Struktura programu GenMachine i schemat blokowy

Program GenMachine składa się z dwóch części. Pierwsza część, umieszczona w pliku main.cpp jest sterująca, rekompilowana dla każdej funkcji celu, druga natomiast jest biblioteką statyczną o nazwie GenMachine.lib, dołączaną za każdym razem do części sterującej i stanowi niezmienny trzon systemu symulacji ewolucji.

Część sterująca ma za zadanie odwoływać się do funkcji biblioteki z pożądanymi w danej chwili argumentami. Wykonuje to w następujących kilkunastu krokach:

- Zainicjowanie zmiennych niezbędnych do pracy programu, takich jak:

logFileName – ścieżka do pliku w którym znajdzie się raport,

iniFileName – ścieżka do pliku tekstowego zawierającego parametry,

population_size – maksymalna liczba osobników w jednym pokoleniu,

vars_number – ilość zmiennych sterujących, czyli długość genotypu.

- Wywołanie konstruktora obiektu klasy GenMachine, wraz z przekazaniem mu nazwy pliku z parametrami, najlepiej ini oraz rozmiaru populacji. W przypadku nie podania drugiego argumentu, konstruktor przyjmuje domyślny rozmiar populacji równy 100 genotypów.
- Ustalenie przedziałów w jakich muszą się mieścić zmienne sterujące funkcji celu i zapisanie ich w zmiennej tablicowej minmax.
- Wywołanie funkcji Init z argumentami określającymi rozmiar genotypu oraz dziedzinę funkcji celu.

- Sprawdzenie czy Etap Inicjalizacji, czyli alokacja pamięci, utworzenie egzemplarza klasy GenMachine oraz utworzenie populacji początkowej przebiegło pomyślnie. Ewentualny numer błędu zostaje zwrócony przez funkcję Init i zachowany w zmiennej
- Uruchomienie iteracyjnego procesu symulacji poprzez wywołanie funkcji Start, zwracającej najlepsze napotkane dostosowanie, czyli notabene poszukiwane optimum funkcji.
- Pobranie wartości zmiennych sterujących z najlepiej dostosowanego genotypu.
- Sporządzenie raportu z pracy programu, zawierającego:

– czas trwania symulacji,

– najlepsze dopasowanie,

– wektor zmiennych sterujących najlepiej dopasowanego genotypu oraz

– następujące dane dotyczące każdego pokolenia: numer pokolenia, najlepsze dostosowanie, średnie dostosowanie, odchylenie standardowe dostosowań, liczba wystąpień krzyżowania każdego typu, liczba wystąpień operatora krzyżowania w ogóle, liczba wystąpień mutowania każdego typu, liczba wystąpień operatora mutowania w ogóle.

- Opcjonalnie, jeśli maksymalna liczba genotypów w pokoleniu nie jest zbyt duża, zachowanie w dodatkowym pliku tekstowym wszystkich zmiennych sterujących, czyli składników każdego osobnika.

Budowę drugiej części programu, czyli biblioteki GenMachine, można przybliżyć przy pomocy podziału ze względu na definicje używanych klas obiektów, lub też ze względu na etapy w jakich jej kod jest wykonywany. W programie używa się trzech klas obiektów:

- Genotype – niewielka klasa służąca do opisu pojedynczego

osobnika. Składa się zaledwie z trzech metod: konstruktora, destruktora, inicjalizatora oraz czterech zmiennych składowych, wystarczających do opisania genotypu:

Gene – jednowymiarowa tablica, czyli wektor wszystkich genów w postaci liczb rzeczywistych;

Fitness – dostosowanie osobnika, czyli wartość jaką zwróciła dla tego osobnika funkcja Formula;

Rfitness – względne dopasowanie osobnika, czyli jego wartość na tle całego aktualnego pokolenia; suma wszystkich względnych dopasowań wynosi jeden, a zatem dopasowanie względne jest prawdopodobieństwem wyboru genotypu do reprodukcji;

Cfitness – dopasowanie łączne, zawiera sumę dopasowania względnego bieżącego osobnika i sumy dopasowań względnych wszystkich osobników o mniejszym numerze porządkowym.

- IniFile – obiekt powiązany z plikiem ini, z zaimplementowanymi algorytmami szybkiego odnajdywania sekwencji znaków w plikach tekstowych, służy do pobrania szesnastu parametrów nie przekazywanych jako argumenty funkcji inicjujących. Oto przykładowa zawartość pliku GenMachine.ini wraz z komentarzem:

[main]

MAXGENS = 40000 – maksymalna liczba pokoleń

[selection]

rozmiar turnieju w turniejowej metodzie selekcji, lub dla wartości 0 i 1 selekcja proporcjonalna

(typu real) prawdopodobieństwo krzyżowania

prawdopodobieństwo wyboru krzyżowania prostego

prawdopodobieństwo wyboru krzyżowania arytmetycznego

prawdopodobieństwo wyboru krzyżowania heurystycznego

cierpliwość krzyżowania heurystycznego, czyli liczba prób spłodzenia potomka dopuszczalnego (wewnątrz dziedziny)

(typu real) prawdopodobieństwo mutacji

prawdopodobieństwo wyboru pierwszej metody mutowania w pierwszym pokoleniu

prawdopodobieństwo wyboru pierwszej metody mutowania w ostatnim pokoleniu

prawdopodobieństwo wyboru drugiej metody mutowania w pierwszym pokoleniu

prawdopodobieństwo wyboru drugiej metody mutowania w ostatnim pokoleniu

prawdopodobieństwo wyboru trzeciej metody mutowania w pierwszym pokoleniu

prawdopodobieństwo wyboru trzeciej metody mutowania w ostatnim pokoleniu

liniowa zmiana prawdopodobieństwa wyboru 1 – tak, 0 – nie

DEGREE = 4 – stopień niejednorodności (3. typ mutacji);

parametr mutacji nierównomiernej (patrz metoda GenMachine::delta())

- GenMachine – główna klasa aplikacji, składająca się z trzydziestu pięciu metod i trzydziestu czterech zmiennych składowych; jeden egzemplarz klasy GenMachine, deklarowany w części sterującej, zawiaduje całym procesem symulacji, deklaruje obiekty pozostałych klas i rezerwuje dla nich zasoby; jej przejrzysty interfejs oraz obsługa błędów umożliwia kalibrację parametrów.

Kod programu GenMachine wykonywany jest w dwóch etapach. W

Etapie Inicjalizacji wywołuje się konstruktora klasy GenMachine, następnie jedną z metod tej klasy o nazwie Init. Etap Symulacji polega na przekazaniu sterowania metodzie Start.

Etap Inicjalizacji

Do czynności jakie wykonywane są po wywołaniu konstruktora klasy GenMachine należą:

- zainicjowanie generatora liczb losowych,
- zapamiętanie przekazanego jako argument wywołania funkcji rozmiaru populacji w zmiennej popsize,
- alokacja pamięci dla zmiennej tablicowej pop, czyli dla podstawowej (aktualnej) populacji obiektów klasy Genotype w liczbie równej popsize,
- alokacja pamięci dla zmiennej tablicowej newpop, czyli dla dodatkowej (nowej) populacji genotypów,
- pobranie parametrów metody z pliku ini, przy pomocy klasy IniFile,
- unormowanie prawdopodobieństw wyboru metod krzyżowania i mutowania,
- rezerwacja pamięci dla tablic raportu oraz pomocniczej tablicy index wspierającej selekcję turniejową.

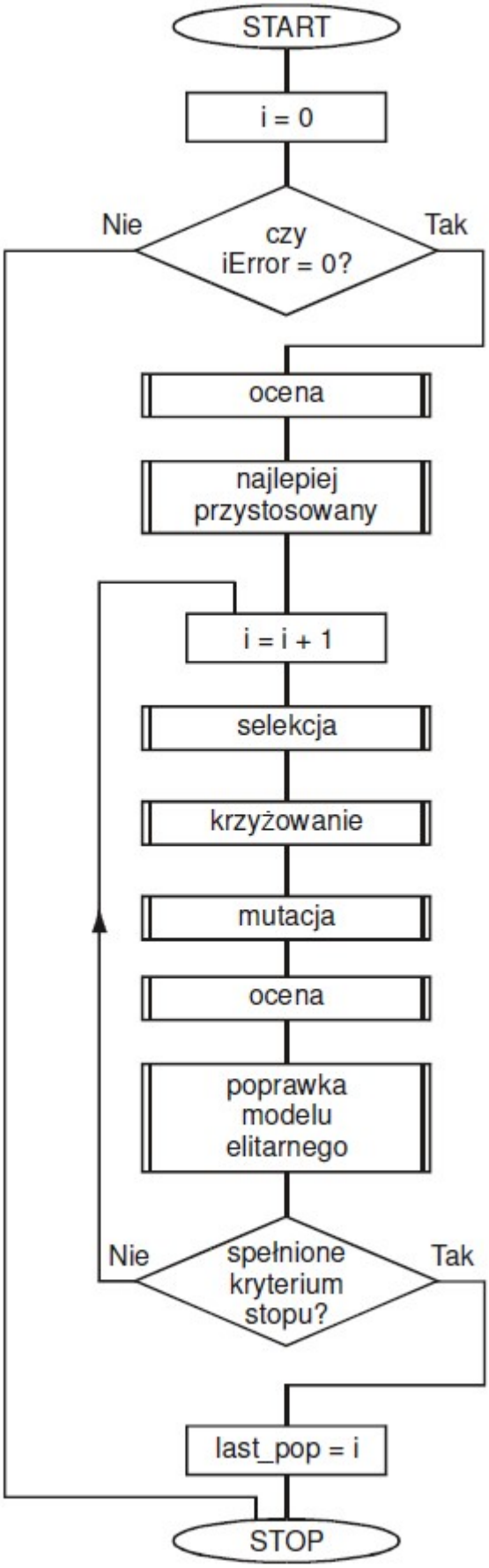
Pozostałe czynności Etapu Inicjalizacji wykonuje metoda Init. Są to:

- sprawdzenie poprawności wykonania konstruktora, czyli kontrola wartości zmiennej iError,
- zapamiętanie podanej jako argument wywołania funkcji Init długości genotypu w zmiennej składowej nvars,
- przepisanie podanych jako argument granic przedziałów do których należeć muszą wartości kolejnych zmiennych decyzyjnych do zmiennej tablicowej minmax,
- utworzenie populacji początkowej przy pomocy inicjalizatora klasy Genotype, który rezerwuje pamięć oraz metody klasy GenMachine o nazwie First – funkcja

First inicjuje wszystkie osobniki wartościami losowymi należącymi do dziedziny funkcji celu,

- wsparcie modelu elitarnego przez rezerwację pamięci dla najlepszego dotychczas osobnika,
- ustawienie flagi iError na zero – **Etap Inicjalizacji** zakończony pomyślnie.

Rys. 4.1. Schemat blokowy Etapu Symulacji


START
i = 0
Nie
czy
iError = 0?
Tak
ocena
najlepiej
przystosowany
i = i + 1
selekcja
krzyżowanie
mutacja
ocena
poprawka
modelu
elitarnego
Nie
spełnione
kryterium
stopu?
Tak
last_pop = i
STOP

Etap Symulacji.

Wszystkie bloki instrukcji wykonywane podczas Etapu Symulacji, czyli po przekazaniu sterowania do funkcji Start oraz kolejność ich wykonywania przedstawione są na Rys. 4.1. Znaczenie poszczególnych klatek schematu blokowego podane jest w poniższym zestawieniu:

- i = 0 – zainicjowanie licznika iteracyjnej pętli głównej zerem. W funkcji Start pełni on zarazem rolę licznika pokoleń.
- Czy iError = 0? – klatka decyzyjna sprawdzająca status błędów aplikacji. Zerowa wartość zmiennej iError oznacza bezbłędne zakończenie Etapu Inicjalizacji. Wartość różna od zera powoduje awaryjne przerwanie działania aplikacji.
- Ocena – wywołanie metody evaluate. Ocena wszystkich osobników aktualnej populacji, poprzez wielokrotne wywołanie funkcji Formula, zawierającej funkcję dostosowania, z wektorem zmiennopozycyjnym jako argumentem wywołania oraz zapisanie wyniku w zmiennej składowej Fitness klasy Genotype.
- Najlepiej przystosowany – wywołanie metody keep_the_best. Implementacja modelu elitarnego selekcji. Sprawdzenie czy po ostatniej ocenie pojawił się osobnik lepiej dostosowany od najlepiej ocenionego dotychczas. Jeśli tak, zapamiętanie najlepszego genotypu.
- i = i + 1 – zwiększenie licznika pokoleń na początku pętli głównej o jeden. Efektem tej zmiany jest zachowanie numerów pokoleń w przedziale [1 , maxgens].
- Selekcja – wywołanie metody select. Wybór genotypów do prokreacji, w zależności od wartości zmiennej składowej tourn, albo przy pomocy selekcji proporcjonalnej po uprzedniej korekcie ujemnego dopasowania, polegającej na przesunięciu wszystkich dopasowań do zbioru liczb nieujemnych, albo przy pomocy selekcji turniejowej, o rozmiarze turnieju ustalonym w ini.

- Krzyżowanie – wywołanie metody crossover(int pop_num). Losowy wybór dwóch osobników zgodnie z podanym przez parametr PXOVER prawdopodobieństwem krzyżowania, następnie utworzenie na ich miejsce dwóch nowych, przy pomocy wybranej losowo metody krzyżowania, zgodnie z przyjętym rozkładem prawdopodobieństwa. Możliwy jest wybór jednej z trzech zaimplementowanych metod:

krzyżowanie proste (jednopunktowe), krzyżowanie arytmetyczne,

krzyżowanie heurystyczne wspierane krzyżowaniem dwupunktowym. Opcjonalny parametr pop_num, dzięki któremu przekazać można metodzie crossover aktualny numer pokolenia, pozwala w razie potrzeby na dynamiczną zmianę częstotliwości krzyżowania (zmienna pxover), lub też zmianę rozkładu prawdopodobieństwa wyboru metody krzyżowania, już podczas trwania symulacji.

- Mutacja – wywołanie metody mut(int pop_num). Sprawdzenie przy pomocy dwóch zagnieżdżonych pętli każdego elementu składowego, wszystkich wektorów zmiennopozycyjnych, będących genotypami aktualnego pokoleniu, czy zgodnie z przyj ętym prawdopodobieństwem wystąpienia mutacji, podanym jako parametr PMUTATION, osobnik zmutował. Jeśli tak, zgodnie z przyjętym statycznym lub dynamicznym rozkładem prawdopodobieństwa, wybór jednego z trzech zaimplementowanych typów mutacji:

mutacji równomiernej, mutacji brzegowej, mutacji nierównomiernej.

Argument wywołania metody mut umożliwia nie tylko zastosowanie dynamicznego, zmieniającego się w czasie rozkładu prawdopodobieństwa wyboru typu mutacji, ale również opcjonalnie zmiany częstotliwości występowania mutacji w zależności od czasu trwania symulacji – czasu rozumianego jako numer populacji.

- Poprawka modelu elitarnego – wywołanie metody elitist. Sprawdzenie całej aktualnej populacji w poszukiwaniu

najlepiej dostosowanego osobnika, porównanie go z najlepszym napotkanym dotychczas genotypem i w zależności od wyniku porównania, albo zapamiętanie go jako najlepszego dotychczas i oznacza to postęp w poszukiwaniu optimum globalnego, albo zastąpienie najgorszego osobnika w bieżącym pokoleniu genotypem jak dotąd najlepszym.

- Spełnione kryterium stopu? – klatka decyzyjna sprawdzająca, czy nie zostało spełnione którekolwiek określonych kryteriów stopu. Niepomijalnym kryterium stopu jest wartość parametru MAXGENS, czyli największego numeru pokolenia. Poza nim na potrzeby eksperymentu, można w metodzie Stop(int pop_num) zdefiniować dodatkowe kryterium zatrzymania symulacji. Może nim być podobnie jak w innych technikach optymalizacji numerycznej różnica między aktualnym wynikiem, a obliczonym inną metodą wynikiem optymalnym, lub prędkość poprawiania się wyniku, lub odchylenie standardowe wartości dostosowań.
- Last_pop = i – zapamiętanie numeru ostatniego pokolenia, na potrzeby części sterującej aplikacji, która generuje raporty. Numer ostatniego pokolenia może zostać pobrany przez część sterującą, przy pomocy metody GetBest.