Biblioteka obsługi algorytmów genetycznych (evol)

Zaawansowane Programowanie

Autorzy: Andrzej Fiedukowicz i Maciej Grzybek

Specyfikacja wstępna.

Opis zagadnienia

Treść zadania:

Zdefiniować zestaw klas służący do implementacji algorytmów ewolucyjnych. Powinna istnieć łatwa możliwość zmiany funkcji oceny, funkcji odpowiadającej za mutację itd. W oparciu o zaproponowane klasy stworzyć dwie wersje algorytmu ewolucyjnego - jedną standardową, natomiast w drugiej wersji osobniki powinny być oceniane parami (osobnik i łączy się z osobnikiem j na n generacji. Jakość każdego z osobników (składników pary) jest funkcją jakości całej pary (jakości osobnika j i osobnika i).

Opis zadania:

Celem projektu jest zrealizowanie biblioteki upraszczającej tworzenie wszelakich algorytmów ewolucyjnych. Głównym zadaniem projektu jest wyizolowanie powtarzającej się części problemów dotyczących wszystkich lub znacznej większości algorytmów genetycznych przy możliwym odcięciu od konkretnych problemów.

Biblioteka evol będzie dostarczała zestawu klas bazowych, których rozszerzenie w większym lub mniejszym zakresie pozwoli zrealizować niemal dowolny algorytm genetyczny, tymniemniej dla zagadnień bardzo złożonych i nietypowych wykorzystanie tej biblioteki może mijać się z celem, ze względu na konieczność nadpisania niemal wszystkich metod klas bazowych. W związku z tym biblioteka jest dedykowana przede wszystkim dla algorytmów o małym lub średnim poziomie złożoności i nie odbiegających w bardzo znaczący sposób od typowych algorytmów genetycznych.

Analiza zagadnienia:

Analiza projektu wymaga przede wszystkim precyzyjnego zdefiniowania czym jest i z jakich stałych elementów składa się algorytm genetyczny. Poniżej przeprowadzono analizę pojęciową.

- *Populacja* – zbiór osobników, które są w danym momencie żywe i mogą zostać poddane reprodukcji albo selekcji.
- *Osobnik* – genotyp + fenotyp, poddawany krzyżowaniu z innymi osobnikami lub mutacji i posiadający pewną wartość funkcji celu (jedyny wymagany fragment fenotypu).
- *Chromosom* – opis określonej cechy określającej pewną właściwość osobnika, posiadającej pewną wartość i zdolność mutacji oraz krzyżowania z innymi chromosomami tego samego rodzaju.
- *Mutacja* – zdefiniowana (typowo - losowa), niewielka zmiana wartości chromosomu (bądź szerzej, genotypu).
- *Krzyżowanie* – operacja na zbiorze chromosomów opisujących tę samą cechę (bądź szerzej, zbiorze genotypów) w wyniku której powstaje nowy chromosom (genotyp).
- *Selekcja* – wybór spośród populacji osobników dających najbliższą oczekiwanej wartość funkcji celu.
- *Funkcja celu* – funkcja określająca jak dobrym rozwiązaniem problemu jest dany osobnik.
- *Rozwiązanie algorytmu* – zbiór chromosomów (genotyp) dający oczekiwaną wartość funkcji celu.
- *Liczebność populacji* – bieżąca liczebność populacji.

- *Pojemność populacji* – zadana liczebność populacji przed fazą reprodukcji.
- *Reprodukcja* – rozszrzenie zbioru osobników w populacji o nowe powstałe w wyniku operacji krzyżowania i mutacji.
- *Genotyp* – zbiór chromosomów opisujących osobnika.
- *Fenotyp* – pochodna genotypu, opisująca cechy „zewnętrzne" osobnika, stanowiący punkt wyjścia oceny osobnika.

Typowy scenariusz działania algorytmu:

1. Wylosowanie populacji początkowej
2. Losowe krzyżowanie osobników bieżącej populacji (domyślnie krzyżowanie kolejnych chromosomów par wylosowanych osobników).
3. Losowe mutacje osobników bieżącej populacji (domyślnie mutacje kolejnych chromosomów wylosowanych osobników).
4. Selekcja (domyślnie usunięcie wszystkich ((Liczebność populacji)-(Pojemność populacji)) najsłabszych (pod względem wartości funkcji celu) osobników).
5. Ocena najlepszego osobnika – czy spełnia zadany warunek nałożony na wartość funkcji celu.
   - Tak – koniec algorytmu, podanie wyniku działania algorytmu.
   - Nie – wróć do punktu 2.

Wymagania implementacyjne dla programisty:

By stworzyć (przy użyciu bliblioteki evol) w pełni funkcjonalny algortym genetyczny programista musi zdefiniować:

1. Funckję celu (wraz z operatorem porównania wartości tej funkcji, oraz zadaną wartością).
2. Metodę wyboru populacji początkowej.
3. Osobnika z co najmniej jednym chromosomem.
4. Opis chromosomów osobnika (jeśli chce skorzystać z tych operacji: mutowanie i krzyzowanie chromosomów).

Planowane funkcjonalności:

1. Przeprowadzenie pełnego algorytmu genetycznego, na podstawie dostarczonych przez programistę informacji (minimum określone powyżej).
2. Możliwość nadpisania w klasach programisty klas pierwszego i drugiego poziomu abstrakcji w celu zmiany typowych scenariuszy algorytmu.

Problemy:

1. Wydzielenie stałych punktów wszystkich algorytmów genetycznych.
2. Zapewnienie, możliwości zmiany fragmentów kodu bez ingerencji w całość (zapewnienie elastyczności).
3. Wybór odpowiedniego poziomu w strukturze klas dla umieszczenia w niej funkcji celu.

Rozwiązania problemów:

1. Stałe punkty zostałe wyznaczone na podstawie analizy pojęciowej problemu.
2. Podział kodu algorytmu na dwa poziomy abstrakcji o różnym stopniu konieczności wprowadzania zmian. Wyższy z poziomów umożliwa wykonywanie określonych akcji w określonych momentach działania algorytmu. W przypadku głębokich zmian w algorytmie programista będzie zmuszony do nadpisania długich metod.
3. Rozwiąznia