WSI

Zadanie 2

Autor:
Michał Paradowski

19.03.2024 (V1.0)

Contents

Zadanie

W tym zadaniu mamy znaleźć minima globalne dwóch podanych funkcji stosując prosty algorytm ewolucyjny z wykorzystaniem mutacji i krzyżowania. Te funkcje to:

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

oraz

$$g(x, y) = -20 \exp\left(-0.2\sqrt{\frac{x^2 + y^2}{2}}\right) - \exp\left(\frac{\cos(2\pi x) + \cos(2\pi y)}{2}\right) + e + 20$$

Są to dobrze znane funkcje Himmelblau i Ackley'a. Wiadomo że minima globalne funkcji Himmelblau osiągane są dla wartości

$$(x, y) \in \{(3, 2), (-2.805118, 3.131312), (-3.779310, -3.283186), (3.584428, -1.848126)\}$$

Zaś minimum globalne funkcji Ackley'a osiągane jest dla

$$(x, y) = (0, 0)$$

W obu przypadkach wartości funkcji w tych minimach to 0.

1 Algorytm

Algorytm Ewolucyjny z którego należy skorzystać można podzielić na następujące kroki:

1 Inicjalizacja początkowej populacji

2 Ocena przystosowania każdego osobnika w populacji

3 Wybór osobników do krzyżowania

4 Krzyżowanie

5 Mutacja

6 Wybór nowej populacji

Przy czym kroki 3-6 powtarzane są przez określoną z góry liczbę iteracji. W opisie poszczególnych kroków zostały przyjęte następujące oznaczenia parametrów wejściowych algorytmu:

- k: Liczba osobników w populacji,
- n: Liczba iteracji,
- L_x, U_x, L_y, U_y: Granice obszaru przeszukiwania,
- p_m: Prawdopodobieństwo muacji,
- p_c: Prawdopodobieństwo krzyżowania,
- σ: współczynnik 'Szybkości' mutacji,
- F: Minimalizowana funkcja (W tym przypadku f lub g)

1.1 Inicjalizacja populacji

Każda populacja składa się z k osobników. Populacja początkowa, składa się z k losowo wybranych elementów zbioru $\{(x, y) : L_x < x < U_x, L_y < y < U_y\}$. Losowy rozkład osobników populacji początkowej zapewnia większą szansę na wylądowanie któregoś z nich blisko szukanego minimum.

1.2 Ocena przystosowania osobników

Najprostszym sposobem na ocenę osobników i tym wykorzystanym w rozwiązaniu tego zadania jest przyporządkowanie każdemu osobnikowi (x_i, y_i) z populacji wartości funkcji F w punkcie (x_i, y_i). Tak więc $O(i) = F(x_i, y_i)$ i osobnik jest tym 'lepszy' im ocena jest niższa.

1.3 Wybór osobników do krzyżowania

W będzie zbiorem osobników wybranych do krzyżowania. Na początku $W = \emptyset$. Zastosowaną w tym rozwiązaniu metodą wyboru osobników do krzyżowania jest metoda turniejowa. Jej zaletą jest prostota implementacji oraz umożliwianie nawet najgorszym osobnikom przejście do następnej populacji (z odpowiednio małą szansą). Metoda ta polega na k-krotnym wyborze dwóch losowych osobników spośród obecnej populacji (!z powtórzeniami!) a następnie dodaniu lepszego z nich do zbioru W. Pojedynczy osobnik może więc trafić do zbioru W więcej niż jednokrotnie.

1.4 Krzyżowanie

K będzie zbiorem osobników po krzyżowaniu. Na początku $K = \emptyset$. By wykonać krzyżowanie należy $\lceil \frac{k}{2} \rceil$ razy wykonać następujące czynności:

1) Należy losowo wybrać dwa elementy zbioru $(x_1, y_2), (x_2, y_2)$
2) Należy wybrać losową liczbę a ze zbioru $(0, 1)$. Jeśli $a > p_c$ należy dodać wylosowane osobniki do zbioru K i wrócić do pkt. 1). W przeciwnym wypadku należy przystąpić do krzyżowania:
3) Należy wylosować wagę krzyżowania β ze zbioru $(0, 1)$ a następnie dodać do zbioru K dwa nowe osobniki: $(\beta x_1 + (1 - \beta)x_2, (1 - \beta)y_1 + \beta y_2)$ i $((1 - \beta)x_1 + \beta x_2, \beta y_1 + (1 - \beta)y_2)$.
4) Powrót do pkt. 1).

Po tych operacjach zbiór K będzie k lub $k + 1$ elementowy. w tym drugim przypadku należy usunąć dowolny element.

Jest to odmiana krzyżowania uśredniającego ze średnią ważoną która umożliwia otrzymanie innych osobników 'spomiędzy' dwóch wylosowanych, niż te będące ich kombinacją liniową.

1.5 Mutacja

Podczas mutacji należy dla każdego osobnika (x_i, y_i) ze zbioru K wykonać następującą sekwencję:

1) Należy wybrać losową liczbę a ze zbioru $(0, 1)$. Jeśli $a > p_m$ to x_i nie zmieni się. W p.p. $x_i \leftarrow x_i + r * \sigma$ gdzie r to liczba ze zbioru $(0, 1)$ wylosowana zgodnie z dystrybucją normalną
2) Należy wybrać losową liczbę a ze zbioru $(0, 1)$. Jeśli $a > p_m$ to y_i nie zmieni się. W p.p. $y_i \leftarrow y_i + r * \sigma$ gdzie r to liczba ze zbioru $(0, 1)$ wylosowana zgodnie z dystrybucją normalną

1.6 Nowa populacja

Zbiór K po zmutowaniu jego elementów staje się nową populacją.

1.7 Podsumowanie działania algorytmu

Analizując przebieg algorytmu można przewidzieć jak poszczególne parametry wpłyną na rezultat. Przykładowo im mniejsze granice obszaru przeszukiwań tym większa szansa że któryś z wylosowanych punktów trafi w okolica minima. Oczywiście wyznaczając obszar przeszukiwań trzeba mieć pewność że szukane minimum się w nim znajduje, Widać też że zwiększenie rozmiaru populacji znacznie zwiększy szansę na szczęśliwy traf ale zwiększy też złożoność pamięciową i obliczeniową. Z tego względu stosuje się raczej małe populacje (rzędu 10) ale za to bardzo wiele iteracji. Zwiększa to precyzję i wpływa pozytywnie na oszczędność zasobów. Parametr σ reguluje rozmiar mutacji. Jest on ważny ponieważ w zależności od właściwości minimalizowanej funkcji mutacja może być niewystarczająca do wyskoczenia z minimum lokalnego lub zbyt duża co skutkuje zmniejszoną precyzją. Dobranie odpowiedniego parametru σ jest kluczowe i często stosuje się algorytmy które ten parametr aktualizują w zależności od wyników dotychczasowych iteracji algorytmu. Wreszcie parametry p_c oraz p_m mogą przyspieszyć działanie algorytmu gdy są większe (losowa mutacja może trafić w szukane minimum), lub wprowadzić zbyt duży chaos gdy są za duże (to może spowodować utratę 'najlepszych' osobników w skutek nadmiernych mutacji).
W trzech eksperymentach poniżej pokazany zostanie wpływ zmiany parametrów p_c, p_m oraz m na błąd wyniku oraz czas pracy algorytmu.

2 Eksperyment 1

W tym eksperymencie przetestowany zostanie wpływ parametru p_c na wyniki.
dla funkcji f parametry wejściowe przyjmą wartości:

- $((L_x, U_x), (L_y, U_y)) = ((-10, 10), (-10, 10))$
- $k = 10$
- $p_m = 0.3$
- $\sigma = 1$
- $n = 100$
- $p_c \in \{\frac{i}{1000} : i \in \{0, 1, 2, ..., 1000\}\}$

Poniżej znajduje się wykres błędu wyniku w zależności od wartości parametru p_c. Błąd wyniku liczony jest wzorem $\epsilon = (f(x_{min}, y_{min}) - min_{R^2}(f))^2 = f(x_{min}, y_{min})^2$ ponieważ minimum globalne obu funkcji z treści zadania to 0.

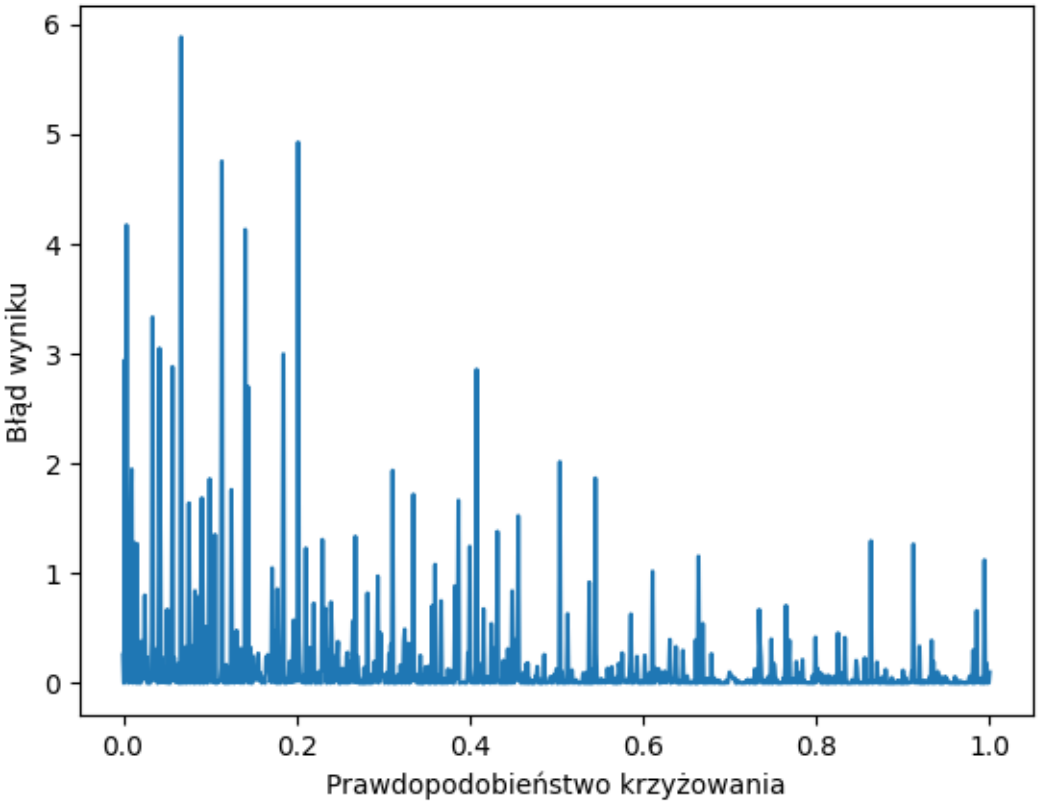

Figure 2.1: Zależność błędu od prawdopodobieństwa krzyżowania (funkcja Himmelblau)

Wszystkie parametry poza p_c zostały eksperymentalnie dobrane tak by wpływ p_c był widoczny. Jak widać na wykresie, wraz ze wzrostem prawdopodobieństwa krzyżowania maleje błąd. Oczywiście nie jest to zmiana monotoniczna ze względu na duży wpływ zdarzeń losowych na ostateczny wynik ale na przestrzeni 1001 eksperymentów wyraźnie zakreśla się przewaga algorytmu z wyższym prawdopodobieństwem mutacji. Wynika to z tego, że podczas krzyżowania występuje duże prawdopodobieństwo poprawy wyniku (na przykład kiedy punkty krzyżowane znajdują się na przeciwnych zboczach otaczających minimum). Oczywiście wynik mutacji może być niekorzystny jednak wtedy w następnej iteracji otrzyma on niską ocenę i z dużym prawdopodobieństwem nie przejdzie fazy selekcji. Tak więc korzyści z krzyżowania znacznie przeważają potencjalne straty. Funkcja Himmelblau ma 4 minima globalne. Jak widać na wykresie poniżej (z tego samego eksperymentu), znajdowane minima rozkładają się mniej więcej po równo na okolice tych 4 prawdziwych.

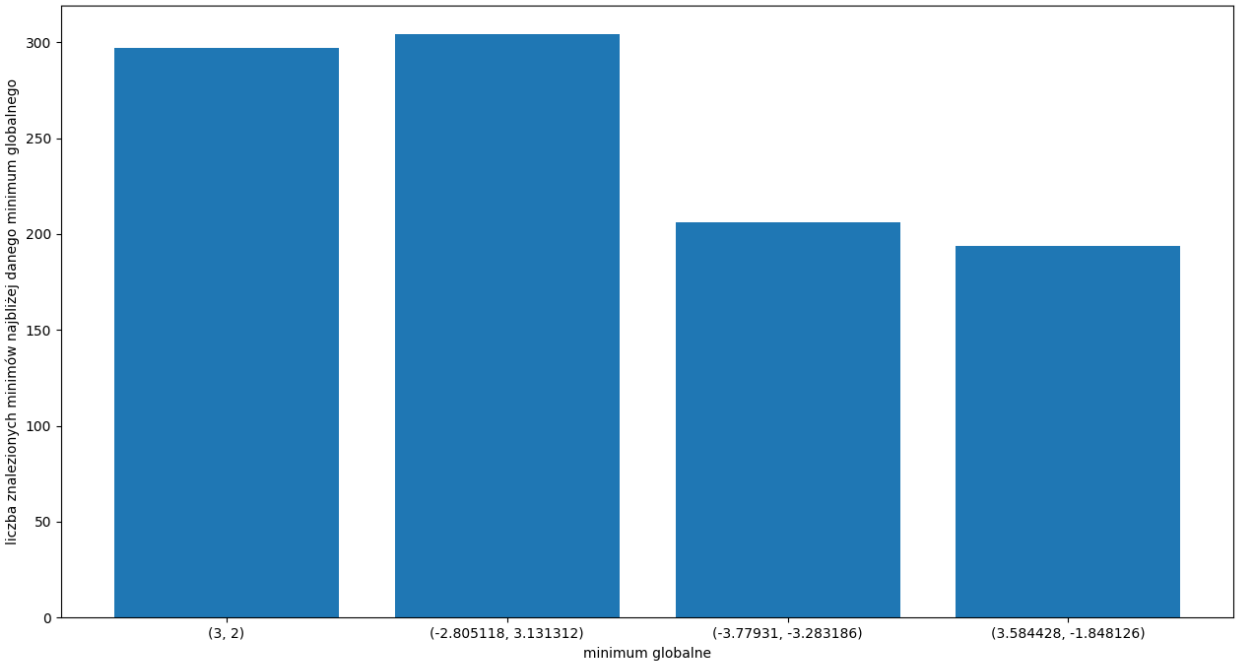

Figure 2.2: Rozkład znalezionych minimów na prawdziwe minima globalne funkcji Hiemmlblau

3 Eksperyment 2

W tym eksperymencie przetestowany zostanie wpływ parametru p_m na wyniki.
Tym razem dla funkcji g parametry wejściowe przyjmą wartości:

- $((L_x, U_x), (L_y, U_y)) = ((-50, 50), (-50, 50))$
- $k = 10$
- $p_c = 0$
- $\sigma = 1$
- $n = 40$
- $p_m \in \{\frac{i}{100} : i \in \{0, 1, 2, ..., 100\}\}$

Na poniższym wykresie widać że prawdopodobieństwo mutacji większe od 40% znacząco poprawia szansę na otrzymanie lepszego wyniku: Wynika to oczywiście z tego, że bez mutacji nie ma

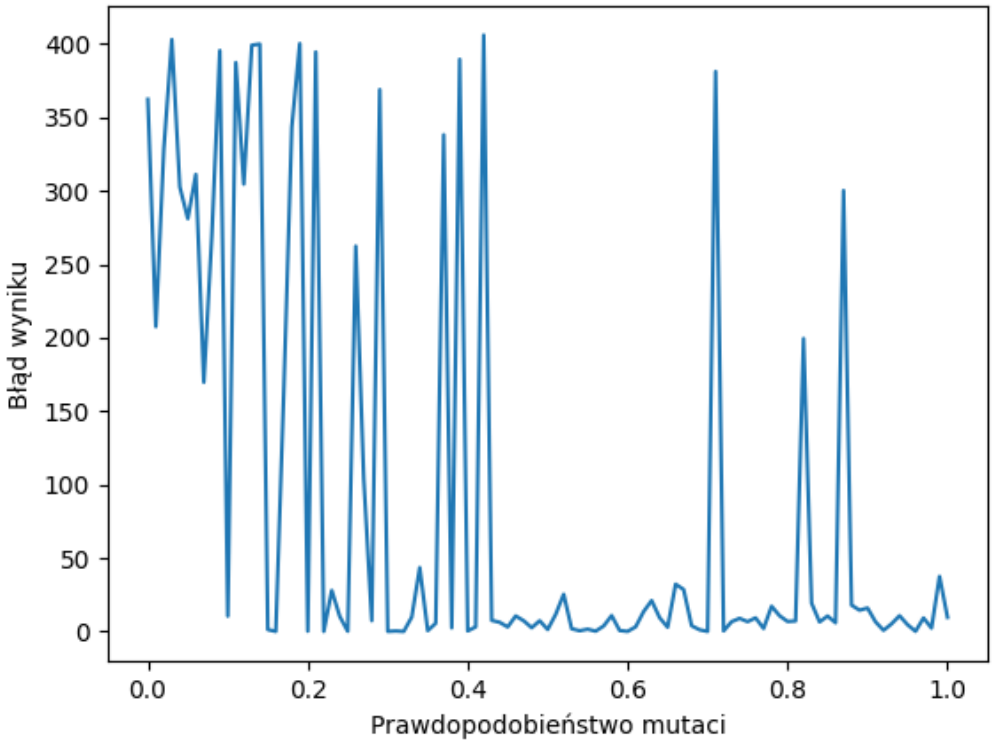

Figure 3.1: Zależność błędu od prawdopodobieństwa mutacji (funkcja Ackley'a)

szans na poprawę wyniku w sposób inny niż krzyżowanie co nie zawsze doprowadza do dobrego

wyniku. Znacznie lepiej jednak widać tą zależność jeśli szukane minimum globalne znajduje się poza początkowymi granicami przeszukiwań:

- $((L_x, U_x), (L_y, U_y)) = ((-10, -5), (-10, -5))$
- $k = 20$
- $p_c = 0.3$
- $\sigma = 1$
- $n = 40$
- $p_m \in \{\frac{i}{1000} : i \in \{0, 1, 2, ..., 1000\}\}$

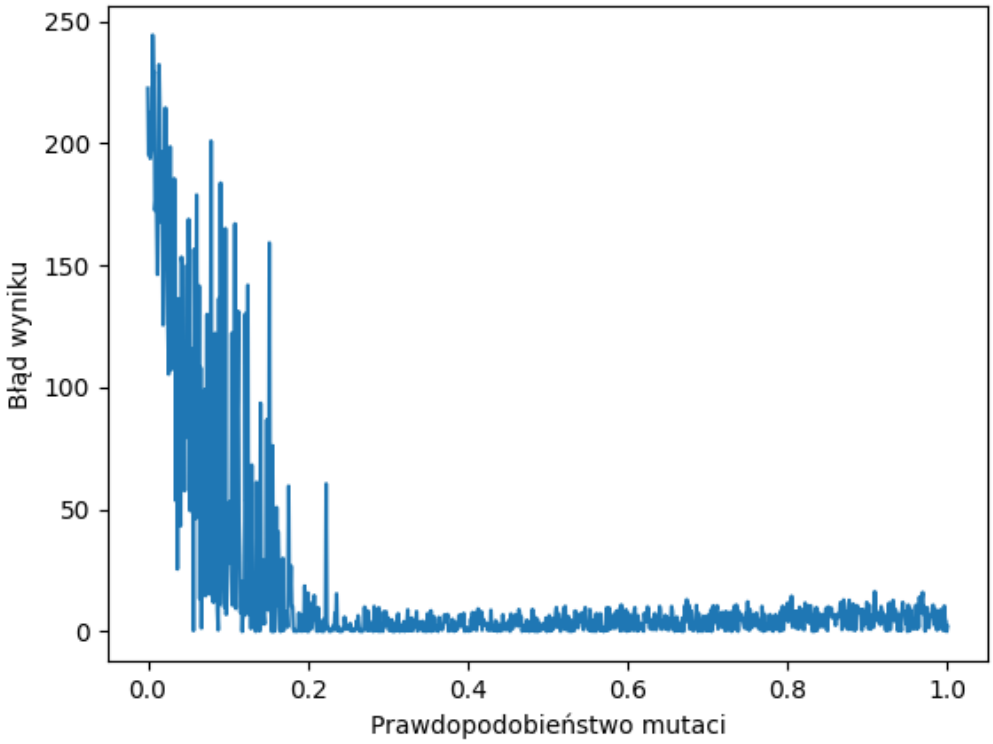

Figure 3.2: Zależność błędu od prawdopodobieństwa mutacji (funkcja Ackley'a)

Tutaj nawet mimo niezerowego prawdopodobieństwa krzyżowania dobre wyniki otrzymane zostały tylko przy odpowiednio dużym prawdopodobieństwie mutacji. Mutacja jest jedynym sposobem na wyjście poza granice startowe gdyż krzyżowanie prowadzi jedynie do stworzenia osobnika którego geny znajdują się pomiędzy genami rodziców co skutkuje skupianiem się populacji w coraz mniejszym obszarze i uniemożliwia tego obszaru opuszczenie.

4 Eksperyment 3

Na podstawie dwóch poprzednich eksperymentów można uznać że $p_c = 0.4$ i $p_m = 0.6$ powinny doprowadzić do uzyskania satysfakcjonujących wyników. W tym eksperymencie zostanie sprawdzone jaki wpływ na wynik ma liczebność populacji. Ten eksperyment przeprowadzony zostanie na obu funkcjach. Na początku przetestowana zostanie funkcja f:

- $((L_x, U_x), (L_y, U_y)) = ((-100, 100), (-100, 100))$
- $p_c = 0.4$
- $\sigma = 1$
- $n = 20$
- $p_m = 0.6$
- $k \in \{1, 2, 3, ..., 100\}$

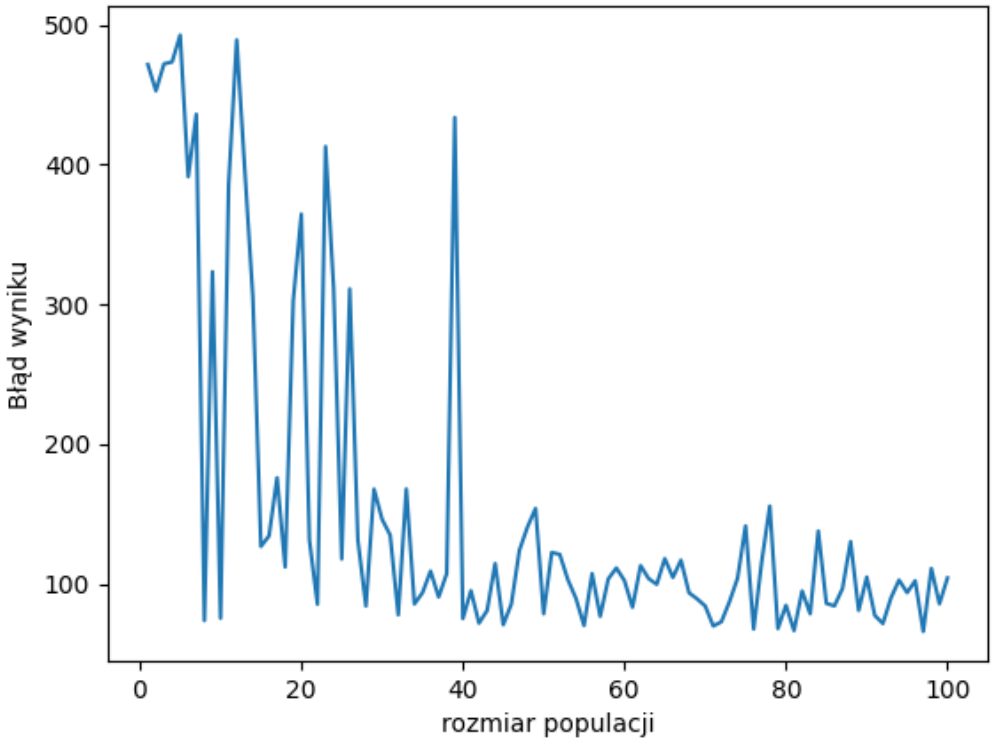

Figure 4.1: Zależność błędu od rozmiaru populacji (funkcja Himmelblau)

Takie wyniki eksperymentu są spodziewane ponieważ przy większej populacji szanse na lepszy wynik wzrastają gdyż obszar przeszukiwań pokrywany jest gęściej. W tym eksperymencie liczba

iteracji została celowo zmniejszona by wpływ wielkości populacji był lepiej widoczny, jednak w poprzednich eksperymentach widać że przy większych wartościach n rozmiar populacji nie musi być aż tak duży. Zawsze warto go zminimalizować by oszczędzić na mocy obliczeniowej. Na następnym wykresie widać jak przy identycznych parametrach zachowa się algorytm na funkcji g: I tutaj widać że większa populacja skutkuje lepszym wynikiem. Obie funkcje testo-

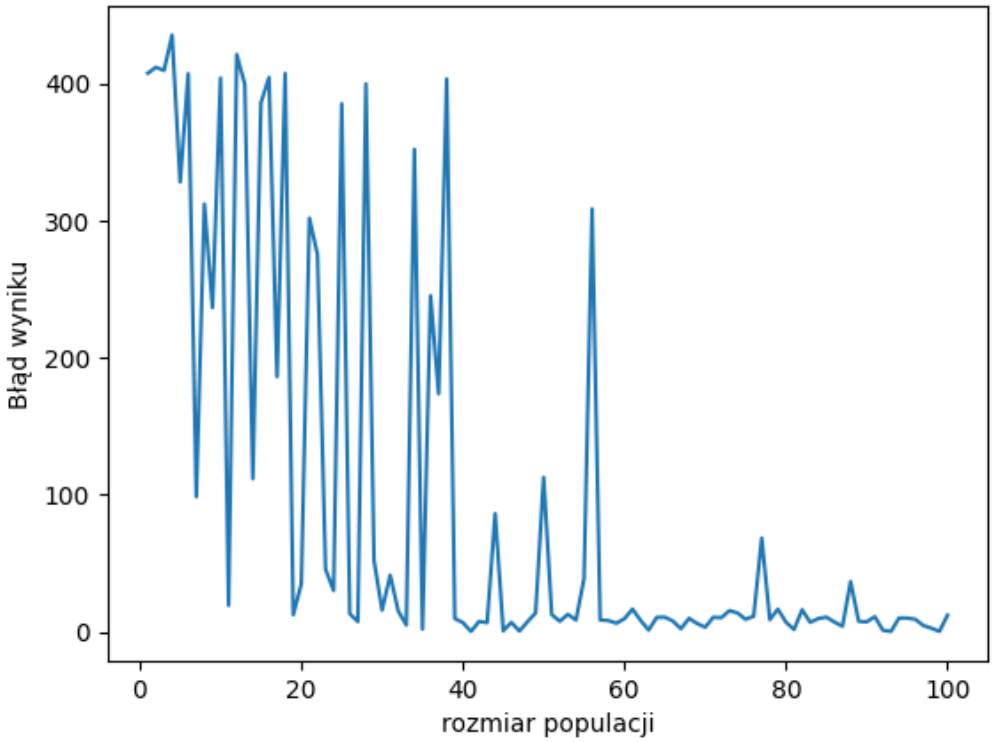

Figure 4.2: Zależność błędu od rozmiaru populacji (funkcja Ackley'a)

we potwierdziły więc logiczne przypuszczenia. W następnej części sprawozdania przedstawione zostaną dwa przykładowe przebiegi algorytmu na funkcjach.

Przykłady

4.0.1 Himmelblau

Przykład algorytmu dla funkcji Himmelblau dla parametrów:

- $((L_x, U_x), (L_y, U_y)) = ((-10, 10), (-10, 10))$
- $p_c = 0.4$
- $\sigma = 1$
- $n = 50$
- $p_m = 0.6$
- $k = 10$

Znalezione minimum: $(x, y, z) = (3.4849, -1.7091, 0.67)$

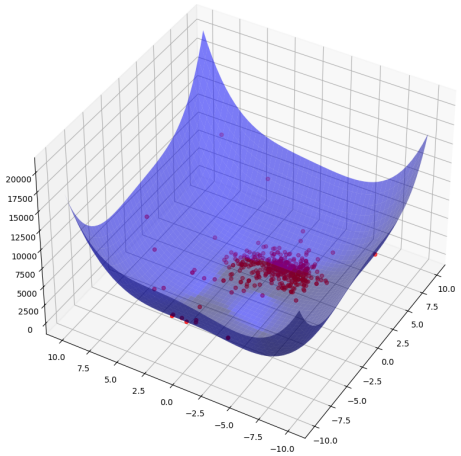

Figure 4.3: Wykres algorytmu dla funkcji Himmelblau

Otrzymany wynik nie jest bardzo precyzyjny ponieważ liczba iteracji została zmniejszona dla jasności wykresu. Na wykresie zaznaczone są wszystkie osobniki wszystkich populacji.

4.0.2 Ackley

Przykład algorytmu dla funkcji Ackley'a dla parametrów:

- $((L_x, U_x), (L_y, U_y)) = ((-10, 10), (-10, 10))$

- $p_c = 0.4$
- $\sigma = 1$
- $n = 50$
- $p_m = 0.6$
- $k = 10$

Znalezione minimum: $(x, y, z) = (0.00538 - 0.07309, 0.34416)$

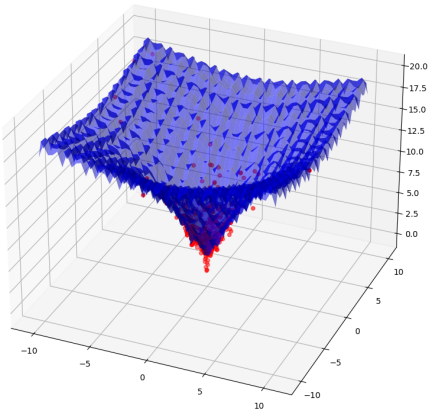

Figure 4.4: Wykres algorytmu dla funkcji Ackley'a

Widać wyraźnie stopniowe zejście kolejnych populacji w stronę minimum.

List of Figures