

UNIwersytet Jagielloński
Wydział Matematyki i Informatyki
Instytut Matematyki

Kamil Hawdziejuk

**Analiza złożoności obliczeniowej
algorytmów ewolucyjnych**

PRACA MAGISTERSKA

NAPISANA POD KIERUNKIEM:

prof. dr hab. Jerzego Ombacha

KRAKÓW, 2008

Spis treści

Wstęp	1
1 Podstawowe definicje	3
1.1 Struktura algorytmu ewolucyjnego	3
1.2 Analiza poślizgu	6
2 Analiza złożoności obliczeniowej	8
2.1 Wprowadzenie i podstawowe oznaczenia	8
2.2 Twierdzenie o wielomianowej złożoności obliczeniowej	9
2.3 Przykład poszukujący optymalnej bryły	11
2.4 Analiza złożoności obliczeniowej przykładu	13
3 Praktyka potwierdzająca teorię	17
3.1 Testy dla przykładu z jednowymiarową bryłą	17
3.2 Testy dla przykładu z wielowymiarową bryłą	25
3.3 Inne zastosowania i wnioski	29
Spis literatury	30

Wstęp

Algorytmy ewolucyjne stanowią efektywną i potwierdzoną w praktyce klasę metod poszukiwania rozwiązań w sposób adaptacyjny. Występująca tu terminologia powstała wskutek inspiracji genetyką i ewolucją. Podejście do problemów zostało bowiem zaczerpnięte z biologii.

Swoją pracę magisterską rozpocznę od wprowadzenia podstawowych pojęć dotyczących problemów optymalizacji i coraz częściej stosowanych do ich rozwiązywania algorytmów ewolucyjnych. W rozdziale 1.2. przedstawię metodę analizy algorytmów ewolucyjnych.

W rozdziale 2. przybliżę pojęcie złożoności obliczeniowej, określając czas działania algorytmów oraz przedstawię również twierdzenia, które mówią o ogólnych wnioskach teoretycznych dotyczących algorytmów ewolucyjnych. Następnie skupię się na konkretnym przykładzie, potwierdzającym w praktyce przedstawioną teorię. Pokażę program stosujący algorytm ewolucyjny i przetestuję jego działanie na tym przykładzie kolejno w wymiarach 1,2,3.

Rozdział 1

Podstawowe definicje

1.1 Struktura algorytmu ewolucyjnego

W technice, ekonomii i innych dziedzinach pojawia się konieczność poszukiwania coraz lepszych rozwiązań różnych problemów — konieczność ich optymalizacji. Optymalizacja, ściśle rozumiana, dotyczy poszukiwania najlepszego rozwiązania. Na ogół chodzi o znalezienie rozwiązania lepszego niż znane dotychczas. Jedną z metod osiągnięcia polepszenia tego rozwiązania dostarcza algorytm ewolucyjny. Przetwarza on populację **osobników**, z których każdy jest propozycją rozwiązania postawionego problemu. Działa on w **środowisku**, które można zdefiniować na podstawie rozwiązywanego **problemu**. W środowisku tym każdemu osobnikowi z populacji jest przyporządkowana wartość liczbowa, określająca jakość reprezentowanego przez niego rozwiązania; wartość ta jest nazywana **przystosowaniem** osobnika.

Wprowadzę następujące oznaczenia:

Niech $V = (D, |\cdot|)$ będzie metryczną przestrzenią poszukiwań, gdzie $D \subset \mathbb{R}^n$ jest zbiorem wartości, a $|\cdot|$ dowolną metryką.

Przez $D_r \subseteq D$ oznaczę r -elementowy zbiór punktów w dziedzinie rozwiązań dopuszczalnych tworzący siatkę, którą będzie przeszukiwać algorytm. Bez

straty ogólności możemy przyjąć, że D_r jest kostką (n -tym iloczynem kartezyjskim zbioru dyskretnego). Elementy tego zbioru to **fenotypy** osobników.

Jedną z podstawowych cech algorytmów ewolucyjnych jest specjalny sposób reprezentowania dopuszczalnych rozwiązań problemu, zwanych tutaj osobnikami. Reprezentację tę uzyskujemy poprzez odpowiednie kodowanie punktów ze zbioru D_r . Zasadniczym celem kodowania jest zmiana zmiennych umożliwiającą przeprowadzenie na kodach punktów specjalnych operacji losowych, zwanych operacjami genetycznymi.

Niech Ω oznacza uniwersum genetyczne stanowiące zbiór kodów, zwanych genotypami osobników próby losowej. Zakładamy zgodność mocy zbiorów Ω i D_r . W rozważaniach moich uniwersum genetyczne stanowiło będzie zbiór wszystkich ciągów binarnych o ustalonej, skończonej długości $n \in \mathbb{N}$.

$$\Omega_n = \{0, 1\}^n.$$

Mamy zatem

$$\#\Omega_n = r = 2^n < \infty.$$

Definicja 1.1 (Kodowanie).

Kodowaniem nazywać będę wzajemnie jednoznaczłą funkcję

$$code : \Omega_n \rightarrow D_r.$$

Ponieważ podstawowe operacje zmierzające do znalezienia przybliżenia rozwiązania wykonywane są na kodach osobników, czyli elementach zbioru Ω_n , konieczne jest zatem przeniesienie funkcji celu do tej dziedziny.

Definicja 1.2 (Funkcja przystosowania).

Określam ograniczoną funkcję przystosowania f_n jako odwzorowanie

$$f_n : \Omega_n \rightarrow [0, M], \quad M < \infty, \quad \forall x \in \Omega_n.$$

oznaczające jakość rozważanego osobnika, zakodowanego jako x .

Problem optymalizacji, który będę rozważać w dalszej części pracy, sformułuję w sposób następujący:

Definicja 1.3 (Problem optymalizacji).

Szukany jest $x^* \in \Omega_n$ taki, że

$$x^* = \arg \max_{x \in \Omega_n} f_n(x).$$

Rozwiązaniem problemu wyjściowego jest oczywiście

$$y^* = \text{code}(x^*).$$

Oznaczę przez $f_{max}^n = f_n(x^*)$.

Definicja 1.4 (Algorytm ewolucyjny).

Algorytm ewolucyjny rozwiązujący problem optymalizacyjny można opisać w następujący sposób:

1. Inicjalizacja:

Niech $k := 0$. Następuje generowanie losowo lub w sposób heurystyczny populacji początkowej N osobników, oznaczanej przez $\xi_0^n = \{x_1, \dots, x_N\}$, gdzie $N > 0$ jest liczbą naturalną. Dla każdej populacji ξ_k^n definiuję

$$f_{\xi_k^n} = \max\{f_n(x_i) : x_i \in \xi_k^n\}.$$

2. Generowanie:

Generowanie nowej pomocniczej populacji poprzez wykorzystanie operatorów krzyżowania (*cross*), mutowania (*mut*), klonowania (czy też każdego innego z operatorów generujących potomka) i oznaczenie jej przez $\xi_{k+1/2}^n$.

3. Selekcja (*sel*):

Wybór i reprodukcja N osobników z sumy populacji $\xi_{k+1/2}^n$ oraz ξ_k^n i uzyskanie nowej pomocniczej populacji ξ_{k+S}^n .

4. Warunek stopu:

Jeśli dla pewnego ustalonego ε zajdzie $|f(\xi_{k+S}^n) - f_{max}| \leq \varepsilon$ lub k osiągnie odpowiednio dużą wartość ustaloną wcześniej K , lub też wartości $f_{\xi_k^n}$ nie

będą się zmieniać w kolejnych iteracjach to zatrzymaj algorytm. W przeciwnym razie przyjmij $\xi_{k+1}^n = \xi_{k+S}^n$ oraz $k := k + 1$ i przejdź do 2.

Algorytmy ewolucyjne stanowią szczególny przypadek stochastycznych algorytmów optymalizacji tworzących i oceniających w każdym kroku zbiór punktów z dziedziny rozwiązań dopuszczalnych, zwany próbą losową lub populacją.

1.2 Analiza poślizgu

Niech $x^* \in \Omega_n$ oznacza optymalne rozwiązanie, czyli osobnika o najwyższej funkcji przystosowania. Wprowadzam funkcję $\bar{d}_n : \Omega_n \times \Omega_n \rightarrow \mathbb{R}^+$ pomiędzy dowolnym osobnikiem x oraz x^* taką, że

$$\bar{d}_n(x, x^*) = 0 \Leftrightarrow x = x^*.$$

Jedynym warunkiem jaki narzucamy na tę funkcję jest jej nieujemność. Ważne jest, aby łączyła się ona z wartością funkcji przystosowania osobnika, określającą jakość rozważanego osobnika, od której zależy również szybkość zbieżności algorytmu do rozwiązania. Przykładowym wyborem może być jej postać

$$\bar{d}_n(x, x^*) = |f_n(x) - f_n(x^*)|.$$

Jeśli istnieje więcej niż jeden punkt optymalny, stosuję oznaczenie Ω_n^* na zbiór takich punktów i przyjmuję:

$$\bar{d}_n(x, \Omega_n^*) = \min\{\bar{d}_n(x, x^*) : x^* \in \Omega_n^*\}.$$

Zakładając, że znane jest optymalne rozwiązanie, używać dalej będę oznaczenia d_n jako funkcji pierwszej zmiennej \bar{d}_n . Jest $d_n(x^*) = 0$ oraz $d_n(x) > 0$ dla $x \notin \Omega_n^*$. Rozważając populację $X = \{x_1, \dots, x_N\} \subset \Omega_n^N$ przyjmujemy

$$d_n(X) = \min\{d_n(x) : x \in X\}.$$

Używam tego oznaczenia do mierzenia odległości danej populacji od rozwiązania optymalnego.

Ciąg $\{d_n(\xi_k^n) : k = 0, 1, 2, \dots\}$ wygenerowany przez algorytm ewolucyjny jest ciągiem zmiennych losowych.

Definicja 1.5 (Poślizg).

Poślizgiem w czasie k nazywam zmienną losową

$$\Delta(d_n(\xi_k^n)) = d_n(\xi_{k+1}^n) - d_n(\xi_k^n).$$

Definicja 1.6 (Moment zatrzymania).

Momentem zatrzymania algorytmu ewolucyjnego nazywam zmienną losową

$$\tau_n = \min\{k : d_n(\xi_k^n) = 0\}.$$

Niech

$$\begin{aligned} \Sigma_k^n &= \{\xi_k^n \in \Omega_n^N : d_n(\xi_k^n) > 0\}. \\ \Sigma_n &= \bigcup_{k=0}^{\infty} \Sigma_k^n. \end{aligned}$$

Definicja 1.7 (Problem rozmiaru n).

Niech będzie dana przestrzeń probabilistyczna $(\Omega_n, \Sigma_n, P_n)$, gdzie $\Omega_n = \{0, 1\}^n$, Σ_n zdefiniowana będzie jak powyżej, a rozkład prawdopodobieństwa P_n określać będą operatory genetyczne zdefiniowane dla konkretnego problemu w dalszej części pracy. Mogę wówczas mówić o problemie rozmiaru n jako zadaniu znalezienia $x^* \in \Omega_n$ metodą algorytmu ewolucyjnego.

W pracy tej przedstawię teorię i warunki narzucone na poślizg, dla których uzyskam wnioski dotyczące czasów działania algorytmów ewolucyjnych. Następnie, używając biblioteki do algorytmów genetycznych zapisanej w języku programowania Java, zweryfikuję ich złożoność na konkretnych przykładach.

Rozdział 2

Analiza złożoności obliczeniowej

2.1 Wprowadzenie i podstawowe oznaczenia

Analiza algorytmu to sposób określenia zasobów, które potrzebne są w celu znalezienia rozwiązania przy pomocy algorytmu. Zasobami są: ilość czasu i miejsca w pamięci, szerokości pasma lub liczby układów logicznych.

W analizie algorytmu czas działania algorytmu spełnia ważną rolę, ponieważ niektóre proste problemy mogą powodować niezwykle długie obliczenia. Miarą złożoności czasowej jest liczba operacji podstawowych w zależności od rozmiaru wejścia. Pomiar rzeczywistego czasu zegarowego jest mało użyteczny ze względu na silną zależność od implementacji algorytmu, użytego kompilatora, maszyny na której algorytm wykonano, a także umiejętności programisty. Dlatego w charakterze czasu wykonania rozpatruje się zwykle liczbę operacji podstawowych (dominujących). Mówi się wówczas o **złożoności obliczeniowej**.

W owej analizie rozważa się przypadek najdłuższego czasu działania algorytmu potrzebnego na znalezienie rozwiązania dla danych wejściowych określonego rozmiaru oraz przypadek średniego czasu oczekiwania na zakończenie

działania danego algorytmu przy założeniu, iż wszystkie dane wejściowe określonego rozmiaru są jednakowo prawdopodobne.

Rozważam problem rozmiaru n . Wprowadzam więc pomocnicze notacje:

Niech

$$g : N \rightarrow [0, \infty].$$

Definicja 2.1 (Notacja O (ograniczenie górne)).

Niech

$$O(g) := \{f : N \rightarrow [0, \infty] : \exists c > 0, \exists n_0 > 0 : 0 \leq f(n) \leq c \cdot g(n), \forall n \geq n_0\}$$

Definicja 2.2 (Notacja Ω (ograniczenie dolne)).

Niech

$$\Omega(g) = \{f : N \rightarrow [0, \infty] : \exists c > 0, \exists n_0 > 0 : 0 \leq c \cdot g(n) \leq f(n), \forall n \geq n_0\}$$

Zwyczajowo przyjęło się pisać, że $f = O(g)$, gdy $f \in O(g)$ oraz $f = \Omega(g)$, gdy $f \in \Omega(g)$.

2.2 Twierdzenie o wielomianowej złożoności obliczeniowej

Twierdzenie 2.3.

Niech dany będzie problem rozmiaru n .

(i) Jeśli istnieje wielomian h_0 taki, że $h_0(n) > 0$ oraz dla każdej populacji, której wartością jest zmienna ξ_k^n zachodzi $d_n(\xi_k^n) \leq h_0(n)$

(ii) Jeśli dla każdego k ($\tau_n \geq k \geq 0$) populacja, której wartość ξ_k^n spełnia $P(d_n(\xi_k^n) > 0) > 0$ oraz istnieje wielomian h_1 ($h_1(n) > 0$) taki, że

$$E[d_n(\xi_k^n) - d_n(\xi_{k+1}^n) | d_n(\xi_k^n)] \geq 1/h_1(n) > 0.$$

Wtedy, startując z dowolnego ξ_0^n , na zbiorze $\{d_n(\xi_0^n) > 0\}$, dostajemy

$$E[\tau_n | d_n(\xi_0^n)] \leq h(n)$$

dla pewnego wielomianu h .

Dowód. Z warunku (ii) otrzymujemy, że

$$E[d_n(\xi_k^n) | d_n(\xi_k^n)] - E[d_n(\xi_{k+1}^n) | d_n(\xi_k^n)] > 0.$$

Ponieważ

$$E[d_n(\xi_k^n) | d_n(\xi_k^n)] = d_n(\xi_k^n),$$

więc

$$d_n(\xi_k^n) \geq E[d_n(\xi_{k+1}^n) | d_n(\xi_k^n)].$$

Dla ustalonego n z warunku (i) dostaję

$$0 \leq d_n(\xi_k^n) \leq h_0(n) \leq \infty,$$

o ile $k < \tau_n$.

Ponieważ $d_n(\xi_\tau^n) = 0$, to

$$E[d_n(\xi_\tau^n) | d_n(\xi_0^n)] = 0.$$

O ile $d_n(\xi_{k-1}^n) > 0$, z warunku (ii) dla $k - 1 < \tau$ mam

$$E[d_n(\xi_{k-1}^n) - d_n(\xi_k^n) | d_n(\xi_{k-1}^n)] \geq 1/h_1(n).$$

$$E[d_n(\xi_{k-1}^n) | d_n(\xi_{k-1}^n)] - E[d_n(\xi_k^n) | d_n(\xi_{k-1}^n)] \geq 1/h_1(n).$$

$$-E[d_n(\xi_k^n) | d_n(\xi_{k-1}^n)] \geq -E[d_n(\xi_{k-1}^n) | d_n(\xi_{k-1}^n)] + 1/h_1(n).$$

$$E[d_n(\xi_k^n) | d_n(\xi_{k-1}^n)] \leq d_n(\xi_{k-1}^n) - 1/h_1(n).$$

$$E[d_n(\xi_{k-1}^n) + \Delta(d_n(\xi_{k-1}^n)) | d_n(\xi_{k-1}^n)] \leq d_n(\xi_{k-1}^n) - 1/h_1(n).$$

Dla każdego $k \in \mathbb{N}$ mam

$$E[d_n(\xi_k^n) | d_n(\xi_0^n)] = E[E[d_n(\xi_{k-1}^n) + \Delta(d_n(\xi_{k-1}^n)) | d_n(\xi_{k-1}^n)] | d_n(\xi_0^n)].$$

Stąd i z powyższego

$$E[d_n(\xi_k^n)|d_n(\xi_0^n)] \leq E[d_n(\xi_{k-1}^n) - 1/h_1(n)|d_n(\xi_0^n)].$$

Przez indukcję

$$E[d_n(\xi_k^n)|d_n(\xi_0^n)] \leq E[d_n(\xi_0^n) - k/h_1(n)|d_n(\xi_0^n)].$$

Ponieważ $d_n(\xi_{\tau_n}^n) = 0$, zatem

$$E[d_n(\xi_{\tau_n}^n)|d_n(\xi_0^n)] = 0.$$

Podstawiając wobec tego $k = \tau_n$ dostaję

$$\begin{aligned} 0 &= E[d_n(\xi_{\tau_n}^n)|d_n(\xi_0^n)] \leq E[d_n(\xi_0^n) - \tau_n/h_1(n)|d_n(\xi_0^n)] \leq \\ &\leq E[d_n(\xi_0^n)] - 1/h_1(n) \cdot E[\tau_n|d_n(\xi_0^n)]. \end{aligned}$$

Stąd

$$E[\tau_n|d_n(\xi_0^n)] \leq h_1(n) \cdot E[d_n(\xi_0^n)] \leq h_0(n) \cdot h_1(n) =: h(n).$$

□

Dygresja:

warunek (i) mówi, że odległość dowolnej populacji od rozwiązania optymalnego jest ograniczona przez funkcję wielomianową z rozmiaru problemu.

warunek (ii) stanowi, że poślizg ciągu zmiennych losowych $\{d_n(\xi_k^n) : k = 0, 1, 2, \dots\}$ w stronę rozwiązania optymalnego jest zawsze dodatni i ograniczony przez odwrotność wielomianu rozmiaru problemu.

2.3 Przykład poszukujący optymalnej bryły

Niech dany będzie wielościan, wyznaczony jednoznacznie przez wektory odległości swoich wierzchołków od środka układu współrzędnych. Rozważam następujący problem. Szukam bryły, w której dla ustalonych kierunków Z wektorów tych odległości bryła ma największą objętość. Ograniczam się do wektorów o długości równej maksymalnie M .

Przyjmuję, że długości wektorów mierzone są liczbami całkowitymi zapisanymi w reprezentacji binarnej przy pomocy n bitów. Niech wobec tego $M = 2^n$.

Przyjmę na początku, że $Z = 1$. W danej postaci problem de facto redukuje się do znalezienia jednego najdłuższego wektora. Oczywiście rozwiązaniem optymalnym jest

$$x^* = (1, \dots, 1) \text{ złożony z } n \text{ jedynek.}$$

Aby rozwiązać ten problem zastosuję algorytm ewolucyjny określony w *Def.1.4*. Krzyżowanie, mutację oraz selekcję zdefiniuję tak jak poniżej.

Stosuję **krzyżowanie jednopunktowe**, tzn. mając dwóch osobników $x = (s_1^{(x)} \dots s_n^{(x)})$ i $y = (s_1^{(y)} \dots s_n^{(y)})$ z populacji ξ_k^n wybieram punkt krzyżowania $m \in \{1, \dots, n-1\}$ w sposób losowy i wymieniam wszystkie bity po m -tym bicie pomiędzy dwoma osobnikami, formułując dwóch nowych osobników x' i y' :

$$x' = (s_1^{(x)} \dots s_{m-1}^{(x)} s_m^{(y)} s_{m+1}^{(x)} \dots s_n^{(x)}),$$

$$y' = (s_1^{(y)} \dots s_{m-1}^{(y)} s_m^{(x)} s_{m+1}^{(y)} \dots s_n^{(y)}).$$

Nowo powstałą populację oznaczam przez ξ_{k+C}^n .

Stosuję operator **mutacji** zmieniający jeden wybrany bit, tzn. mając osobnika $x = (s_1, \dots, s_n)$ z populacji ξ_{k+C}^n wybieram pojedynczy bit s_i w sposób losowy i zamieniam go na przeciwny. Nowo powstałą populację oznaczam przez ξ_{k+M}^n .

Stosuję wreszcie **selekcję elitarną**, tzn. wybieram N osobników z populacji ξ_k^n oraz ξ_{k+M}^n w sposób następujący: najlepszy osobnik z największą funkcją przystosowania kopiowany jest z prawdopodobieństwem przynajmniej $(1 - e^{-n})$ do nowej populacji ξ_{k+S}^n a pozostałe osobniki trafiają do niej zgodnie z rozkładem wartości ich funkcji przystosowania.

2.4 Analiza złożoności obliczeniowej przykładu

Twierdzenie 2.4.

Dla przedstawionego powyżej problemu rozmiaru n i określonego dla niego algorytmu ewolucyjnego zachodzi

$$E[\tau_n | \xi_0^n = X] \leq O(n^2),$$

gdzie X jest dowolną populacją taką, że $d_n(X) > 0$.

Dowód. Definiuję funkcję odległości $d_n(x) = \sum_{i=1}^n |s_i - 1|$. Zgodnie z Tw.2.3 wystarczy sprawdzić, czy ciąg zmiennych losowych $\{d_n(\xi_k^n) : k = 0, 1, 2, \dots\}$ spełnia warunki (i) oraz (ii).

Z powyższej definicji wynika, że dla każdej populacji X

$$d_n(X) \leq n.$$

Wobec tego ciąg zmiennych losowych $\{d_n(\xi_k^n) : k = 0, 1, 2, \dots\}$ spełnia warunek (i).

Niech $I\{A\}$ oznacza zdarzenie, spełniające warunek A. Dla każdego $k \geq 0$ oraz populacji ξ_k^n z $d_n(\xi_k^n) > 0$ badam teraz wpływ krzyżowania na poślizg. Może zajść jeden z trzech przypadków: (1)przypadek $I\{d_n(\xi_{k+C}^n) < d_n(\xi_k^n)\}$, (2)przypadek $I\{d_n(\xi_{k+C}^n) = d_n(\xi_k^n)\}$, (3)przypadek $I\{d_n(\xi_{k+C}^n) > d_n(\xi_k^n)\}$.

Pokażę najpierw, że przypadek $I\{d_n(\xi_{k+C}^n) > d_n(\xi_k^n)\}$ nie może zajść. Innymi słowy, krzyżowanie nie produkuje gorszej populacji. Załóżmy, że x_1 i x_2 są dwoma osobnikami w populacji ξ_k^n oraz y_1 i y_2 są ich potomkami. Ponieważ krzyżowanie nie zmienia ilości bitów równych 1 w x_1 i x_2 , dostaję

$$d_n(y_1) + d_n(y_2) = d_n(x_1) + d_n(x_2),$$

co daje

$$d_n(y_1) - d_n(x_1) = -(d_n(y_2) - d_n(x_2)).$$

Oznacza to, że ze wzrostem poślizgu osobników związany jest spadek poślizgu innych osobników. Dlatego też krzyżowanie nie spowoduje pogorszenia populacji ξ_{k+C}^n . Zdarzenie $I\{d_n(\xi_{k+C}^n) > d_n(\xi_k^n)\}$ nie może więc zajść.

Założmy, że zaszło $I\{d_n(\xi_{k+C}^n) = d_n(\xi_k^n)\}$. Wtedy jeden z następujących przypadków miał miejsce: (a) przypadek $I\{d_n(\xi_{k+M}^n) < d_n(\xi_{k+C}^n)\}$, (b) przypadek $I\{d_n(\xi_{k+M}^n) > d_n(\xi_{k+C}^n)\}$, (c) przypadek $I\{d_n(\xi_{k+M}^n) > d_n(\xi_{k+C}^n)\}$.

Zdarzenie $I\{d(\xi_{k+M}^n) = d(\xi_{k+C}^n)\}$ nie może zajść, gdyż mutacja występuje zawsze zmieniając jeden bit na przeciwny. Prawdopodobieństwo zdarzenia $I\{d_n(\xi_{k+M}^n) < d_n(\xi_{k+C}^n)\}$ nie jest mniejsze od $1/n$ (jeśli $d_n(\xi_{k+C}^n) > 0$), więc prawdopodobieństwo zdarzenia $I\{d_n(\xi_{k+M}^n) > d_n(\xi_{k+C}^n)\}$ nie jest większe od $(n-1)/n$. Jeśli $d_n(\xi_{k+C}^n) = 0$, to populacja ξ_{k+C}^n ma jednego osobnika z najwyższą funkcją przystosowania.

Założmy, że zaszło zdarzenie $I\{d_n(\xi_{k+C}^n) < d_n(\xi_k^n)\}$. Następnie jedno z trzech zdarzeń mogło mieć miejsce: (a') przypadek $I\{d_n(\xi_{k+M}^n) < d_n(\xi_{k+C}^n)\}$, (b') przypadek $I\{d_n(\xi_{k+M}^n) = d_n(\xi_{k+C}^n)\}$, (c') przypadek $I\{d_n(\xi_{k+M}^n) > d_n(\xi_{k+C}^n)\}$. Prawdopodobieństwa tych trzech zdarzeń są podobne do tych analizowanych w przypadku $I\{d_n(\xi_{k+C}^n) = d_n(\xi_k^n)\}$.

Zajmijmy się teraz rolą selekcji. Osobnicy z najlepszą funkcją przystosowania pojawią się w następnej populacji ξ_{k+S} z prawdopodobieństwem $1 - e^{-n}$, więc prawdopodobieństwo $P(d_n(\xi_{k+S}^n) < d_n(\xi_k^n) | d_n(\xi_{k+M}^n) < d_n(\xi_k^n))$ nie jest mniejsze od $1 - e^{-n}$, a prawdopodobieństwo $P(d_n(\xi_{k+S}^n) > d_n(\xi_k^n) | d_n(\xi_{k+M}^n) < d_n(\xi_k^n))$ nie jest większe od e^{-n} . Wtedy prawdopodobieństwo $P(d_n(\xi_{k+S}^n) > d_n(\xi_k^n) | d_n(\xi_{k+M}^n) > d_n(\xi_k^n))$ nie jest większe od e^{-n} , a zdarzenie $I\{d_n(\xi_{k+S}^n) < d_n(\xi_k^n) | d_n(\xi_{k+M}^n) > d_n(\xi_k^n)\}$ nie może zajść.

Dla czytelności zapisu wprowadzę następujące oznaczenia:

$$D_k^n := d_n(\xi_k^n),$$

$$D_M^n := d_n(\xi_{k+M}^n),$$

$$D_S^n := d_n(\xi_{k+S}^n),$$

$$D_C^n := d_n(\xi_{k+C}^n).$$

Mając na uwadze wszystkie powyżej rozważane przypadki, otrzymujemy:

$$\begin{aligned} E[D_{k+1}^n - D_k^n | D_k^n] &= \\ &= E[(D_{k+1}^n - D_k^n)I\{D_C^n < D_k, D_M^n < D_C^n, D_S^n < D_k^n\} | D_k^n] + \\ &+ E[(D_{k+1}^n - D_k^n)I\{D_C^n < D_k, D_M^n < D_C^n, D_S^n > D_k^n\} | D_k^n] + \\ &+ E[(D_{k+1}^n - D_k^n)I\{D_C^n < D_k, D_M^n > D_C^n, D_S^n < D_k^n\} | D_k^n] + \\ &+ E[(D_{k+1}^n - D_k^n)I\{D_C^n < D_k, D_M^n > D_C^n, D_S^n > D_k^n\} | D_k^n] + \\ &+ E[(D_{k+1}^n - D_k^n)I\{D_C^n = D_k, D_M^n < D_C^n, D_S^n < D_k^n\} | D_k^n] + \\ &+ E[(D_{k+1}^n - D_k^n)I\{D_C^n = D_k, D_M^n < D_C^n, D_S^n > D_k^n\} | D_k^n] + \\ &+ E[(D_{k+1}^n - D_k^n)I\{D_C^n = D_k, D_M^n > D_C^n, D_S^n > D_k^n\} | D_k^n]. \end{aligned}$$

Zauważywszy, że $|D_{k+1}^n - D_k^n| \leq n - 1$ otrzymujemy:

$$\begin{aligned} E[D_{k+1}^n - D_k^n | D_k^n] &\leq \\ &\leq (-1)P(D_C^n < D_k^n, D_M^n < D_C^n | D_k^n)(1 - e^{-n}) + \\ &\quad + (n - 1)P(D_C^n < D_k^n, D_M^n < D_C^n | D_k^n)e^{-n} + \\ &+ (-1)P(D_C^n < D_k^n, D_k > D_M^n > D_C^n | D_k^n)(1 - e^{-n}) + \\ &\quad + (n - 1)P(D_C^n < D_k^n, D_M^n > D_C^n | D_k^n)e^{-n} + \\ &+ (-1)P(D_C^n = D_k^n, D_M^n < D_C^n | D_k^n)(1 - e^{-n}) + \\ &\quad + (n - 1)P(D_C^n = D_k^n, D_M^n < D_C^n | D_k^n)e^{-n} + \\ &\quad + (n - 1)P(D_C^n = D_k^n, D_M^n > D_C^n | D_k^n)e^{-n}. \end{aligned}$$

Ponieważ

$$P(D_C^n < D_k^n, D_k > D_M^n > D_C^n | D_k^n)(1 - e^{-n}) < 0$$

oraz

$$P(D_C^n < D_k^n | D_k^n) + P(D_C^n = D_k^n | D_k^n) = 1,$$

to dostaję

$$\begin{aligned}
E[D_{k+1}^n - D_k^n | D_k^n] &\leq (-1)P(D_C^n < D_k^n | D_k^n)1/n(1 - e^{-n}) + \\
&\quad + (n-1)P(D_C^n < D_k^n | D_k^n)(n-1)/ne^{-n} + \\
&\quad + (n-1)P(D_C^n < D_k^n | D_k^n)(n-1)/ne^{-n} + \\
&\quad + (-1)P(D_C^n = D_k^n | D_k^n)1/n(1 - e^{-n}) + \\
&\quad + (n-1)P(D_C^n = D_k^n | D_k^n)(n-1)/ne^{-n} + \\
&\quad + (n-1)P(D_C^n = D_k^n | D_k^n)(n-1)/ne^{-n} \leq \\
&\leq (-1)1/n(1 - e^{-n}) + 2(n-1)(n-1)/ne^{-n} \leq \\
&\leq -(1 - e^{-n} - 2(n-1)^2e^{-n})/n.
\end{aligned}$$

Niech

$$\begin{aligned}
l(n) &:= n/(1 - e^{-n} - 2(n-1)^2e^{-n}), \\
h_1(n) &:= 2n.
\end{aligned}$$

Zatem

$$\begin{aligned}
E[d_n(\xi_{k+1}^n) - d_n(\xi_k^n) | d_n(\xi_k^n)] &\leq -1/l(n), \\
E[d_n(\xi_k^n) - d_n(\xi_{k+1}^n) | d_n(\xi_k^n)] &\geq 1/l(n)
\end{aligned}$$

Wtedy przy $n \rightarrow \infty$, $l(n) \leq h_1(n) = O(n)$.

Otrzymaliśmy więc, że

$$E[d_n(\xi_k^n) - d_n(\xi_{k+1}^n) | d_n(\xi_k^n)] \geq 1/h_1(n).$$

Ponadto

$$\lim_{n \rightarrow \infty} h_1(n) > 0.$$

Udowodniłem więc, że ciąg zmiennych losowych $\{d_n(\xi_k^n) : k = 0, 1, 2, \dots\}$ spełnia założenia (i) oraz (ii) Tw.2.3, zatem

$$E[\tau_n | \xi_0^n = X] \leq h(n),$$

gdzie $h(n) = O(n^2)$. □

Rozdział 3

Praktyka potwierdzająca teorię

Do testowania przykładów skorzystam z powszechnie wykorzystywanej biblioteki do algorytmów ewolucyjnych GALib (Genetic Algorithm Library), dostępnej na stronie <http://sourceforge.net/projects/java-galib/>. Jej autorem jest Jeff Smith.

Pokażę kod, wykorzystujący tę bibliotekę, potwierdzając wyniki teoretyczne dla przedstawionego powyżej przykładu. Na koniec przybliżę jak zastosować algorytmy do szerszej i ciekawszej klasy problemów.

3.1 Testy dla przykładu z jednowymiarową bryłą

Poniżej przedstawiam źródła programu napisanego w Javie, realizującego powyższy przykład. Java jest obecnie najczęściej wykorzystywanym na świecie obiektowym językiem programowania stworzonym przez firmę Sun Microsystems (<http://java.sun.com/>).

W celu uruchomienia programu zainstalować należy środowisko Java, dostępne na stronie <http://java.sun.com/javase/downloads/index.jsp>. Do wygodnego pisania w nim programów warto ściągnąć też jedno ze środowisk programistycznych takich jak Eclipse (<http://www.eclipse.org/>), utworzyć w nim projekt, którego źródła znajdują się poniżej oraz dołączyć do niego

wspomnianą bibliotekę.

Działanie programu (*Rys.3.1*) przedstawia się następująco. Wszystko rozpoczyna się od wykonania funkcji `main` (linia 31). Następnie deklarowany jest obiekt `bryla1D` (linia 32), na którym określone są operatory genetyczne zdefiniowane w bibliotece `GALibrary` (dołączonej w linii 1). Potem ustalana jest wielkość populacji (linia 34), a następnie (linie 35-38) tworzone są obiekty reprezentujące bryłę (tutaj wykonuje się konstruktor zdefiniowany w liniach 15-29). Dla każdego z nich uruchamiany jest wątek przetwarzający populację zgodnie ze zdefiniowaną (linie 5-13) funkcją przystosowania. W tym przypadku funkcja zlicza ilość bitów równych 1 w reprezentacji danego chromosomu. Rozmiarem problemu jest tu $n = sizeOfChrom$, czyli wielkość chromosomu, zmieniającego się w programie od $n = 4$ do $n = 500$.

Zamieszczone poniżej wykresy, skonstruowane przy pomocy narzędzia Matlab, przedstawiają zależność czasu działania (w milisekundach) algorytmu od wielkości chromosomu, przy ustalonej liczbie chromosomów w populacji. Na czerwono dopasowany jest wielomian. Zgodnie z *Tw.2.4* jest on rzędu $O(n^2)$. Jego współczynniki a_2, a_1, a_0 otrzymałem, wykorzystując komendę narzędzia Matlab $a = polyfit(X, Y, 2)$, gdzie X jest wektorem przechowującym rozmiary chromosomów, a Y odpowiednio wektorem czasów działania algorytmu. Dla uzyskanych wyników obliczam średni błąd oraz maksymalne odchylenie od wartości przewidzianego wielomianu. Dla każdego współczynnika obliczam również testy jego istotności.

```

1  import com.softtechdesign.ga.*;
2
3  public class GABryla1D extends GAString {
4      /* dla danego chromosomu oblicz wartość funkcji przystosowania */
5      protected double getFitness(int iChromIndex)
6      {
7          String s = this.getChromosome(iChromIndex).getGenesAsStr();
8          int numberOfOnes = 0;
9          for (int i=0;i<s.length();i++) {
10             if (s.charAt(i)=='1') numberOfOnes++;
11         }
12         return (double)numberOfOnes;
13     }
14
15     public GABryla1D(int sizeOfChromosome, int sizeOfPopulation) throws GAException
16     {
17         super(sizeOfChromosome, //chromosom złożony z tylu znaków
18             sizeOfPopulation, //populacja złożona z tylu chromosomów
19             0.9, //prawdopodobieństwo krzyżowania
20             5, //losowa szansa selekcji (niezależnie od funkcji przystosowania)
21             5000, //liczba generacji po której przerywamy obliczenia
22             0, //maksymalna liczba uruchomień przed rozpoczęciem działania
23             10, //maksymalna liczba generacji przed rozpoczęciem działania
24             0.9, //prawdopodobieństwo mutacji
25             0, //liczba dziesiętnych miejsc w reprezentacji chromosomu
26             "01", //przeszren stanów genu (możliwe wartości '0' lub '1')
27             Crossover.ctOnePoint, //typ krzyżowania
28             true); //czy prowadzić statystyki ?
29     }
30
31     public static void main(String[] args) {
32         GABryla1D bryla1D;
33         try {
34             int sizeOfPop = 50;
35             for (int sizeOfChrom = 4; sizeOfChrom<500; sizeOfChrom++) {
36                 bryla1D = new GABryla1D(sizeOfChrom,sizeOfPop);
37                 bryla1D.run();
38             }
39         }
40         catch (GAException gae) {
41             System.out.println(gae.getMessage());
42         }
43         catch (Exception e) {
44         }
45     }
46 }

```

Rysunek 3.1: Kod źródłowy

Rysunek 3.2: Wielkość chromosomu wynosi 10. Dopasowany wielomian $h(n) = 0.0020 \cdot n^2 - 0.0104 \cdot n + 8.9442$,

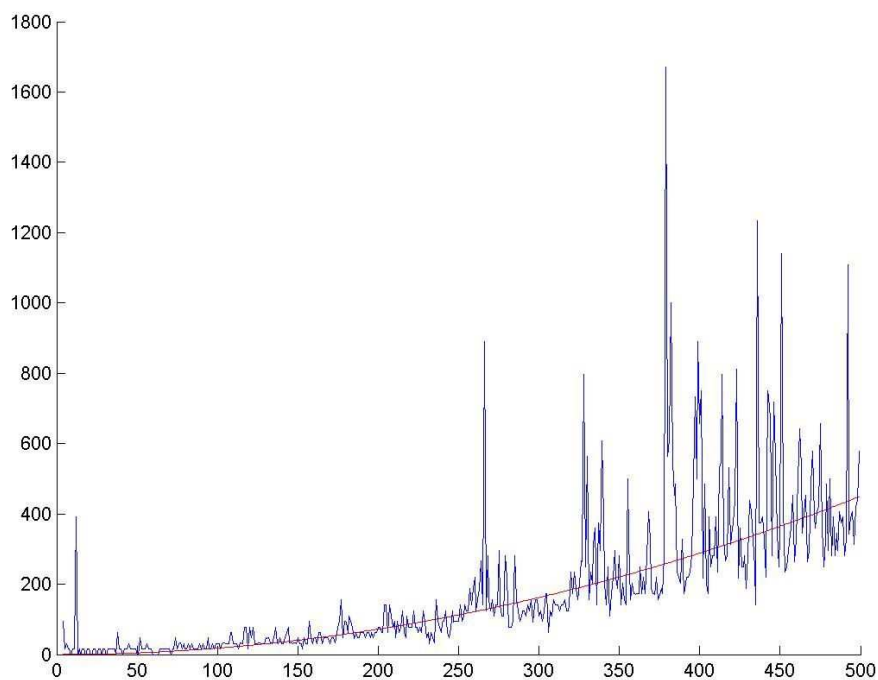
Średni błąd: 0.0701458 sek.

Maksymalny błąd: 1.374 sek.

Testy istotności współczynników:

Współczynnik	$t - Stat$	Wartość- p
a_0	0,466374097	0,641153753
a_1	-0,059153024	0,952854195
a_2	6,017769436	3,45296E-09

Mała wartość- p dla współczynnika a_2 oznacza, że odgrywa on znaczącą rolę.



Rysunek 3.3: Wielkość chromosomu wynosi 20. Dopasowany wielomian $h(n) = 0.0053 \cdot n^2 + 0.0717 \cdot n + 22.3785$,

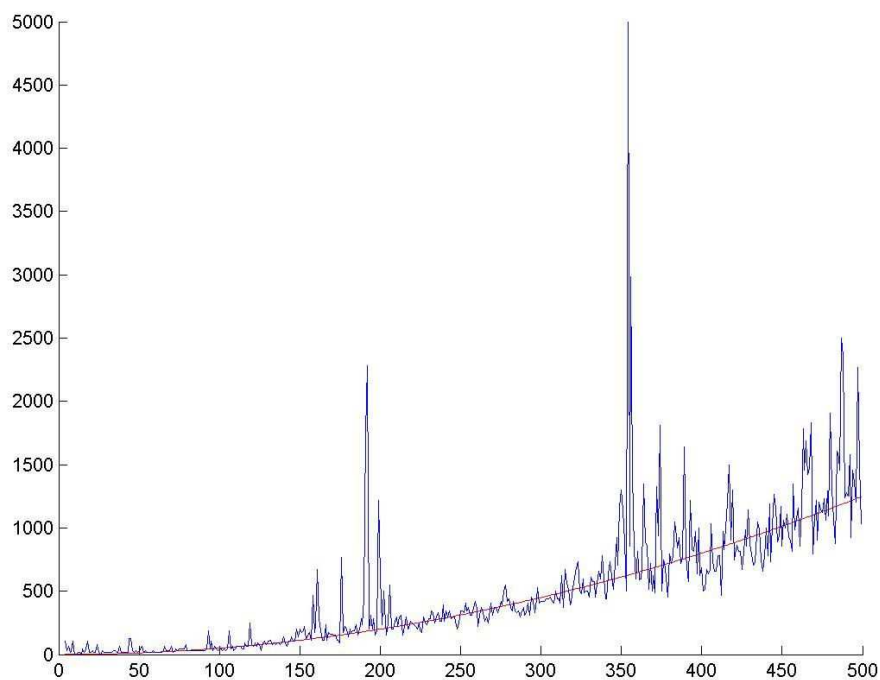
Średni błąd: 0.0701458 sek.

Maksymalny błąd: 1.374 sek.

Testy istotności współczynników:

Współczynnik	$t - Stat$	Wartość- p
a_0	0,51728339	0,60519035
a_1	0,18058007	0,856771407
a_2	6,919111763	1,41821E-11

Mała wartość- p dla współczynnika a_2 oznacza, że odgrywa on znaczącą rolę.



Rysunek 3.4: Wielkość chromosomu wynosi 30. Dopasowany wielomian $h(n) = 0.0141 \cdot n^2 - 1.9596 \cdot n + 174.6478$,

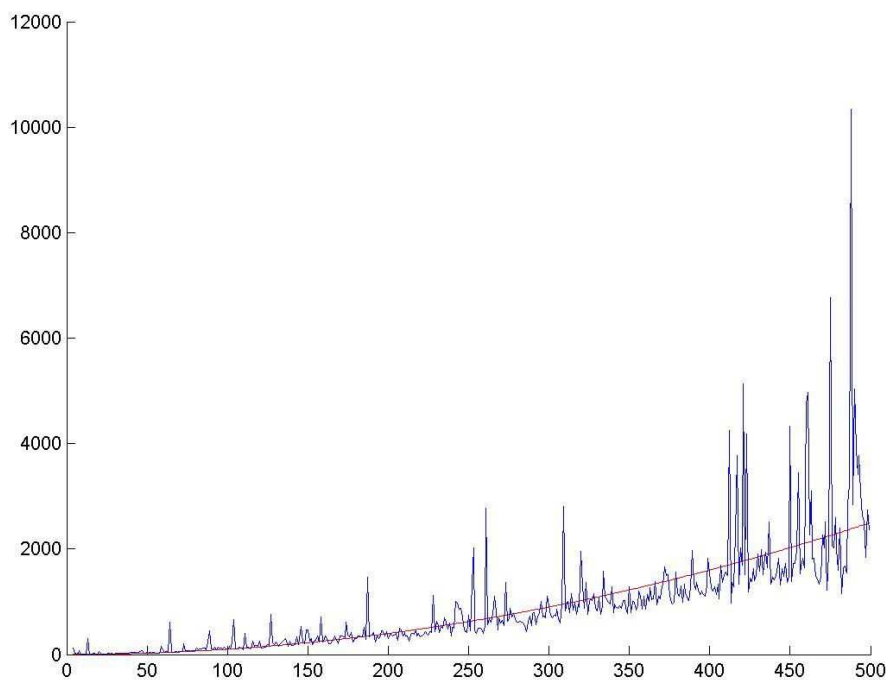
Średni błąd: 0.2701193 sek.

Maksymalny błąd: 7.7737 sek.

Testy istotności współczynników:

Współczynnik	$t - Stat$	Wartość- p
a_0	2,050787639	0,040815708
a_1	-2,508340901	0,012449987
a_2	9,351295947	3,02575E-19

Mała wartość- p dla współczynnika a_2 oznacza, że odgrywa on znaczącą rolę.



Rysunek 3.5: Wielkość chromosomu wynosi 40. Dopasowany wielomian $h(n) = 0.0039 \cdot n^2 + 3.8817 \cdot n - 164.9574$,

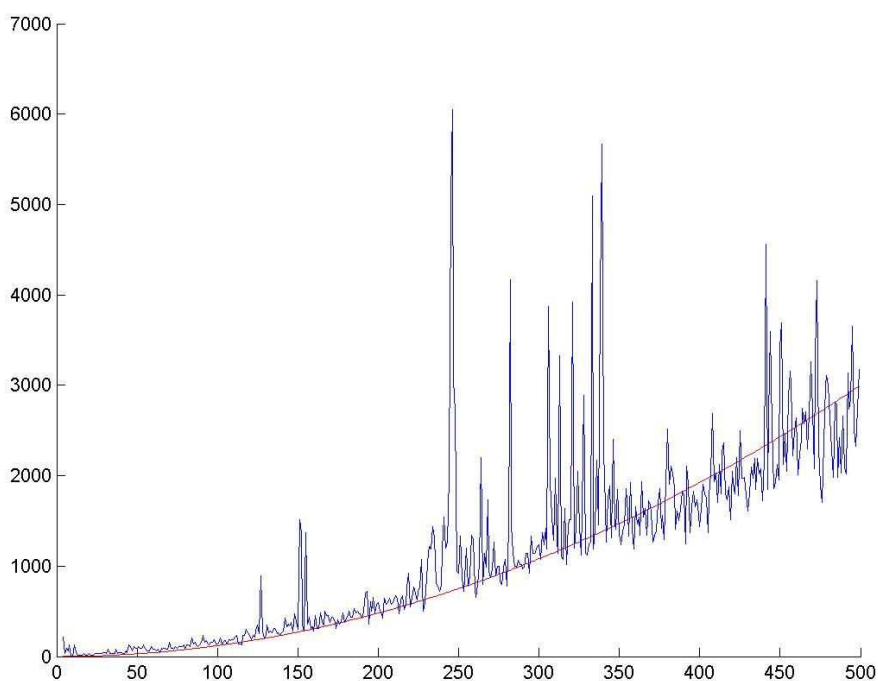
Średni błąd: 0.2996259 sek.

Maksymalny błąd: 5.0218 sek.

Testy istotności współczynników:

Współczynnik	$t - Stat$	Wartość- p
a_0	-2,081933889	0,037863708
a_1	5,340347245	1,41888E-07
a_2	2,775895751	0,005714309

Małe wartości- p dla współczynników a_1 oraz a_2 oznaczają, że odgrywają one znaczącą rolę.



Rysunek 3.6: Wielkość chromosomu wynosi 50. Dopasowany wielomian $h(n) = 0.0175 \cdot n^2 - 0.0222 \cdot n + 77.1842$

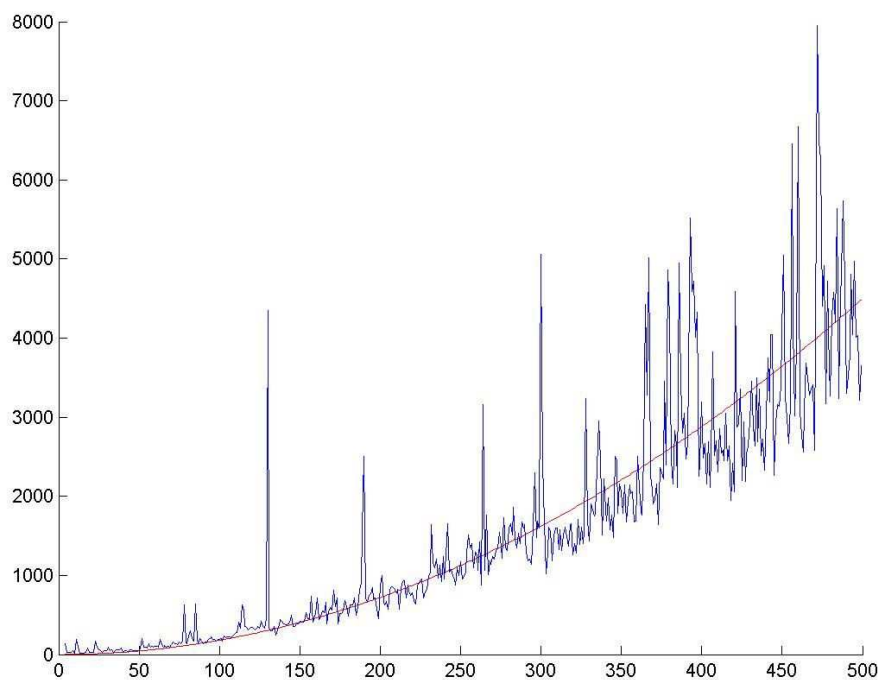
Średni błąd: 0.3318902 sek.

Maksymalny błąd: 3.9963 sek.

Testy istotności współczynników:

Współczynnik	$t - Stat$	Wartość- p
a_0	0,891175386	0,373269804
a_1	-0,02800131	0,977672437
a_2	11,4067119	6,57945E-27

Mała wartość- p dla współczynnika a_2 oznacza, że odgrywa on znaczącą rolę.



3.2 Testy dla przykładu z wielowymiarową bryłą

Przejdę teraz do przykładu z figurą przy większej ilości rozpinających ją wektorów ($Z > 1$), a więc również do możliwości interpretacji jej w przestrzeni dwuwymiarowej czy trójwymiarowej.

Pomysłem na zastosowanie biblioteki GALibrary w tym przypadku jest potraktowanie chromosomu jako zbioru zbudowanego z tych wektorów. Każdy chromosom zbudowany będzie z Z kolejno ustawionych po sobie ciągów zer-jedynkowych o długości $M = 2^n$.

W tym przypadku funkcja przystosowania wyłuskuje z takiej postaci chromosomu wektory, dzieląc chromosom na Z części, po czym przy ustalonych kierunkach oblicza dla nich pole powierzchni (lub objętość w przypadku 3D) rozpinanego przez nie wielokąta (wielościanu). Dla prostoty obliczeń zakładamy, że kąty pomiędzy kolejnymi kierunkami rozpinających wektorów są takie same i wynoszą α .

Wartość przystosowania osobnika dla dwóch wymiarów wyraża się więc wzorem

$$\sum_{i=0}^{Z-1} (1/2) \cdot v(i) \cdot v((i+1) \bmod Z) \sin(\alpha)$$

i jest sumą pól wszystkich trójkątów zbudowanych na sąsiadujących wektorach, z których zbudowany jest wielokąt.

W celu analizy złożoności obliczeniowej tego przykładu metodą poślizgu należy znać rozwiązanie optymalne. Największe pole powierzchni (objętość w 3D) ma oczywiście figura, w której wszystkie wyznaczające ją wektory mają maksymalną długość. Wynika to z prostego spostrzeżenia, że każda inna figura jest w niej zawarta.

Stąd

$$x^* = (\underbrace{1 \dots 1}, \dots, \underbrace{1 \dots 1})$$

złożony jest z Z grup jedynek po n/Z każda.

Rysunek 3.7: Wielkość chromosomu wynosi 20. Dopasowany wielomian $h(n) = 0,0024887 \cdot n^2 + 0,401304838 \cdot n - 16,14821243$.

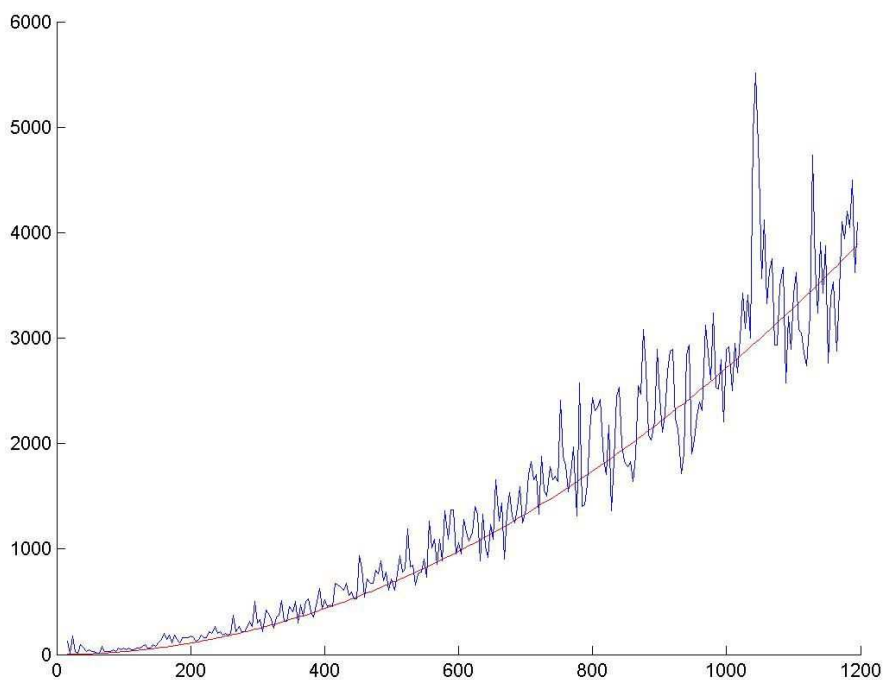
Średni błąd: 0.2701193 sek.

Maksymalny błąd: 7.7737 sek.

Testy istotności współczynników:

Współczynnik	$t - Stat$	Wartość- p
a_0	-0,261275924	0,794063289
a_1	1,706687657	0,088939626
a_2	13,2296369	1,20563E-31

Mała wartość- p dla współczynnika a_2 oznacza, że odgrywa on znaczącą rolę.



Rysunek 3.8: Wielkość chromosomu wynosi 40. Dopasowany wielomian $h(n) = 0,008949 \cdot n^2 - 0,71547 \cdot n - 19,64421$.

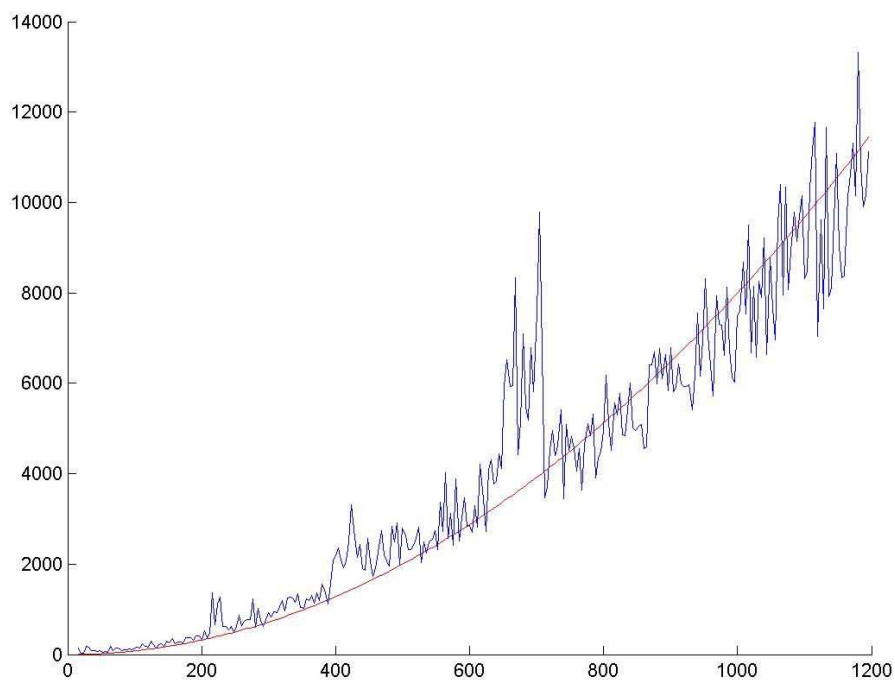
Średni błąd: 1.2188 sek.

Maksymalny błąd: 3.5111 sek.

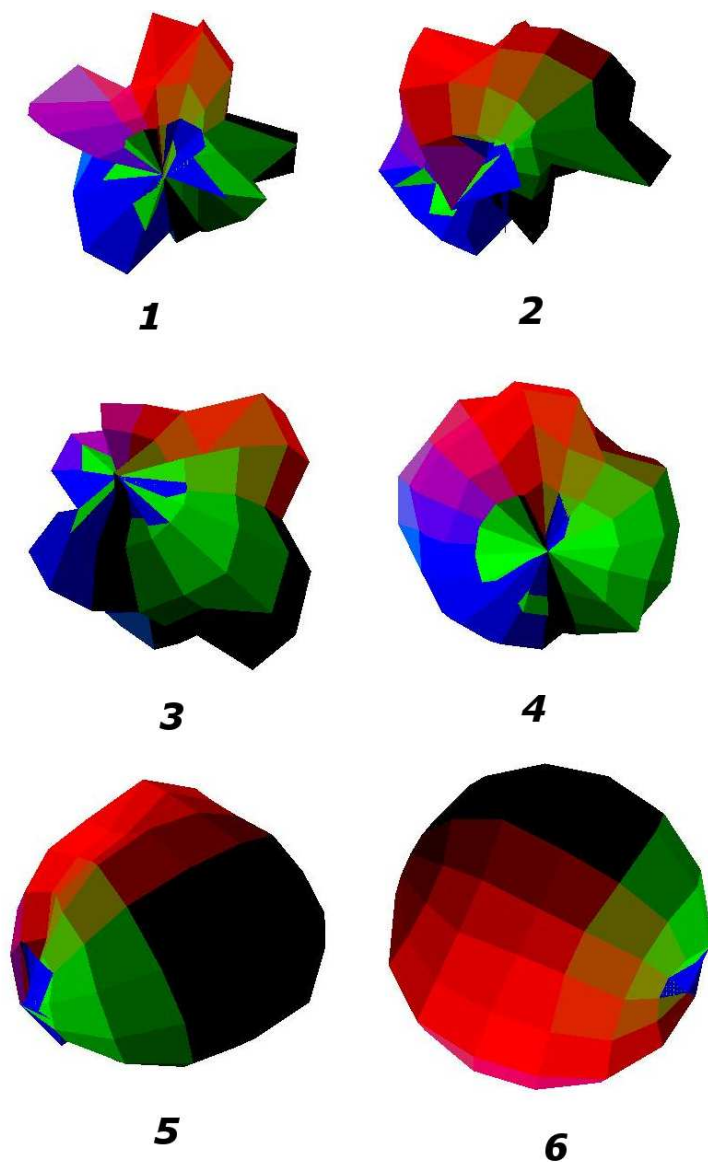
Testy istotności współczynników:

Współczynnik	$t - Stat$	Wartość- p
a_0	-0,044269999	0,964719341
a_1	-0,423809653	0,672015598
a_2	6,626279638	1,64844E-10

Mała wartość- p dla współczynnika a_2 oznacza, że odgrywa on znaczącą rolę.



Rysunek 3.9: W celu zasymulowania powyższego problemu w trzech wymiarach napisałem program (w języku C++). Wizualizacja jego działania, podczas którego wyświetlane są zmiany bryły pod wpływem operatorów genetycznych przy rozmiarze $n = 64$ przedstawiona jest poniżej.



3.3 Inne zastosowania i wnioski

Głównym obszarem zastosowań algorytmów ewolucyjnych jest optymalizacja. Można wyobrazić sobie modyfikację rozważanego przeze mnie problemu poprzez zdefiniowanie funkcji przystosowania jako ilorazu pola powierzchni (objętości) rozważanej figury 2D (3D) do jej obwodu (powierzchni) czy też maksymalizacji jej objętości przy z góry zadanej powierzchni. Ostatecznie definiować można funkcje oceny kształtu figury, dla których analitycznie bardzo trudno znaleźć jest rozwiązanie optymalne. I choć analiza poślizgu nie ma wtedy zastosowania, algorytmy ewolucyjne i tak podsuwają dobrze sprawdzającą się metodę na szukanie rozwiązania z nie do końca wiadomą złożonością obliczeniową. Zupełnie tak jak w przyrodzie...

Spis literatury

- [1] Jun He i Xin Yao, *Drift Analysis and Average Time Complexity of Evolutionary Algorithms*, Artificial Intelligence journal (2001).
- [2] Jarosław Arabas, *Wykłady z algorytmów ewolucyjnych*, wyd.2, Warszawa (2004).
- [3] Robert Schaefer, *Podstawy genetycznej optymalizacji globalnej*, wyd.1, Kraków (2002).
- [4] <http://sourceforge.net/projects/java-galib/> .