Politechnika Śląska

Wydział Automatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Temat projektu: (6) Darwin

Autor:	Kornel Gut
Prowadzący:	mgr inż. Marek Kokot
Rok akademicki:	2018/2019
Kierunek:	informatyka
Rodzaj studiów:	SSI
Semestr:	1
Termin lab:	poniedziałek, 12:00 – 13:30
Sekcja:	15
Termin oddania:	2019-01-18

Treść zadania

Napisać program symulujący ewolucję populacji osobników. Populacja może liczyć dowolną liczbę osobników. Każdy osobnik zawiera chromosom, który jest ciągiem liczb naturalnych. Chromosomy moga być róznej długości. W każdym pokoleniu wylosowywanych jest *k* par osobników, które się nastepnie krzyżują. Krzyżowanie polega na tym, że u każdego osobnika dochodzi do pękniecia chromosomu w dowolnym miejscu. Czesc początkowa chromosomu jednego osobnika łączy się z częścią końcową drugiego.

Inaczej mówiąc: osobniki wymieniają się fragmentami swoich chromosomów. Jeden osobnik może być wylosowany do kilku krzyżowań. Po dokonaniu wszystkich krzyżowań w pokoleniu sprawdzane jest przystosowanie osobników do warunków środowiska. W tym celu dla każdego osobnika wyznaczana jest wartość funkcji dopasowania. Osobnicy, dla których wartość $f < w$ (gdzie w jest progiem wymierania), są usuwane z populacji. Osobnicy, dla których $f > r$ (gdzie r jest progiem rozmnażania) są klonowane.

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników (kolejność przełączników jest dowolna):

-i plik wejściowy z populacją

-o plik wyjściowy z populacją

-w współczynnik wymierania *w* [0, 1]

-r współczynnik rozmnażania *r* [0, 1]

-p liczba pokoleń *p*

-k liczba *k* par osobników losowanych do krzyżowania

Plik wejściowy ma nastepującą postać: Każda linia zawiera jednego osobnika.

Osobnik charakteryzowany jest chromosomem, który jest przedstawiony jako ciąg liczb naturalnych rozdzielonych białymi znakami.

Przykładowy plik wejściowy zawierający populację złożoną z czterech osobników:

2 9 84 9 5 6 25 12

2 98 56 2 54

5 2

8 5 22 5 48 6 1 9 8 7 554 25 235 32

Plik wyjściowy ma identyczny format.

Analiza zadania

Zagadnienie przedstawia problem znajdowania maksimum funkcji przy użyciu algorytmu genetycznego.

Struktury danych

W programie wykorzystano listę jednokierunkową z możliwością deklarowania typu przechowywanej zmiennej. Lista typu <int> przechowuje kolejne geny, reprezentuje więc kolejnych osobników / chromosomy.
Lista ”population” zawiera zagdnieżdżone listy<int>.
Przechowuje kolejnych osobników stanowiących populację.

Algorytmy

Do selekcji osobników w celu rozmmażania lub wymierania użyto metody wybierania przez odrzucanie. Losowany jest osobnik spośród populacji oraz liczba z zakresu pomiędzy minimalną a maksymalną wartością funkcji dopasowania.
Jeśli wartość funkcji dopasowania osobnika jest większa lub równa tej liczbie, osobnik zostaje wybrany do rozmnażania.

W przypadku selekcji osobników do wymarcia, wartość dopasowania mniejsza od wylosowanej (z tego samego zakresu) kwalifikuje osobnika do usunięcia.

W skrajnych przypadkach, gdy zróżnicowanie osobników jest bardzo duże albo bardzo małe, algorytm ten robi się bardzo kosztowny, wykonując wiele pętli odrzucających. Dla tego też odrzucenia są zliczane, a przekroczenie 5000 dla selektora rozmnażania i 10 000 dla śmierci, powoduje przerwanie losowania i zwrócenie wartości false. Powoduje to w funkcji wywołującej przejście do selektora rankingowego, który jest mniej dokładny, ale bardziej wydajny.
Losuje on osobnika z populacji, następnie, jeśli przystosowanie osobnika
jest powyżej średniej, ma on 80% szans na rozmnażanie,
lub 20% na śmierć w przypadku selektora wymierania.
Dla osobników poniżej średniej prawdopodobieństwa są przeciwne.

Rozmnażanie osobników polega na skopiowaniu ich, a następnie skrzyżowaniu wg następującego schematu:

[A,A,A,B,B,B]

[C,C,C,D,D,D]

[A,A,A,D,D,D]

[B,B,B,C,C,C]

Istnieje także 0,01% szansy na mutacje, która polega na usunięciu losowego genu u obu potomnych i dodaniu po jednym genie na początku, z zakresu minimalnej i maksymalnej wartości genu.

Funkcja oceny (dopasowania / dostosowania) oblicza sumę wartości poszczególnych genów, która jest dzielolna przez 2 * n, gdzie n to liczba genów. Zwracany jest sześcian tego wyniku. (w celu zwiększenia różnicy między podobnymi osobnikami).

$$f = \left(\frac{\sum f'(a_i)}{2 * n} \right)^3$$

$f'(a_i)$ to wartość funkcji dla poszczególnego genu, obliczana z funkcji 1 lub 2

1. $f'(a) = -0.040625 * (a - 10) * (a - 90)$
2. $f'(a) = -0.0006205 * a * (a - 950)$

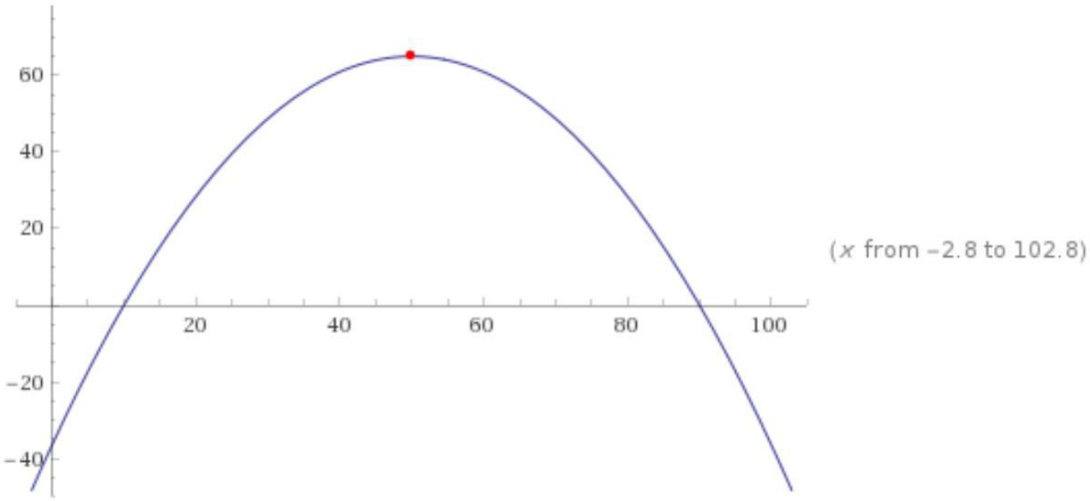
Wykresy

1.

Global maximum:

$\max\{-0.040625\,(a-10)\,(a-90)\} = 65$ at $a = 50$

Plot:

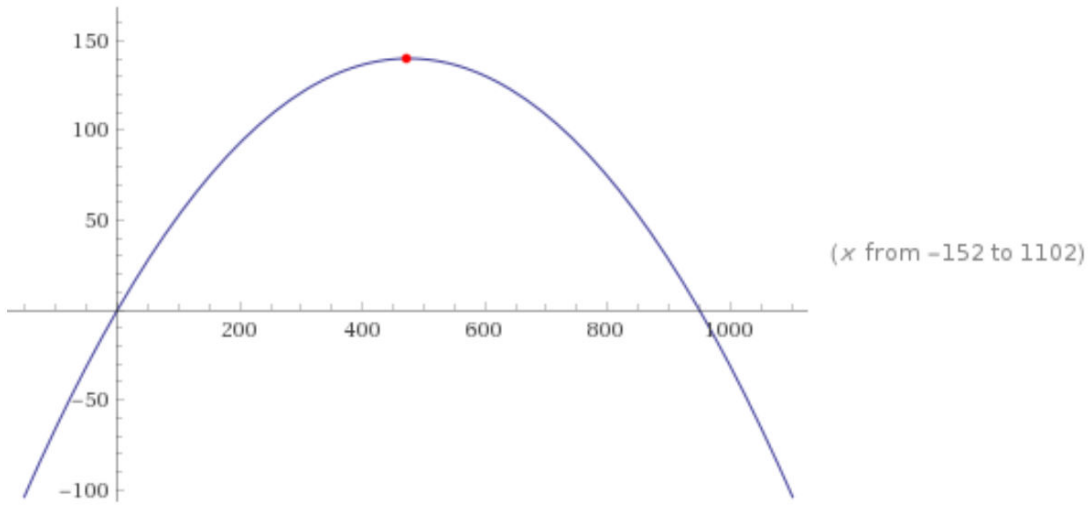


2.

Global maximum:

$$\max\{-0.0006205\, a\, (a - 950)\} = \frac{448\,001}{3200} \text{ at } a = 475$$

Plot:



Specyfikacja zewnetrzna

Program jest uruchamiany z listy poleceń z możliwymi przełącznikami:

-i plik wejściowy z populacją

-o plik wyjściowy z populacją

-w współczynnik wymierania *w* [0, 1]

-r współczynnik rozmnażania *r* [0, 1]

-p liczba pokoleń *p*

-k liczba *k* par osobników losowanych do krzyżowania

Przykład:

Start program.exe -i wejscie.txt i- wyjscie.txt -p 200

Program można uruchomić z dowolnymi parametrami, w dowolnej kolejności, lub bez parametrów. W tym przypadku użyte zostaną parametry domyślne.

Wartości parametrów domyślnych:

-i "First Population.txt"

-o "Final Population.txt"

-w 0.8

-r 0.8

-p 200

-k 0.5

Ponadto w pliku evolution.h dostępny jest przełącznik funkcji oceny

`int func_nr` który dla 1 ustawia funkcję z maksimum = 50,

natomiast dla wartości 2 f(max) = 475.

Dostępne są także parametry generowania pliku z populacją takie jak:

- Liczebność populacji.
- Długość chromosomu - może zostać podana stała, lub wartości z przedziału których długość będzie losowana. W tym przypadku należy stałą ustawić na 0 i nadać pożądane wartości zmiennym chrMin i chrMax tak, aby chrMin < chrMax.
- Minimalna i maksymalna wartość genu.

Specyfikacja wewnetrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym.

W programie rozdzielono interfejs (komunikacje z uzytkownikiem) od logiki aplikacji.

Ogólna struktura programu

Zasadniczą część programu stanowi funkcja void Evolution::Evolve(),

w której w odpowiedniej kolejności wykonywane są poszczególne funkcje składające się na całość działania programu, a są to:

- `generate_random_population_file();`
- `read_population_file();`

Które generują i wczytują plik z populacją.

Następnie w pętli, której ilość iteracji odpowiada parametrowi -p liczby pokoleń, wykonywane są następujące czynności:

- `adjustmentFunction(); // funkcja dopasowania.`
- `RejectionReproductionSelector(); // selekcja do reprodukcji`
- `reproduce(); // reprodukcja`
- `RejectionDeathSelector(); // selektor wymierania`

Po zakończeniu pętli populacja jest zapisywana w pliku wyjściowym przez funkcję `write_population_file();`

Testowanie

- Minimalna długość chromosomu wynośi 2. Jeśli będzie wynosiła 1, gen zostanie skopiowany i dodany do listy.

- W pliku z populacją nieprawidłowe znaki są ignorowane.
- Gdy plik z populacją zawiera mniej osobników niż jest dopuszczalne dla wybranego parametru -k, wyświetlany jest komunikat błędu i minimalna wielkość populacji. Wsp. -k określa procentowy udział populacji w reprodukcji, natomiast niezbędne są do tego przynajmniej 2 chromosomy.
- Zwiększanie długości chromosomów powyżej 20 znacząco wpływa na długość wykonywania programu, zarówno pod względem ilości obliczeń na pokolenie jak i ilości pokoleń.
- Zwiększenie zakresu możliwych wartości genów powyżej ok 10 000 dla funkcji 1 z max = 50 i powyzej 50 000 dla funkcji 2 z max = 475 znacznie zmniejsza dokładność wyniku, należy też zwiększyć populację aby zwiększyć szansę na wystąpienie w zbiorze wartości szukanych.
- Najbardziej optymalna długość chromosomu wynosi 3.
- Optymalny rozmiar populacji to 400 - 2000 w zależności od zróżnicowania genów (potrzebna większa gdy duże) i długości chromosomów (mniejsza gdy długie).
- Program przepisano także na język C# w celu porównania szybkości języka C++ i C#. Porównanie, zgodnie ze spodziewaniami wyszło zdecydowanie na korzyść C++.

Wnioski

Algorytmy genetyczne znajdują maksimum lub minimum spośród wielkiego zbioru losowych danych wejściowych poprzez ich ocenę i odpowiednie probabilistyczne kształtowanie zbioru, tak aby usyskać wyjsciowy zbiór optymalnych wartości dla danej funkcji.

Algorytmy te używane są w problemach np złożonych w których można określić funkcję oceny np. Problem komiwojażera, lub projektowanie układów cyfrowych / elektrycznych.

Znajdują one optymalne rozwiązania, odrzucając te, które są niesprawne i powielając oraz modyfikując lepsze. Dzięki temu odrzucany jest wielki obszar poszukiwań, który nie zawiera interesujących nas rozwiązań, bez potrzeby sprawdzania każdego przypadku z osobna.

Źródła

http://dariusz.banasiak.staff.iiar.pwr.wroc.pl/si/SI_wyklad10.pdf

http://web.mit.edu/deweck/www/PDF_archive/2%20Refereed%20Journal/2_8_SMO_VCLGA.pdf

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.25.5297&rep=rep1&type=pdf

https://www.youtube.com/watch?v=816ayuhDo0E

https://en.wikipedia.org/wiki/Rejection_sampling