Informatyka, studia dzienne, mgr jednolite semestr IX

Obliczenia ewolucyjne **2010/2011**

Prowadzący: mgr inż. Łukasz Chomątek wtorek, 14:30

Data oddania: _____ Ocena: _____

Cezar Pokorski 138077

Artur Czajka 137971

Zadanie 4: Całkowanie przez GP*

1. Istota zadania

Dla zadanej całki nieoznaczonej należy ewolucyjnie wygenerować formułę symboliczną reprezentującą funkcję pierwotną metodami programowania genetycznego.

2. Implementacja

Do rozwiązania tego zadania posłużyliśmy się językiem Python oraz pakietami `matplotlib` i `numpy`.

2.1. Reprezentacja

Jako podstawę rozwiązania utworzyliśmy klasę `Node` — jej obiekty reprezentują elementy struktury drzewiastej, która z kolei stanowi reprezentację formuły. `Node` może być wartością stałą, operatorem jedno lub dwuargumentowym.

Poszukujemy wzoru punkcji pierwotnej dla zadanej funkcji podcałkowej. Oznacza to, że dla każdego ewoluowanego wzoru obliczamy pochodną metodą ilorazu różnicowego. Obliczamy te wartości w 21 punktach kontrolnych, co 5% wartości przedizału. Następnie porównujemy otrzymane wartości z wektorem wartości funkcji podcałkowej. *Błąd* danego osobnika określa się na podstawie sumy wartości absolutnych powstałych różnic.

* HG: `https://aelabnull.googlecode.com/hg/4/`

2.2. Ewolucja

Nową populację tworzą pary osobników pochodzące z krzyżowania, które polega na podmianie dwóch, losowo wybranych poddrzew w kopiach rodziców.

W każdej epoce jest pewna szansa, domyślnie $\frac{3}{5}$, że osobnik zostanie wybrany do rozrodu. Największą szansę na przekazanie genów mają osobniki z najmniejszym *błędem*. Jeśli osobnik nie zostanie wybrany do rozrodu, ma pewną szansę, domyślnie $\frac{1}{5}$, na przebycie mutacji, która jednak jest maskowana przed współplemieńcami. Proces wyboru powtarza się, dopóki nowa populacja nie osiągnie rozmiarów starej.

Operator mutacji powoduje losową zmianę w danym osobniku. W $\frac{1}{3}$ przypadków zmianie ulega sama wartość danego węzła, operator lub stała. Następna $\frac{1}{3}$ przypadków to rekurencyjna mutacja jednego z podwęzłów osobnika. Pozostała część to podmiana całej gałęzi na nową, losową. Jeśli dany węzeł nie posiada węzłów potomnych — jest szansa 50/50 na wybór jednego z pozostałych wariantów.

3. Wyniki i wnioski

Końcowym rezultatem działania algorytmu jest wzór funkcji będącej przybliżeniem funkcji pochodnej funkcji wejściowej. Opcjonalnie rysowany jest wykres ukazujący obie funkcje. W przypadku idealnym czerwona krzywa jest wykresem funkcji pochodnej funkcji zaznaczonej kolorem niebieskim.

Wzór najlepszej funkcji można również obejrzeć podczas procesu ewolucji, co 20 iteracji. Jest prezentowana w notacji polskiej, z nadmiarowymi nawiasami, znanymi m. in. z języka LISP.

Dla porównania otrzymanych przez nas wyników ze stanem faktycznym posługiwaliśmy się Wolframem Alpha[1].

Za każdym przebiegiem algorytmu zauważamy, że:

— Podobnie jak w zadaniach prostej maksymalizacji, wraz z rosnącą ilością przebytych iteracji, maleje różnorodność populacji. Osobniki reprezentujące najlepsze dotąd rozwiązanie mają tendencję do przeżywania i namnażania się w niemal niezmienionej postaci w wielu kopiach (po np. 200 iteracjach). Wynika to z zastosowanej modyfikacji algorytmu, w której po wybraniu osobników do krzyżowania, przechodzimy po populacji raz jeszcze od najlepszego algorytmu. Różnorodność w tym wariancie zapewniają niższe wartości prawdopodobieństwa krzyżowania, jakość osobników zaś — wybór najlepszych z niewielkiej ich liczby. Metoda ta okazuje się bowiem dawać ciekawsze (z punktu widzenia dążenia do rozwiązania, oczywiście) wyniki dla niewielkich populacji (domyślnie mamy 30 osobników).
— Dla niektórych funkcji, zwłaszcza na odpowiednio małym badanym przedziale, algorytm preferuje drzewa reprezentujące tak naprawdę jedynie wartości stałe, w przybliżeniu zgodne z oczekiwanym wynikiem. Zwykle jednak poszerzenie zakresu, na którym wartości są kontrolowane powoduje wymuszenie uwzględnienia wreszcie zmiennej x.

[1] `http://http://www.wolframalpha.com/`

— W zadaniu tym zmniejszenie pradopodobieństwa rozrodu pomaga zachować różnorodność genetyczną, która z kolei pozwala procesowi ewolucji wyjść z lokalnych minimów błędów. W praktyce przyjęte przez nas wartości domyślne pomagają uzyskać satysfakcjonujące wyniki.
— Ponieważ w programie w celu ułatwienia implementacji drzew zawierających dowolne funkcje matematyczne, jak i przeróżnych (dowolnych dopuszczalnych) operatorów matematycznych, korzystamy ochoczo ze skryptowości (jako cechy) języka python, a wartości wyrażeń obliczamy przy użyciu funkcji **eval**, może to tworzyć spory narzut wydajnościowy. Dlatego też taki sposób realizacji powoduje niezbyt szybkie wykonywanie algorytmu dla złożonych działań, dużych populacji czy też rozbudowanych drzew. Pamiętać należy także, że drzewa nie dokonują żadnej optymalizacji (np. jeśli duża gałąź ewaluuje się zawsze do tej samej wartości), a wręcz byłoby to niewskazane (z punktu widzenia potencjalnej mutacji i krzyżowania).