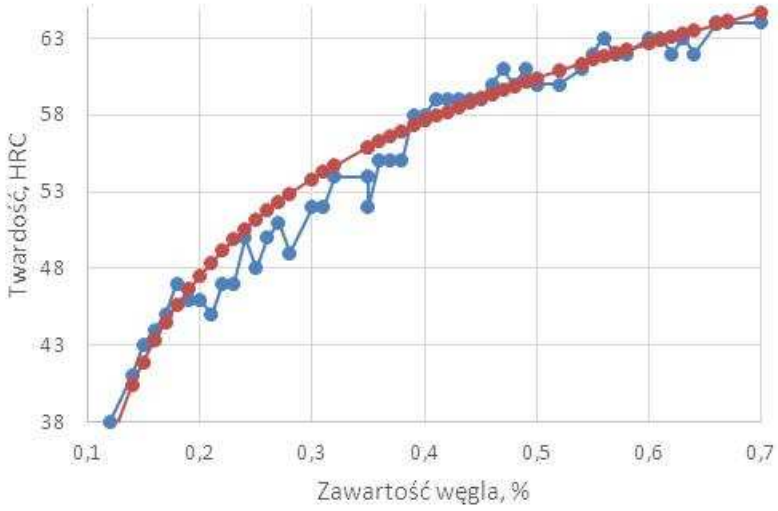


Rysunek 8. Przystosowanie trzeciej znalezionej funkcji
Figure 8. Fitness of third found function

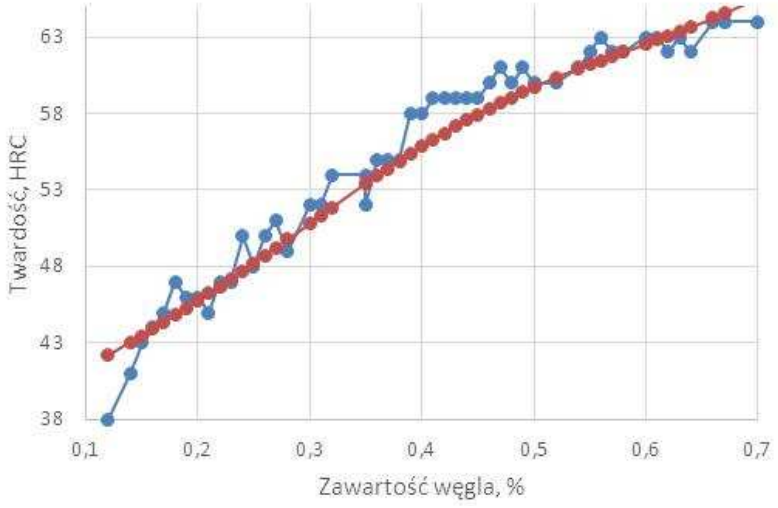
Czwarty przebieg programu został wykonany dla metody selekcji “Ruletka” i podstawowego zestawu operatorów. Przystosowanie odnalezionej funkcji (wzór 4) pokazuje, że selekcja metodą „Ruletka” pozwoliła odzwierciedlić bardzo ważne załamanie na wykresie funkcji (rys. 9). Średni błąd funkcji w porównaniu z danymi źródłowymi wynosi 1,104 HRC, ale pomimo błędu większego niż wcześniej, analizując wykres przystosowania, można wywnioskować, że ta metoda może dać bardzo dobre rezultaty dla tego zestawu danych.

$$f(x) = 60 - \frac{3}{x} + \frac{64x}{5}$$

Wzór 4. Wzór czwartej znalezionej funkcji
Equation 4. Equation of fourth found function



Rysunek 9. Przystosowanie czwartej znalezionej funkcji
Figure 9. Fitness of fourth found function



Rysunek 10. Przystosowanie piątej znalezionej funkcji
Figure 10. Fitness of fifth found function

Piąty przebieg programu został wykonany dla metody selekcji “Ruletka” i rozszerzonego zestawu operatorów. Przystosowanie odnalezionej funkcji (wzór 5) pokazuje, że podstawowy zestaw operatorów dał lepsze rezultaty od zestawu rozszerzonego (rys. 10). Średni błąd funkcji w porównaniu z danymi źródłowymi wynosi tylko 0,87865 HRC, ale pomimo błędu mniejszego niż wcześniej, funkcja przystosowania sugeruje, że w szerszym spektrum przystosowanie może być gorsze.

$$f(x) = 20x + log(x) + (2x + 5)(\log(x) + 7) - \log(2x) + \cos(12x + \cos(x)) + 9$$

Wzór 5. Wzór piątej znalezionej funkcji
Equation 5. Equation of fifth found function

3. WNIOSKI

Wyniki otrzymane jako rezultat aproksymacji są bardzo obiecujące, jeżeli wziąć pod uwagę, że zbiór danych weryfikujących zawierał jedynie węgiel i ignorował pozostałe pierwiastki stopowe, które również mają wpływ na twardość stali. Programowanie genetyczne okazało się bardzo przydatną metodą do rozwiązywania tego typu problemów. Wyniki pozwalają przypuszczać, że ta metoda może przyczynić się do rozwiązania wielu problemów inżynierii materiałowej, szczególnie dzięki swojej uniwersalności. Programowanie genetyczne stanowi również dobrą alternatywę dla innych metod aproksymacji, częściej stosowanych w przypadku rozwiązywania problemów inżynierii materiałowej. Największą wadą metody jest jednak trudność w implementacji, ponieważ nie istnieją uniwersalne narzędzia pozwalające na jej zastosowanie, a sama implementacja wymaga dogłębnej znajomości metody.

LITERATURA

1. J. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, The MIT Press, Cambridge, Massachusetts, 1992.
2. W. Sitek, Modelowanie zależności między składem chemicznym i hartownością stali konstrukcyjnych stopowych, rozprawa doktorska, Biblioteka Główna Politechniki Śląskiej, Gliwice, 1997.
3. D.E. Goldberg, Algorytmy genetyczne i ich zastosowania, WNT, Warszawa, 2003.