ALHE – dokumentacja i sprawozdanie
Temat: SK.ALHE.6

Patryk Pankiewicz, Michał Turski

22.01.2019

1 Temat projektu

Masz 10 kart ponumerowanych od 1 do 10. Znajdź przy użyciu Algorytmu Ewolucyjnego sposób na podział kart na dwie kupki w taki sposób, że suma kart na pierwszej kupce jest jak najbliższa wartości A, a suma kart na drugiej kupce jest jak najbliższa wartości B. Należy zastosować dodatkowo inny wybrany algorytm i porównać wyniki.

2 Opis funkcjonalny

Do rozwiązania problemu został użyty algorytm ewolucyjny, wykorzystujący krzyżowanie i mutacje. Celem algorytmu jest minimalizacja różnicy. Wartości kart są przechowywane jako list i nie ma potrzeby by była ona kopiowana dla każdego osobnika - jest on reprezentowany jako lista przynależności kart do stosów. Wykorzystane metody zostały wyszczególnione poniżej.

- **Funkcja oceniająca** - musi być dostosowana do celu minimalizacji i zostały zaimplementowane jej następujące wersje:
  - suma modułów różnicy wartości kart na stosach
  - suma różnicy wartości kart na stosach podniesiona do potęgi >= 2 (jest ona parametrem algorytmu
- **Selekcja** osobników - zostały stworzone trzy wskazane w czasie wykładów z przedmiotu podejścia:
  - metoda turniejowa
  - metoda progowa
  - metoda proporcjonalna
- **Mutacja** - realizowana poprzez zmianę przynależności karty do stosu
- **Krzyżowanie** - zostały opracowane dwa rozwiązania:
  - dla każdej karty z listy losowany będzie z jednakowym prawdopodobieństwem osobnik z którego zostanie wzięta informacja gdzie ma przynależeć karta
  - wylosowanie indeks lub indeksów względem których przynależność kart zostanie zmieniona na przeciwną

3 Opis interfejsu użytkownika

Aplikacja posiada interfejs konsolowy. Argumenty aplikacji:

- -1 wartość - wartość oczekiwana dla pierwszego stosu, wymagane
- -2 wartość - wartość oczekiwana dla pierwszego stosu, wymagane
- rodzaj metody selekcji, wymagane - jeden argument z poniższych:
  - -r wartość, selekcja proporcjonalna, wartość odpowiada liczbie *beta*, która jest używana przy obliczaniu wartości prawdopodobieństwa ze wzoru $exp(\frac{-\beta \cdot z_i}{z_{max}})$, gdzie z_i odpowiada wartości celu i-tego osobnika
  - -t wartość, selekcja turniejowa, wartość odpowiada rozmiarowi szranków
  - -h wartość, selekcja progowa, wartość odpowiada wartości progu
- -m wartość - wartość odpowiada prawdopodobieństwu mutacji, wymagane
- -v wartość - wybór funkcji celu, wartość odpowiada potędze do której zostanie podniesiona różnica wartość kart na obu stosach, wymagane
- rodzaj metody krzyżowania, wymagane - jeden argument z poniższych:
  - -u - wybór właściciela dla każdej karty z osobna
  - -k wartość - wybór k punktów względem których zostanie zmieniona przynależność kart, wartość odpowiada liczbie k
- -s wartość - wartość odpowiada wartości ziarna, jeśli nie podane zostanie ustawione na losową
- -c wartość - wartość odpowiada ilości kart, wymagane
- -a wartość - wartość odpowiada rozmiarowi populacji, wymagane
- -b wartość - wartość odpowiada prawdopodobieństwu krzyżowania, wymagane
- -i wartość - wartość odpowiada ilości iteracji, wymagane
- -f wartość - wartość odpowiada prefiksowi pliku wynikowego

4 Postać plików wynikowych

Pliki wynikowe są plikami w formacie csv, z separatorem ",". Nazwa pliku wynikowego jest wartością podaną dla argumentu -f, będącego prefiksem, i argumentów programu, tak by ułatwić identyfikacje plików wynikowych.

Przykład nazwy pliku wynikowego:

```
wynik_-1_100_-2_100_-r_1_-m_0.09_-v_2_-u_-c_10_-a_30_-b_0.10_-i_200.csv
```

Program został wywołany z następującymi argumentami:

```
-1 100 -2 100 -r 1 -m 0.09 -v 2 -u -c 10 -a 30 -b 0.10 -i 200 -f wynik
```

Pozwala to w prosty sposób zidentyfikować wybrane argumenty. Program nie posiada żadnych dodatkowych plików konfiguracyjny - wszystkie parametry są ustalane za pomocą argumentów.

Dla każdej iteracji w wynikowym pliku csv zapisywane są następujące informacje:

- wartość funkcji celu dla najmniejszego osobnika
- wartość mediany populacji
- wartość średnia populacji
- wartość odchylenia standardowego populacji

5 Opis implementacji

Algorytm został zaimplementowany w języku C++ w standardzie C++14 z użyciem wyłącznie bibliotek dostępnych w standardzie. Program został zaprojektowany w sposób modularny z wykorzystaniem klas abstrakcyjnych. Główne klasy algorytmu:

- **Mutation** - klasa odpowiedzialna za przeprowadzenie mutacji z zadanym prawdopodobieństwem
- **CrossoverAlgorithm** - klasa abstrakcyjna będąca klasą bazową dla algorytmów krzyżowania
- **ScoringFunction** - klasa, która pozwala na obliczenie zadanej funkcji celu
- **SelectionAlgorithm** - klasa abstrakcyjna będąca klasą bazową dla algorytmów selekcji
- **EvolutionaryAlgorithm** - główna klasa algorytmu
- **CSVFileWriter** - klasa opowiadająca za zapis wyników do pliku csv

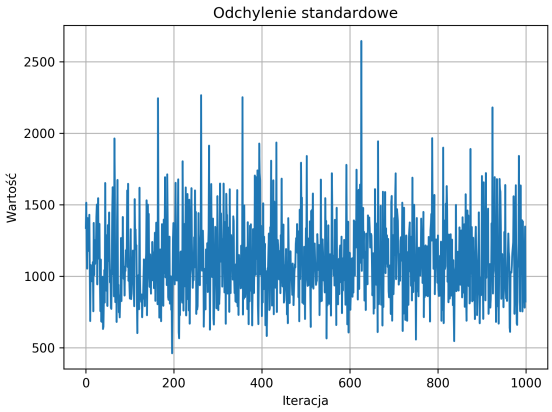
Dodatkowo do zautomatyzowania testowania i generacji wykresów z danych wynikowych został użyty język Python. Zostały użyte następujące moduły i biblioteki tego języka:

- pandas[1], biblioteka umożliwiająca łatwą obsługę plików o rozszerzeniu csv
- numPy[2], biblioteka obliczeniowa, tu użyta do obsługi tablic i prostych algorytmów
- matplotlib[3], biblioteka do generowania wykresów
- subprocess[4], moduł umożliwiający wywołanie plików wykonywalnych

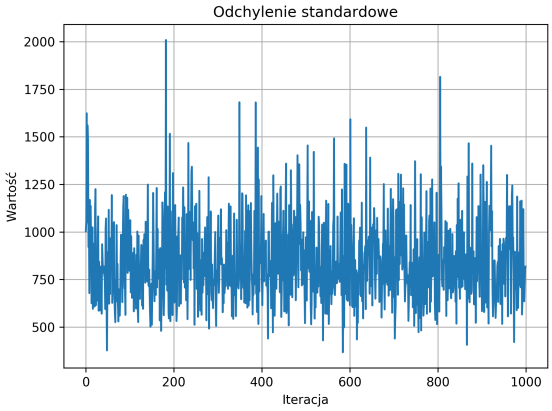
[1] https://pandas.pydata.org/
[2] http://www.numpy.org/
[3] https://matplotlib.org/
[4] https://docs.python.org/3/library/subprocess.html

6 Badania

6.1 Selekcja progowa



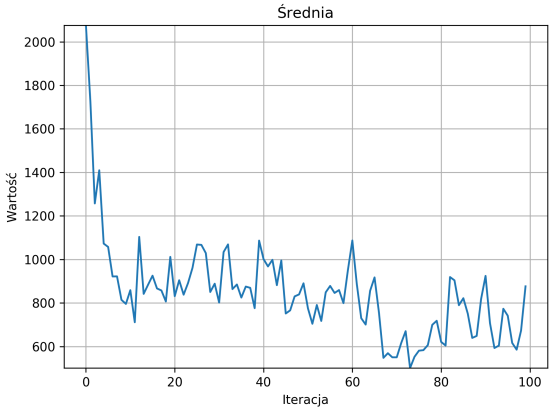
Rysunek 1: Próg ustalony na 50 najlepszych kart.



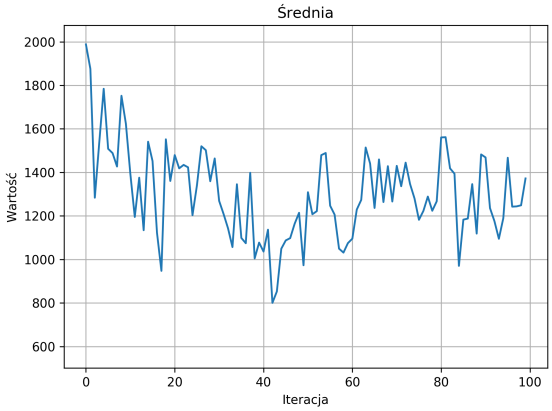
Rysunek 2: Próg ustalony na 10 najlepszych kart.

Powyższe wykresy pokazują różnice odchylenia standardowego w zależności od wybranego progu dla 100 kart. Są one zgodne z intuicją - dla większej wartości progu odchylenie standardowe wykazuje większe wahania.

6.2 Selekcja proporcjonalna

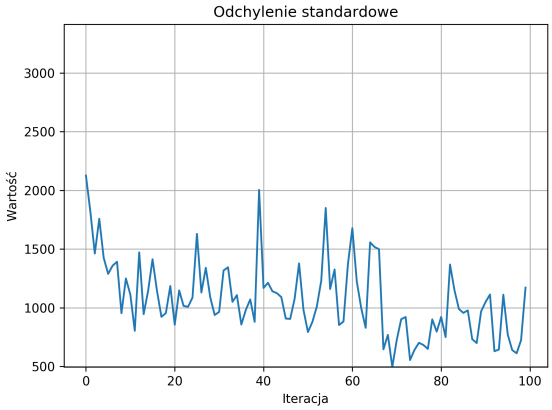


Rysunek 3: Prawdopodobieństwo krzyżowania równe 0.1.

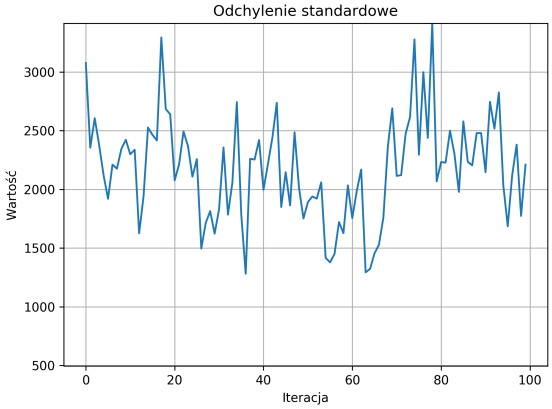


Rysunek 4: Prawdopodobieństwo krzyżowania równe 0.5.

Powyższe wykresy obrazują stopniowe zmniejszanie wartości średniej populacji. Przy małej wartości mutacji i krzyżowania proces ten dość szybko stabilizuje populacje(Rysunek 3). Natomiast dla dużego prawdopodobieństwa krzyżowania proces ten zachodzi wolniej z powodu krzyżowania lepszych osobników z gorszymi(Rysunek 4).



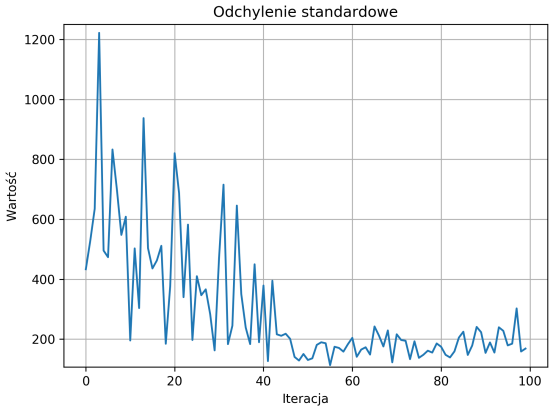
Rysunek 5: Wartość *beta* równa 1.



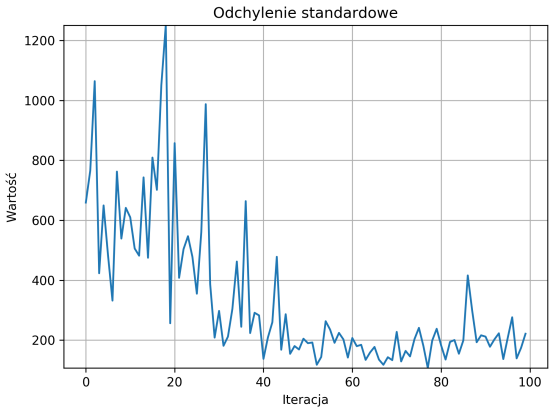
Rysunek 6: Wartość *beta* równa 0.5.

Powyższe wykresy ukazują sposób wywierania presji przy selekcji proporcjonalnej przy obliczaniu wartości funkcji celu z użyciem wzoru $exp(\frac{-\beta \cdot z_i}{z_{max}})$, gdzie z_i odpowiada wartości celu i-tego osobnika. Im jest to większa wartość tym presja selekcji jest ostrzejsza - z tych powodów populacja z wykresu pierwszego ustabilizowała się szybciej i charakteryzuje się mniejszą wartością odchylenia standardowego.

6.3 Selekcja turniejowa



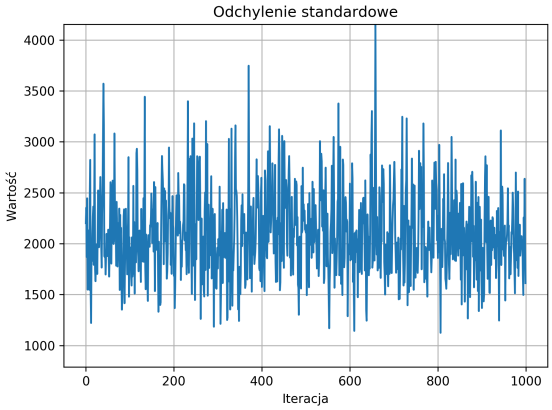
Rysunek 7: Wartość szranków równa 50% ilości kart.



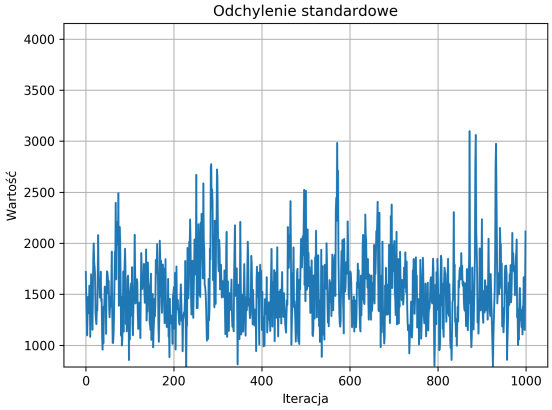
Rysunek 8: Wartość szranków równa 10% ilości kart.

Powyższe wykresy ukazują wpływ wielkości szranków na stabilizacje populacji. Przy dużej wielkości szranków często wybierane są osobniki z czołówki populacji, stąd przy małej wartości mutacji populacji silnie zbiega do wartości najlepszego osobnika. W przypadku mniejszej wielkości szranków(Rysunek 8.) proces ten nie jest tak jednostajny.

6.4 Badanie krzyżowania k-punktowego



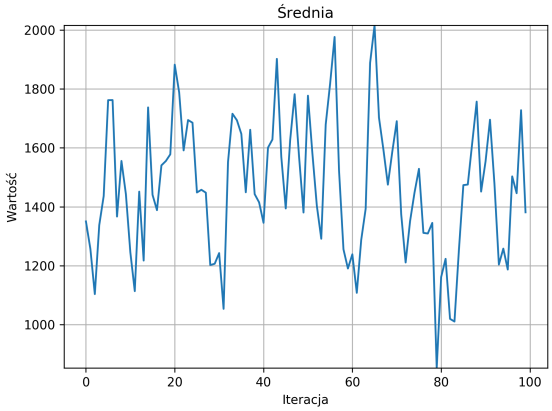
Rysunek 9: Przebieg dla krzyżowania z wyborem właściciela karty dla każdej osobno.



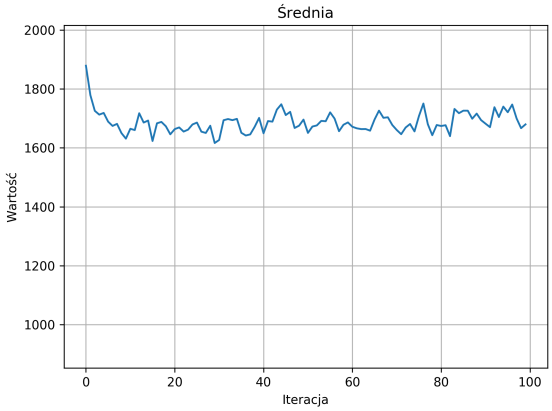
Rysunek 10: Przebieg dla krzyżowania k-punktowego, k = 5.

Jak można wywnioskować z wykresów krzyżowanie k-punktowe nie wpływa tak znacząco na poszerzanie przestrzeni przeszukiwań algorytmu jak wybór właściciela dla każego osobnika z osoba i jest to zgodne z przewidywaniami. Testy były przeprowadzone dla osobnika reprezentowanego przez 100 kart.

6.5 Rozmiar populacji



Rysunek 11: Rozmiar populacji równy 100.



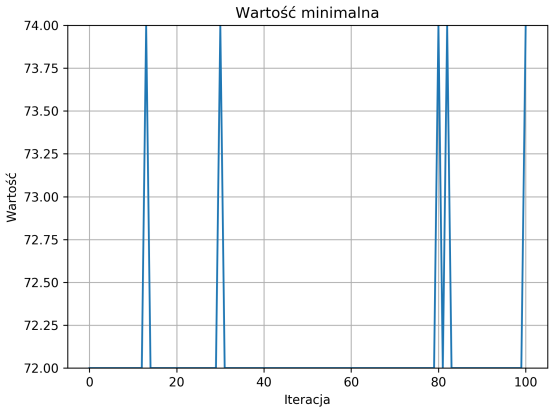
Rysunek 12: Rozmiar populacji równy 10000.

Przy badaniach została wykorzystana selekcja proporcjonalna z wartością *beta* równą 1 i prawdopodobieństwem krzyżowania równym 0.3. Duża populacja nie może zostać zaburzona w tak prosty sposób i oscyluje ona niezmiennie wokół jednej wartości.

6.6 Komentarz do wyników

Przedstawione wykresy pokazują działanie zaimplementowanych algorytmów. Populacja zachowuje się zgodnie z oczekiwaniami - wraz z iteracjami zmniejsza się wartość odchylenia standardowego i średniej. Problemem okazała się za to przestrzeń przy zbliżonych wartościach oczekiwanych dla obu stosów.

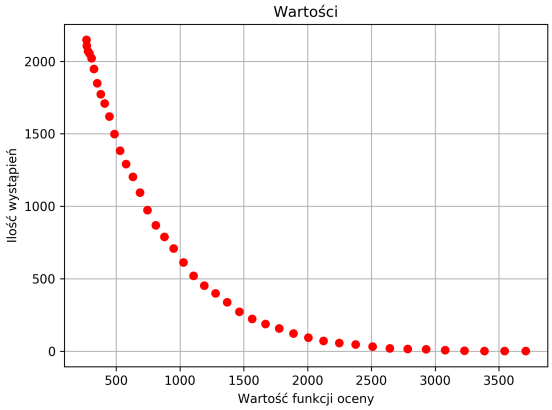
Niepokojące były wykresy przedstawiające wartość fukncji celu najepszego elementu populacji. Wyglądały one następująco:



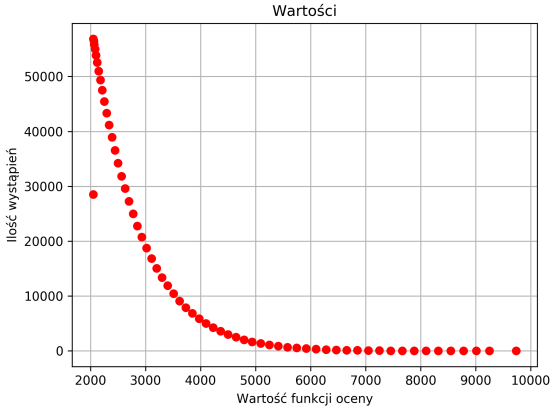
Rysunek 13: Minimlna wartość funkcji celu w zależności od numeru iteracji.

Na wykresie tym widać, że najlepsza znaleziona wartość funkcji celu znajdowała się już w populacji początkowej. Tylko nieliczne populacje nie zawierały elementu optymalnego. Jednocześnie, jak pokazują inne statystyki, takie jak średnia wartość funkcji celu czy odchylenie standardowe, populacja w tym czasie zmieniała się znacznie. Wzbudziło to tak duże wątpliwości, że zdecydowano się na dokładniejsze zbadanie przeszukiwanej przestrzeni.

Badania przestrzeni realizowano poprzez przejście po całej przestrzeni i zapisanie poszczególnych uzyskanych wyników i tego, jak często one występowały. Dla małych problemów(do 20 kart) problem taki jest rozwiązywany w wystarczająco krótkim czasie. Uzyskane wyniki zaprezentowano na poniższych wykresach:



Rysunek 14: Ilość kart równa 15, wartość na pierwszym stosie 25, a na drugim 30. Wartości kart: 8, 3, 4, 4, 10, 1, 8, 4, 2, 6, 9, 10, 1, 4, 4.



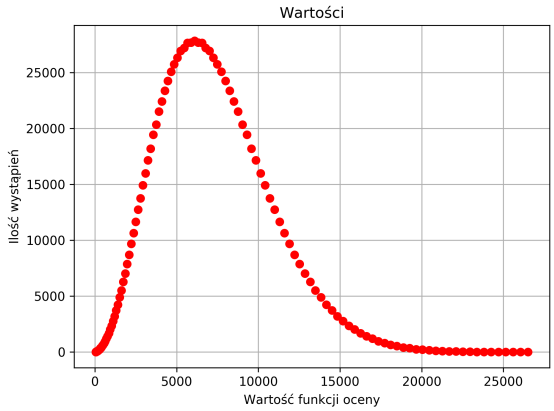
Rysunek 15: Ilość kart równa 20, wartość na pierwszym stosie 25, a na drugim 30. Wartości kart: 4, 4, 10, 9, 8, 8, 9, 4, 4, 5, 8, 7, 5, 5, 4, 3, 6, 6, 2, 8.

Jak widać, wraz ze zmniejszaniem się wartości funkcji celu zwiększa się liczba wystąpień danej oceny. Co ciekawe w przypadku ukazanym na rysunku nr. 13 możliwych do uzyskania było tylko 42 różnych wartości funkcji celu, a w przypadku drugim 62. Są to liczby niespodziewanie małe, lecz wynikają one z wartości kart i sposobu oceny.

Jak można zauważyć, rozwiązanie optymalne lub bliskie optymalnego stanowią lwią część wszystkich wartości i stanowią w zależności od przyjęcia górnej granicy nawet do 30% wszystkich możliwości, stąd dla populacji o

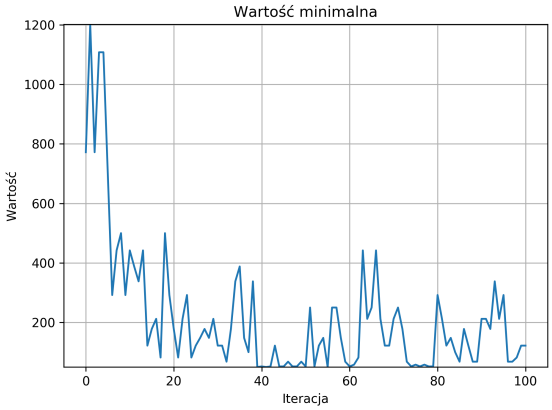
liczności równej np. 10 nie sposób by nie została wylosowana jedna z takich wartości.

Zupełnie inaczej przedstawia się przestrzeń przy zadaniu, w którym wartości kart na stosach są zupełnie różne. Aby to pokazać, przeanalizowano przestrzeń dwudzestokartową w której na pierwszym stosie oczekujemy wartości 110, na drugim zaś 0. Wyniki zaprezentowano na poniższym histogramie:



Rysunek 16: Ilość kart równa 20, wartość na pierwszym stosie 110, a na drugim 0. Wartości kart losowe.

W tej "trudnej" przestrzeni wykresy przedstawiające zbieganie się wartości minimalnej w iteracji przedstawiają się ciekawiej:



Rysunek 17: Minimlna wartość funkcji celu w zależności od numeru iteracji, problem "trudny".

W tym przypadku widać, że początkowa populacja nie zawiera dobrych rozwiązań. Dopiero po około 40 iteracjach udaje się znaleźć rozwiązanie bliskie optymalnemu. Dzieje się tak dlatego, że dobrych rozwiązań jest w tej przestrzeni znacznie mniej, niż w przestrzeni "łatwej", a co za tym idzie "trafienie" do optimum przy pomocy krzyżowania i mutacji jest znacznie mniej prawdopodobne.

7 Podsumowanie

Do rozwiązanie problemu został zaimplementowany poprawny algorytm ewolucyjny z wieloma możliwościami parametryzacji. Przy wybranej funkcji celu i zadanej przestrzeni nie można poprawnie zobrazować działania algorytmu przy znajdowaniu najmniejszej wartości. Sytuacja mogłaby ulec zmianie w przypadku dodania dodatkowych celów do spełnienia np. podobnej ilości kart lub jak najmniejszego odchylenia standardowego ich wartości.