Koncepcja, implementacja i testowanie algorytmu ewolucji różnicowej, w którym selekcja odbywa się probabilistycznie, podobnie jak w algorytmie symulowanego wyżarzania

Michał Bobowski, Marcin Cieślikowski

2013-12-12

1 Opis algorytmu

Poniżej znajduje się opis algorytmu ewolucji różnicowej, który został zaimplementowany w ramach projektu.
Zdefiniowane zostają następujące operacje:

- select(P) – wybór punktu z populacji.
- select_pair(P) – wybór pary punktów z populacji.
- crossover(x1,x2) – krzyżowanie punktów.

Dopóki nie osiągnięto warunku stopu:
    Dla każdego elementu xi z populacji P:
      xj = select(P)
      (xk, xl) = select_pair(P)
      y = xj + F(xk, xl)
      z = crossover(xi,y)
      xi = tournament(xi,z)

1.1 Alternatywne modele selekcji

Głównym polem do eksperymentów było wprowadzenie alternatywnych modeli selekcji, których idea była podobna do algorytmu symulowanego wyżarzania. Początkowo algorytm działa w warunkach zbliżonych do wersji

bazowej, ale z czasem wprowadzana jest presja na wybór lepszych punktów. Dzięki temu następuje przejście z fazy eksploracji do fazy eksploatacji. Technicznie zastosowano dwa podejścia opisane poniżej.

1.1.1 Selekcja medianowa

W modelu przechowywana jest wartość mediany funkcji celu dla punktów z populacji, aktualizowana po każdym przebiegu pętli głównej. Jeżeli wylosowany punkt ma lepszą wartość funkcji celu niż mediana, to jest akceptowany. W przeciwnym przypadku prawdopodobieństwo akceptacji jest przybliżone rozkładem o liniowej funkcji gęstości:

$pdf(i) =\mid \frac{max-i}{max} - 0.2 \mid$

Gdzie:

- i - numer bieżącej iteracji.
- max - liczba wszystkich iteracji.

1.1.2 Selekcja progowa

W modelu przechowywana jest posortowana według wartości funkcji celu populacja, aktualizowana po każdym przebiegu pętli głównej. Na podstawie populacji ustalany jest próg wartości funkcji celu, od którego akceptowane są punkty. Jest on równy wartości i-tego punktu (począwszy od najgorszego), przy czym i może maksymalnie wynieść wartość o 5 niższą od rozmiaru populacji. Wylosowanie punktu gorszego od progu skutkuje powtórzeniem losowania.

2 Kod źródłowy

Całość źródeł jest dostępna w repozytorium pod adresem: https://github.com/mbobowsk/meum

2.1 Konfiguracja środowiska

Kod źródłowy został napisany w języku R (wersja 2.11.1). Dodatkowo zostały wykorzystane dwie biblioteki:

- cec2005benchmark w wersji 1.0.3.
- Hmisc w wersji 3-10.1 (najnowsza wersja powoduje błąd).

2.2 Opis plików

Głównym plikiem źródłowym jest *main.r*, do którego załączane są wszystkie pozostałe pliki. Do przeprowadzenie symulacji potrzebne jest uruchomienie funkcji *test*, której argumentami są kolejno:

- Liczba wymiarów - powinna wynosić 2, 10, 30 lub 50.
- Liczba niezależnych uruchomień algorytmu
- Numer funkcji z benchmarku CEC 2005 - obsługiwane są funkcje 1,4 oraz 9.

Wynikiem działania funkcji jest wykres dystrybuanty empirycznej dla trzech metod selekcji. Należy pamiętać o ręcznej zmianie ścieżki bezwzględnej do pliku wynikowego.

Przebieg algorytmu ewolucji różnicowej znajduje się w pliku *de.r*. Funkcje przeprowadzające selekcję z populacji znajdują się w *selection.r*. Pliki *cec.r* i *config.r* zawierają parametry takie jak rozmiar populacji czy liczba iteracji w ramach uruchomienia.

3 Testy

Celem testów było porównanie działania różnych metod selekcji dla różniących się poziomem trudności funkcji celu. Szczególnie interesujące było to, czy bardziej skomplikowane modele selekcji znacząco przyspieszą szybkość dochodzenia do optymalnego rozwiązania.

Dla każdej funkcji celu zostały przeprowadzone eksperymenty wstępne, które pozwalały ustawić liczbę iteracji na odpowiednim poziomie. Przy odpowiednio dużej liczbie wszystkie algorytmy dochodziły do podobnych rozwiązań, a przy zbyt małej nie można było zaobserwować różnicy.

W porównaniu z podstawową wersją benchmarku CEC 2005 zamieniony został znak funkcji celu (rozwiązywany jest problem maksymalizacji, a nie minimalizacji). Pozwala to na narysowanie dystrybuanty empirycznej.

3.1 Parametry środowiska testowego

Aby ograniczyć liczbę zmiennych, przyjęto następujące założenia:

- Rozmiar populacji wynosił 20.
- Problemy były rozwiązywane w kostce o rozmiarach od -100 do 100 (dla f1 i f4) lub od -30 do 30 (dla f9).

- Liczba wymiarów wynosiła 50 (dla f1 i f4) lub 10 (dla f9).
- Liczba iteracji była każdorazowo dobierana doświadczalnie.

3.2 Funkcja f1

3.3 Funkcja f4

3.4 Funkcja f9

4 Wnioski

Liczba przeprowadzonych testów nie jest wystarczająca do formułowania całościowych wniosków na temat badanych metod selekcji. Można niemniej zauważyć, że dla funkcji prostych takich jak f1 i f4 zaobserwowano widoczną poprawę szybkości dochodzenia do minimum lokalnego. Z kolei funkcja f9 nie pokazała żadnych różnic pomiędzy metodami selekcji, gdyż posiadała wiele lokalnych minimów.