Metody inteligencji obliczeniowej - algorytmy genetyczne

Wiktor Wierzchowski

June 10, 2025

1 Wstęp

Poniższy raport jest podsumowaniem części tematycznej przedmiotu Metody Inteligencji Obliczeniowej poświęconej algorytmom genetycznym. Zawiera on ogólny przegląd tego czym są te algorytmy wraz z opisem 3 zadań jakie przerabialiśmy na laboratorium.

2 Zasada działania i implementacja

Algorytmy ewolucyjne są jednym z wielu rodzajów algorymtów heruerstycznych. Podejmują one próbe optymalizacji danego problemu nie przez podążanie za gradientem, a z wykorzystaniem zależności ewolucyjnych zaobserowanych w przyrodzie. Zjawisko ewolucji organizmów biologicznych i rządzące nim mechanizmy są bardzo skomplikowane ale przez zaadaptowanie najważniejszych ich elementów, kluczowych dla zachodzenia ewolucji, algorytmy genetyczne są w stanie w relatywnie prosty sposób zaimplementować próbę rozwiązania problemów optymalizacyjnych.

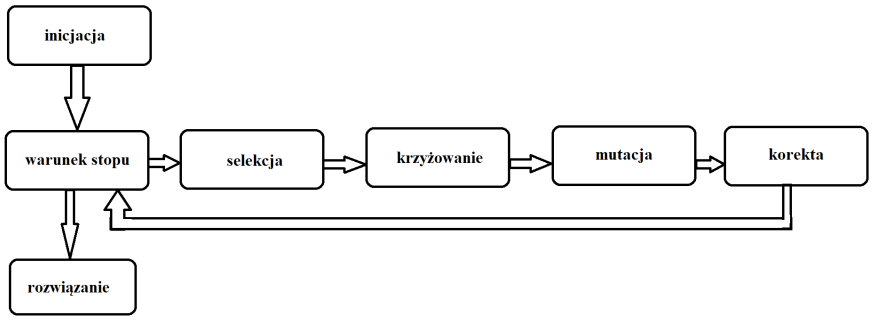

Figure 1: Schemat działania algorytmów genetycznych.

Zasada działania oparta jest na symulowaniu uproszczonego procesu ewolucji pewnej populacji osobników, gdzie każdy osobnik jest propozycją rozwiązania dla zadanego problemu. Genotyp każdego z osobników jest oparty o schemat reprezentujący takie rozwiązanie i pozwala na wykonywanie następujących kluczowych operacji.

2.1 Selekcja

Selekcja to proces, w którym naszym celem jest uszeregowanie osobników z populacji względem tego na ile trafne rozwiązanie problemu prezentują. Następnie zgodnie z tą kolejnością chcemy wybrać osobniki do krzyżowania w sposób, który gwarantuje, że lepsze osobniki będą wybierane z większym prawdopodobieństwem. Najpopularniejsze metody to selekcje ruletkowe, rankingowe oraz turniejowe. Dodatkowo istnieje możliwość zawarcia na tym etapie wstępnej selekcji elitarnej gwarantującej pewne wybranie niewielkiej ilości najlepszych osobników.

2.2 Krzyżowanie

Celem krzyżowania jest wygenerowanie nowego zestawu populacji w oparciu o rodziców wybranych z poprzedniej generacji. Konkretny sposób implementacji jest badzo zależny od postaci problemu, z którym się mierzymy. Przykładowe krzyżowania mogą łączyć dostępne elementy rozwiązania losowo lub próbować zachować fragment potencjalnej poprawności od rodziców przez punktowe przecięcie genotypów. Proporcje tego który z rodziców w większym stopniu wpływa na wygląd nowego osobnika lub ma pierwszeństow w przypadku konfliktów mogą być uzależnione od poziomu przystosowania.

2.3 Mutacja

Etap mutacji pozwala na wprowadzenie losowego szumu do populacji. Każdy algorym rozpoczyna się do losowej inicjacji populacji, jednak gdyby dalsze jej działanie opierało się tylko na krzyżowaniu różnorodność osobników była by ograniczona do tej danej przez inicjacje co mocno ograniczało by zdolność szukania dobrych rozwiązań. W związku z tym dla pewnej frakcji osobników w generacji przeprowadzany jest proces mutacji, również w pełni zależny od konkretnej specyfikacji rozwiązywanego problemu. Ważne żeby treść mutacji była losowa.

Każdy krok algorytmu, w którym wykonana zostanie ta sekwencja operacji określamy jako pojedynczą generację organizmów. Jako że w procesie selekcji premiujemy organizmy lepiej przystosowane powinniśmy się spodziewać, że na przełomie wielu generacji rozważane osobniki będą coraz lepiej odpowiadać na postawiony problem. Warto pamiętać, że optymalizacja najczęściej zawierać się musi w ramach narzuconych ograniczeń i oprócz powyższych etapów zawrzeć należy weryfikacje technicznej poprawności generowanych osobników. Taka weryfikacja może następnie odrzucać niepoprawne rozwiązania i wymuszać ponowne wykonanie pojedyńczego kroku lub próbować skorygować rozwiązania w najprostrzy sposób.

3 Laboratorium 1 - prosta implementacja

Napisanie podstawowego algorytmu genetycznego z mutacją gaussowską i krzyżowaniem jednopunktowym. Funkcje do rozwiązania: prosta $x^2 + y^2 + 2z^2$ oraz pięciowymiarowa funkcja Rastrigina.

3.1 Prezentacja selekcji

W ramach pierwszego zadania przygotowana została selekcja oparta o metodę ruletkową oraz rankingową z prawdopodobieństwem opisanym przez funkcję wykładniczą $e^{\lambda \cdot rank}$. Gdzie lambda dobrana została tak aby prawdopodobieństwo w $cutoff$ stawki wynosiło $grace$ ($\lambda = -\frac{\ln(cutoff)}{grace}$). Do testów przyjęto cuttoff= 0.5 oraz grace= 0.2.

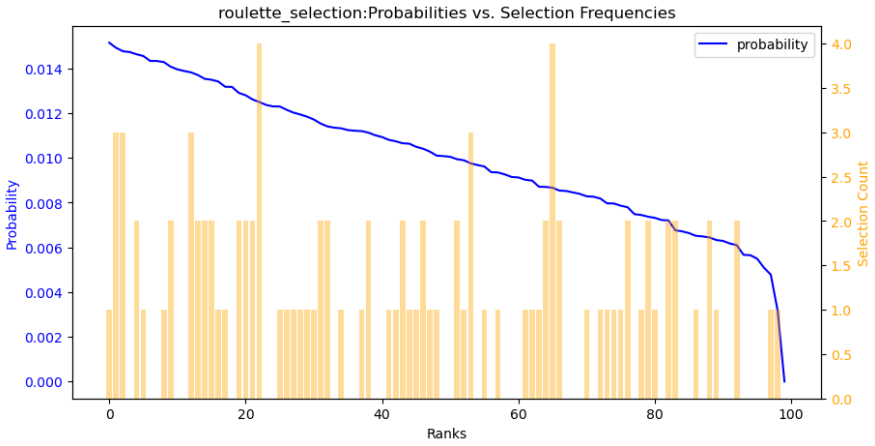

Figure 2: Prezentacja prawdopodobieństw selekcji w selekcji ruletkowej.

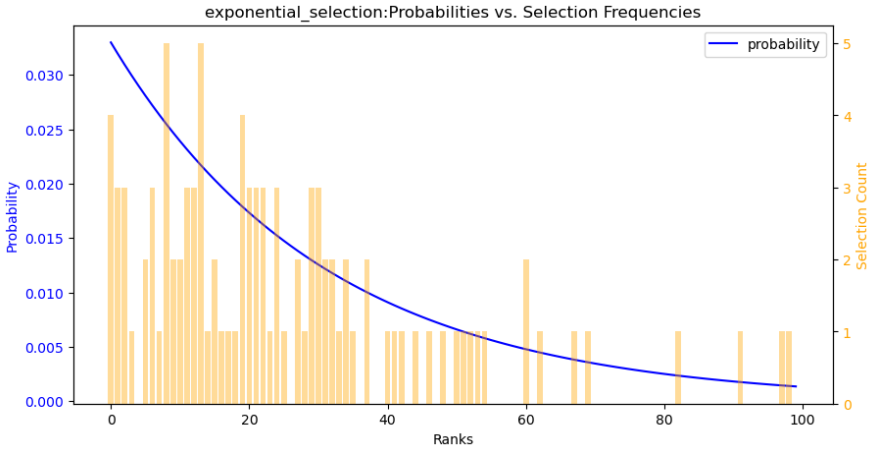

Figure 3: Prezentacja prawdopodobieństw selekcji w selekcji rankingowej wykładniczej.

3.2 Prosta funkcja

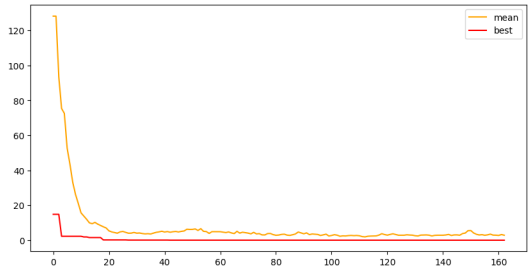

Figure 4: Wyniki uczenia dla prostej funkcji selekcją ruletkową.

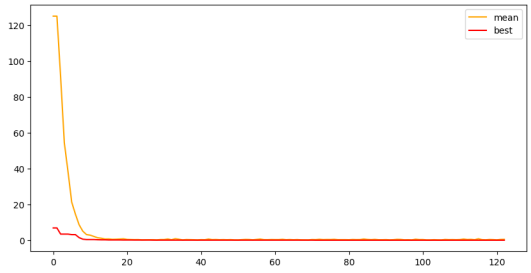

Figure 5: Wyniki uczenia dla prostej funkcji selekcją rankingową wykładniczą.

3.3 Funkcja rastring

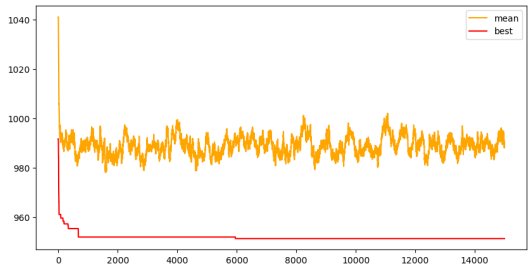

Figure 6: Wyniki uczenia dla funkcji rastring selekcją rankingową wykładniczą z dużą populacją.

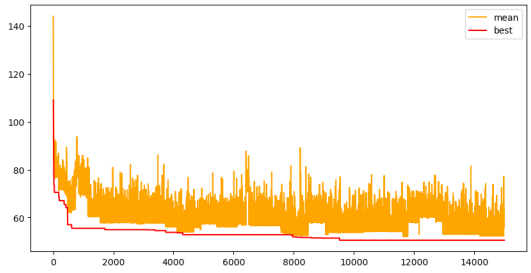

Figure 7: Wyniki uczenia dla funkcji rastring selekcją rankingową wykładniczą z małą populacją.

3.4 Laboratorium 2 - cutting problem

Rozwiązanie problemu cutting stock. Dane jest koło o promieniu r wraz ze zbiorem dostępnych prostokątów zadanych przez trzy liczby: wysokość, szerokość i wartość. Celem jest ułożenie prostokątów w kole tak, aby zmaksymalizować sumę ich wartości, spełniając następujące warunki:

- boki wszystkich prostokątów były równoległe do osi układu,
- wnętrza prostokątów nie miały części wspólnej (intuicyjnie: prostokąty nie nachodzą na siebie, ale mogą się stykać bokami),
- każdy prostokąt można wstawić dowolnie wiele razy.

Wybrane kodowanie opisuje osobniki przez listę pozycji (x, y) oraz wymiarów (a, b). Pozycje wskazują na prawy dolny róg prostokątów. Przetestowane zostały selekcje ruletkowe, rankingowe oraz turniejowe. Krzyżowania polegały na dodaniu do dziecka wszystkich prostokątów od rodziców, a następnie korekcie kolidujących prostokątów przez usunięcie jednego z nich. Mutacje pozwalały na losowe przesunięcie prostokątów, obrócenie, usunięcie lub ich dodanie.

3.4.1 r800

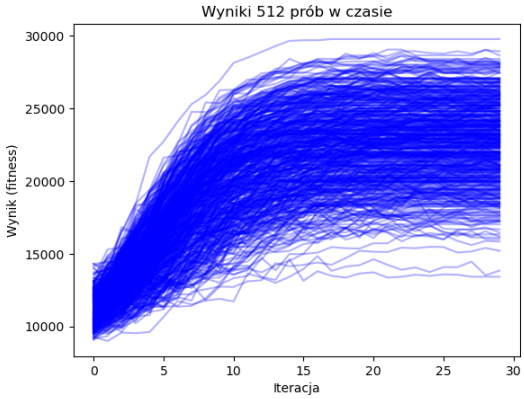

Figure 8: Przegląd grid search dla zadania r800.

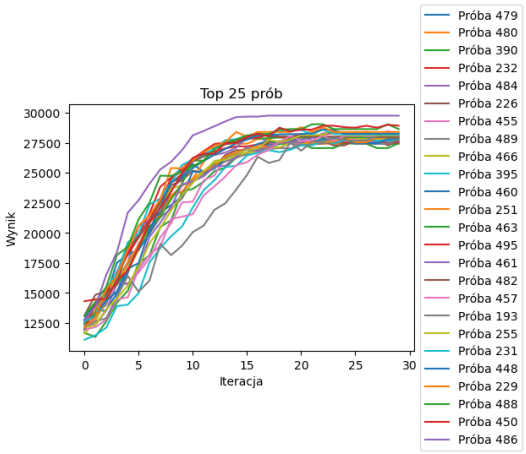

Figure 9: Przegląd najlepszych w grid search dla zadania r800.

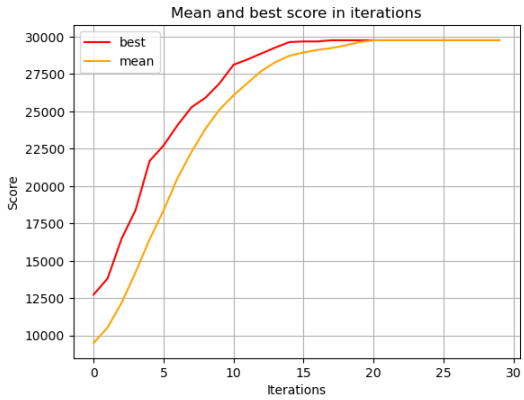

Figure 10: Przegląd najlepszego uczenia dla zadania r800.

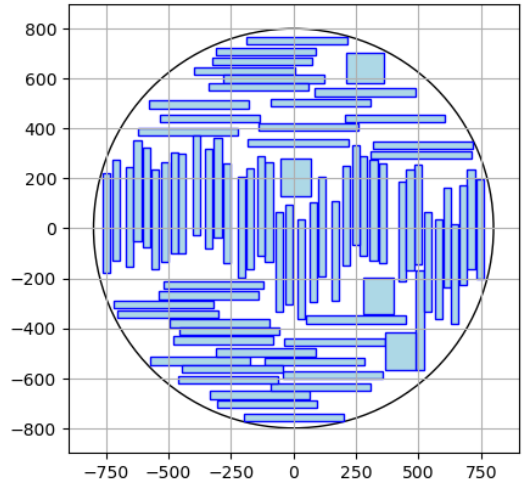

Figure 11: Przegląd najlepszego osobnika dla zadania r800.

3.4.2 r1000

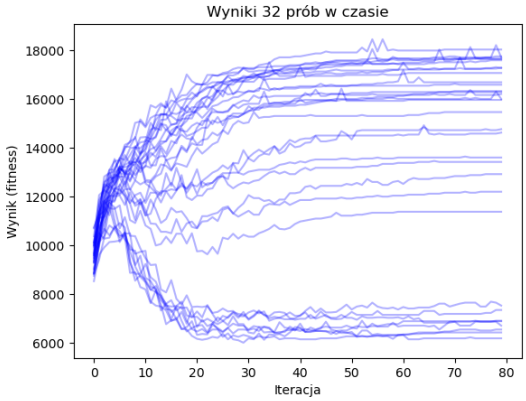

Figure 12: Przegląd grid search dla zadania r800.

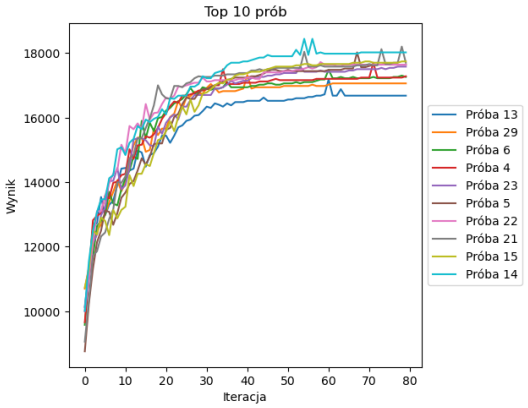

Figure 13: Przegląd najlepszych w grid search dla zadania r800.

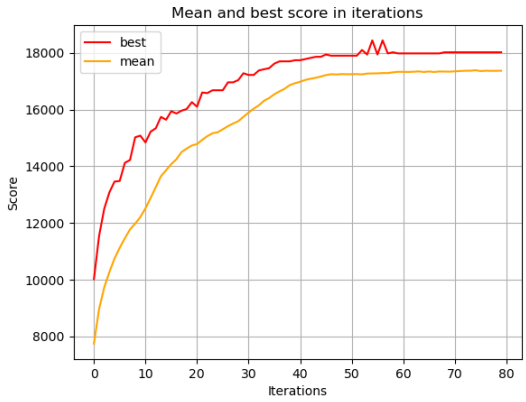

Figure 14: Przegląd najlepszego uczenia dla zadania r800.

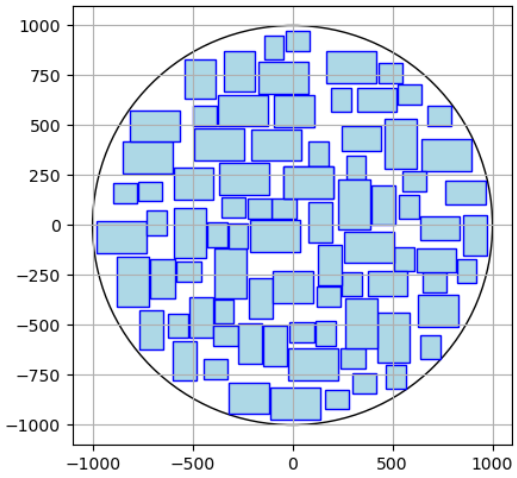

Figure 15: Przegląd najlepszego osobnika dla zadania r800.

3.4.3 r1100

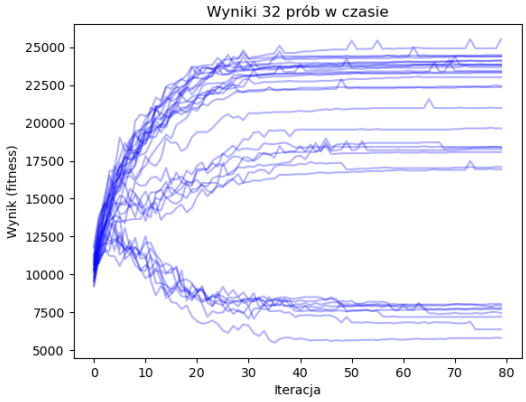

Figure 16: Przegląd grid search dla zadania r800.

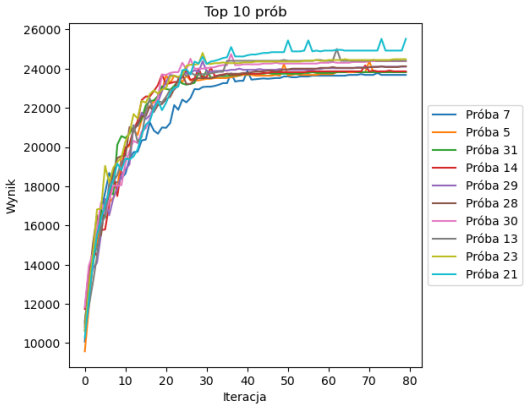

Figure 17: Przegląd najlepszych w grid search dla zadania r800.

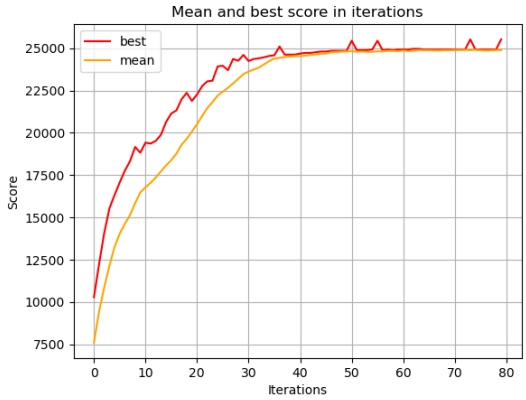

Figure 18: Przegląd najlepszego uczenia dla zadania r800.

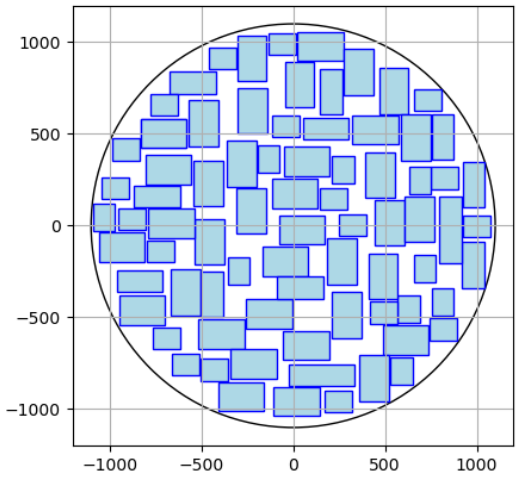

Figure 19: Przegląd najlepszego osobnika dla zadania r800.

3.4.4 r1200

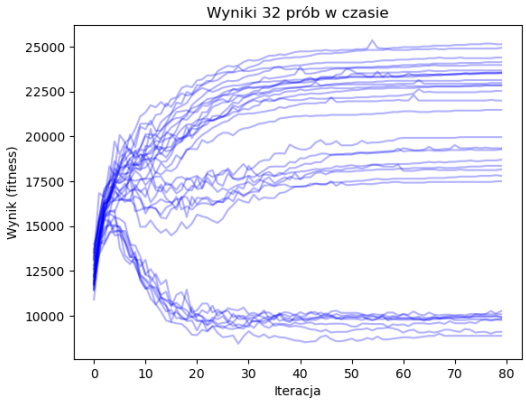

Figure 20: Przegląd grid search dla zadania r800.

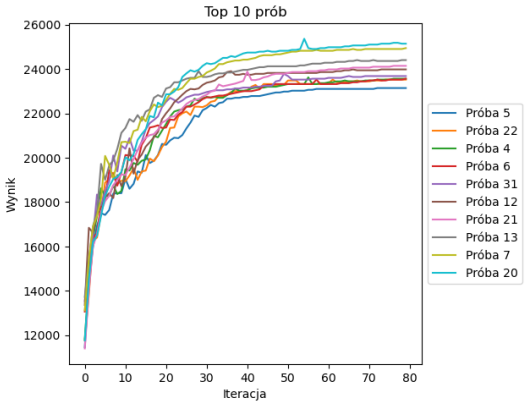

Figure 21: Przegląd najlepszych w grid search dla zadania r800.

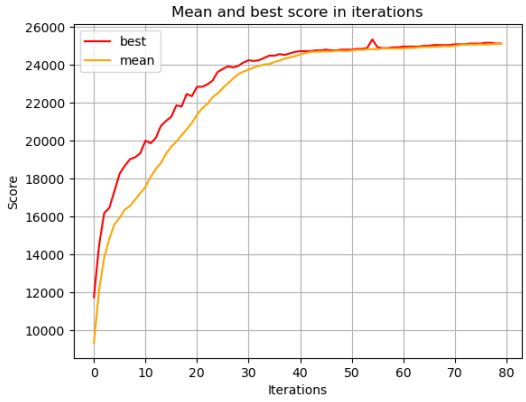

Figure 22: Przegląd najlepszego uczenia dla zadania r800.

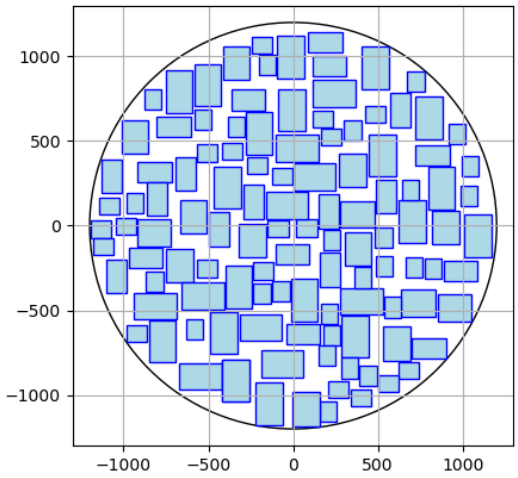

Figure 23: Przegląd najlepszego osobnika dla zadania r800.

3.5 Laboratorium 3 - algorytmy ewolucyjne w trenowaniu sieci MLP

Zaimplementowanie prostego uczenia MLP z użyciem algorytmu genetycznego. Na wejściu przyjmowana jest struktura sieci neuronowej i dane uczące. Optymalizowana funkcja to funkcja przekształcająca wektor wag sieci na błąd na zbiorze uczącym. W implementacji wykorzystano kod przygotowany wcześniej w bloku tematycznym sieci neuronowych. Mutacje oparto na losowych zmianach wag i biasów, a krzyżowanie odbywało się przez losowe przecięcie wag i biasów z rodziców.

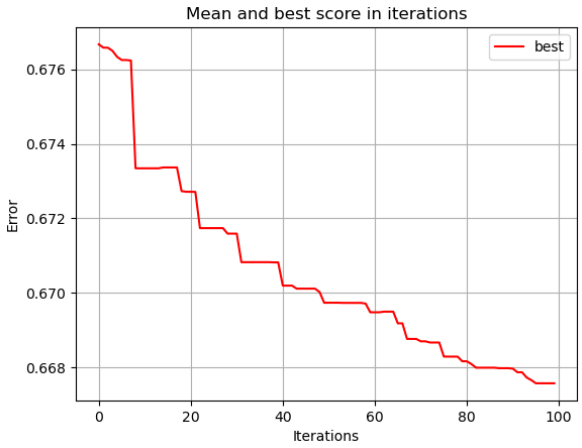

Figure 24: Proces uczenia dla zbioru Iris.

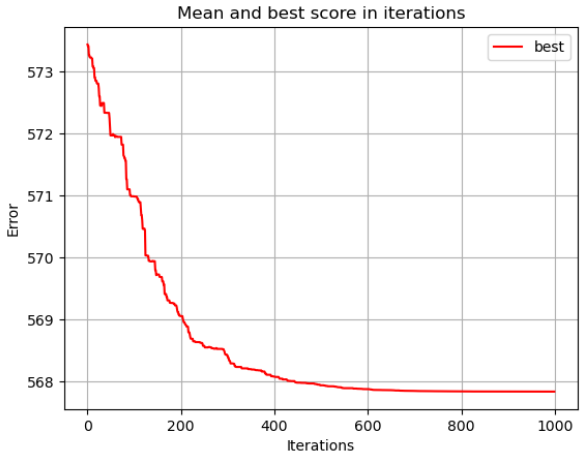

Figure 25: Proces uczenia dla zbioru auto_mpg.

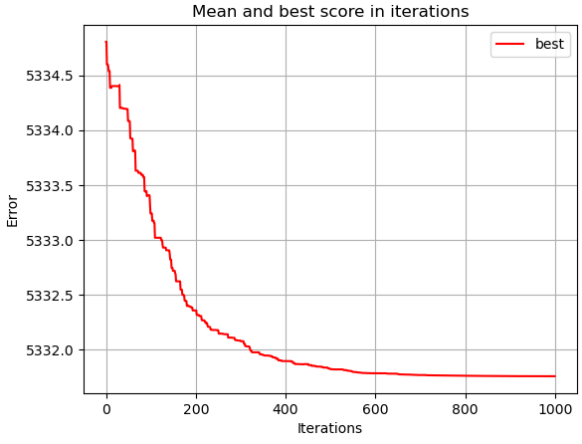

Figure 26: Proces uczenia dla zbioru multimodal large.

4 Wnioski

Implementacja z pierwszego laboratorium zawierała kilka błędów co wiązało się z osobliwą silną zależnością jakości rozwiązania od wielkości populacji. Mianowicie rozwiązania były tym gorsze im populacja była większa. W ramach pracy nad drugim laboratorium błędy te zostały namierzone i efekt zniknął.

Początkowe próby poszukiwania parametrów przez grid search dla problemu cutting stock również były obarczone błędami. Jednak część z wyciągniętych wniosków była wiarygodna. Mianowicie dominacja selekcji turniejowej, preferencja liczniejszej populacji i inicjacji prostokątów, prawdopodobieństwo mutacji w okolicach $0,5$ oraz niechęć wobec selekcji elitarnej. Nie znaleziono natomiast większego znaczenia w liczbie osobników wybieranych do porównania w selekcji turniejowej. Dalsze poszukiwania parametrów po wprowadzeniu poprawek wykazały duży wpływ szczegółowych parametrów mutacji. Przy dopuszczaniu dużej liczby prób dodawania prostokątów algorytm na siłe wciska mniejsze co blokuje go w nieoptymalnym stanie. Obniżenie parametru 'addition_variants_number' i ustawienie wyskoko 'translation_variants_number' poprawiało wyniki. Zato parametr 'max_translation_step' nie wpływa na ostateczny wynik. Jego wielkość określa jedynie tępo zbiegania do stabilnej punktacji.

Wyniki 3 laboratrium pokazują, że algorytm działa i uczy się. Z powodu jutrzejszego (11/06) egzaminu z Fizyki nie byłem w stanie znaleźć czasu na szukanie skutecznych parametrów uczenia. W związku z czym zakres tego uczenia nie był satysfakcjonujący.