

Optymalizacja trasy wiertła dla płytki PCB

Opis projektu

Podstawowe (ogólne) założenia projektu:

- Cel: stworzenie aplikacji, która będzie wyznaczała trasę końcówki robota (w której umieszczone jest wiertło). Robot ma za zadanie wywiercić zaznaczone otwory o zadanej średnicy na powierzchni płytki PCB.
- Na powierzchni płytki możliwe jest wywiercenie otworów o różnych średnicach - dlatego też konieczna będzie zmiana średnicy wiertła (przezbrojenia).
- Koszt (czas) jest proporcjonalny do odległości dwóch kolejnych otworów między którymi ma przemieścić się kocówka robota.
- Ramię robota porusza się wyłącznie prostoliniowo między dwoma kolejnymi punktami, w których następuje wiercenie otworu.
- Każde przebrojenie (zmiana wiertła na wiertło o innej średnicy) związane jest z poniesieniem określonego kosztu.
- Czas wiercenia każdego z otworów (niezależnie od średnicy) przyjmujemy jednakowy.
- Możliwe jest narzucenie częściowej kolejności (na przykład ze względów wytrzymałościowych). Naruszenie częściowej kolejności związane jest z poniesieniem kary o bardzo dużej wartości.

Przezbrojenia (zmiana średnicy wiertła) - założenia:

- Przebrojenie wykonywane jest nad wierconym otworem (zakładamy, że robot dysponuje kocówką obrotową, z zamontowanymi niezbędnymi wiertłami - o odpowiednich średnicach zewnętrznych).
- W trakcie zmiany wiertła na wiertło o innej średnicy kocówka robota pozostaje nieruchoma (nad położeniem otworu).
- Zmiana średnicy wiertła dokonywana jest od razu po wywierceniu poprzedzającego otworu.
- Proces zmiany wiertła wymaga określonego czasu - przy każdorazowej zmianie zostaje poniesiony pewien koszt przebrojenia.

Częściowa kolejność - założenia:

- Wyznaczenie częściowej kolejności polega na wybraniu pewnego podzbioru punktów z całego zbioru punktów określających położenie otworów do wiercenia. Dla wybranego podzbioru określona zostaje kolejność, która nie powinna zostać naruszona.
- Naruszenie częściowej kolejności związane jest z poniesieniem znacznej kary.
- Podczas wyznaczania optymalnej trasy między punktami ze zbioru związanego z kolejnością częściową mogą pojawić się punkty nienależące do tego zbioru. Istotne jest tylko, aby w rozwiązaniu końcowym nie została zaburzona narzucona kolejność (punkty z tego zbioru nie muszą więc występować kolejno - jeden po drugim).

Model matematyczny problemu

n - liczba otworów do wywiercenia

$[c_{ij}]_{n \times n}$ - macierz $n \times n$, gdzie c_{ij} oznacza czas przemieszczenia wiertła z punktu i do punktu j :

- droga wyznaczona jako: $s_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$
- $c_{ij} = \begin{cases} \frac{s_{ij}}{v}, & v = \text{const} \quad \text{gdy } i \neq j \\ \infty & \text{gdy } i = j \end{cases}$

$A = [a_{ij}]_{n \times n}$ - macierz $n \times n$, gdzie a_{ij} oznacza konieczność przebrojenia

- $a_{ij} = \begin{cases} 1 & \text{gdy } \phi_i \neq \phi_j \\ 0 & \text{gdy } \phi_i = \phi_j \end{cases}$
- gdy w macierzy występuje 1 (przebrojenie), to podczas obliczeń pobierana jest wartość kosztu przebrojenia (dostępna globalnie)

Rozwiązania problemu przechowywane są w postaci permutacji numerów otworów. Ze względu na konieczność określania poprawności częściowej kolejności zakładamy, że otwór z numerem 1 jest wykonywany w pierwszej kolejności.

m - liczba otworów z narzuconą częściową kolejnością

$S = [s_i]_m$ - wektor przechowujący numery otworów z narzuconą częściową kolejnością

$G = [g_i]_n$ - permutacja numerów otworów (otwór z numerem 1 znajduje się na pierwszej pozycji)

Funkcja celu:

$$f_c(x, y) = \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n (c_{ij}x_{ij} + a_{ij}D) + K \sum_{i=1}^{m-1} p_i \rightarrow \min$$

$$p_i = \begin{cases} 1 & \text{gdy } k < j : \{s_i = g_j, s_{i+1} = g_k\} \\ 0 & \text{gdy } k > j : \{s_i = g_j, s_{i+1} = g_k\} \end{cases}$$

- $x_{ij} \in \{0; 1\}$ - decyduje o przejściu z otworu i do j
- $a_{ij} \in \{0; 1\}$ - decyduje o przebrojeniu podczas zmiany otworu i do j (na podstawie macierzy A)
- D - koszt przebrojenia (jest stały, nie zależy od średnic otworów)
- K - wartość kary za zaburzenie narzuconej kolejności otworów

Ograniczenia:

$$\left. \begin{array}{l} \sum_{\substack{i=1 \\ i \neq j}}^n x_{ij} = 1 \\ \sum_{\substack{j=1 \\ j \neq i}}^n x_{ij} = 1 \end{array} \right\} \text{tylko 1 raz może być osiągnięty dany otwór (problem OTSP - brak cykli)}$$

Opis algorytmu

Do rozwiązania problemu przedstawionego w projekcie wykorzystaliśmy metodę opartą na algorytmie ewolucyjnym. Poprzez wykorzystanie mechanizmów ewolucji oraz doboru naturalnego algorytm ten przeszukuje przestrzeń alternatywnych rozwiązań w poszukiwaniu rozwiązania optymalnego (suboptymalnego).

Podstawowe parametry algorytmu:

n – wielkość populacji

p – stosunek ilości rodziców do wielkości populacji

m – prawdopodobieństwo mutacji

g – ilość pokoleń

Proces poszukiwania rozwiązania zaczynamy od wygenerowania losowej populacji – n osobników. Każdy osobnik posiada materiał genetyczny – permutację kolejności otworów. Następnie populacja jest poddawana ocenie zgodnie z przedstawioną powyżej funkcją celu. Kolejnym krokiem jest wybór $n \cdot p$ (przy czym musi być to liczba parzysta większa niż 1) osobników do reprodukcji – rodziców. Selekcji możemy dokonać za pomocą jednej z dwóch zaimplementowanych metod:

Metoda koła ruletki

W metodzie tej tworzymy wirtualne koło, którego wycinki odpowiadają poszczególnym osobnikom. Każdy osobnik otrzymuje wycinek proporcjonalny do jego wartości oceny. Prawdopodobieństwo wylosowania osobnika jest tym większe im większy jest odpowiadający mu wycinek koła.

Metoda rankingowa

Na początku szeregujemy osobniki niemalejąco w stosunku do funkcji oceny. Następnie wybieramy $n \cdot p$ najlepszych rozwiązań.

Po wybraniu rodziców przystępujemy do etapu reprodukcji. Osobniki krzyżowane są ze sobą za pomocą jednego z wybranych operatorów. Do dyspozycji mamy 3 różne operatory:

- PMX
- OX
- CX

W wyniku działania powyższych operatorów z dwóch osobników rodziców otrzymujemy dwoje osobników potomków. Następnie są one poddawane losowej mutacji z prawdopodobieństwem m . Dostępne operatory mutacji:

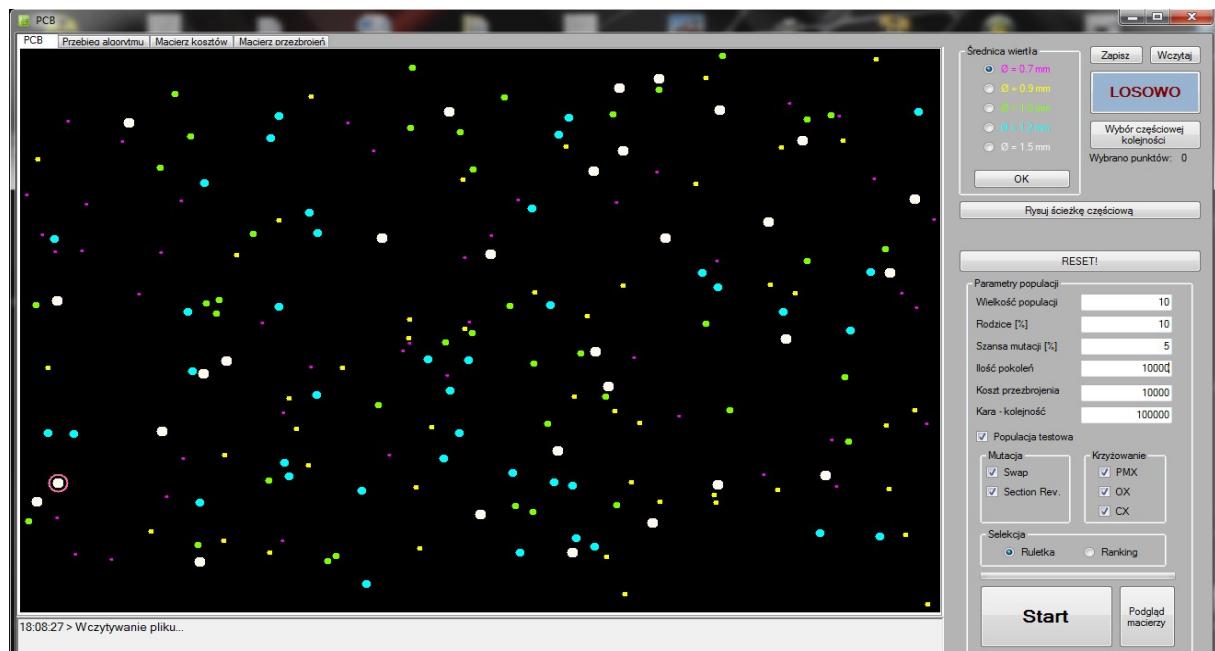
- zamiana dwóch elementów w permutacji
- zamiana kolejności elementów permutacji między dwoma losowo wybranymi pozycjami

Po zakończeniu mutacji nowe osobniki poddawane są ocenie. Do kolejnej populacji wchodzą wszystkie nowe rozwiązania oraz $n \cdot (1 - p)$ najlepszych osobników z populacji poprzedniej. Następnie począwszy od kroku selekcji powtarzamy kolejne kroki g razy.

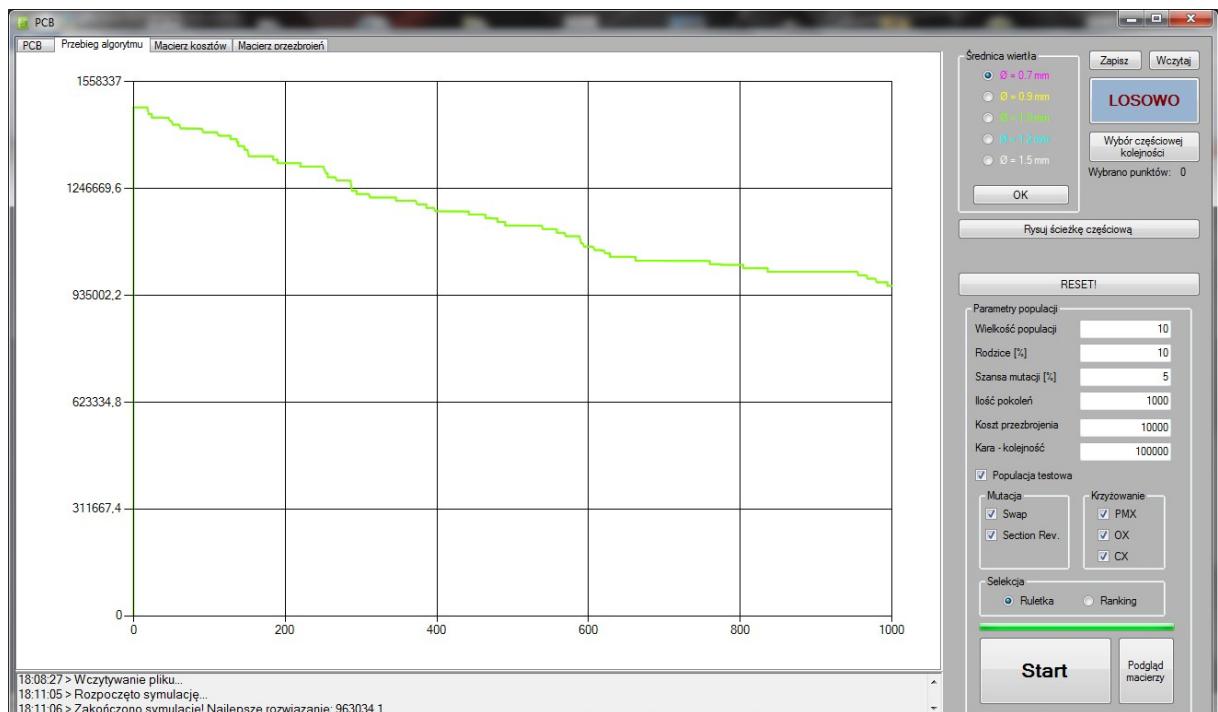
Omówienie parametrów algorytmu

Algorytm wykonuje obliczenia w oparciu o parametry, które można dowolnie zmieniać. Każdy z nich w jakiś sposób oddziałuje na przebieg algorytmu. Wpływ poszczególnych parametrów na pracę został omówiony poniżej.

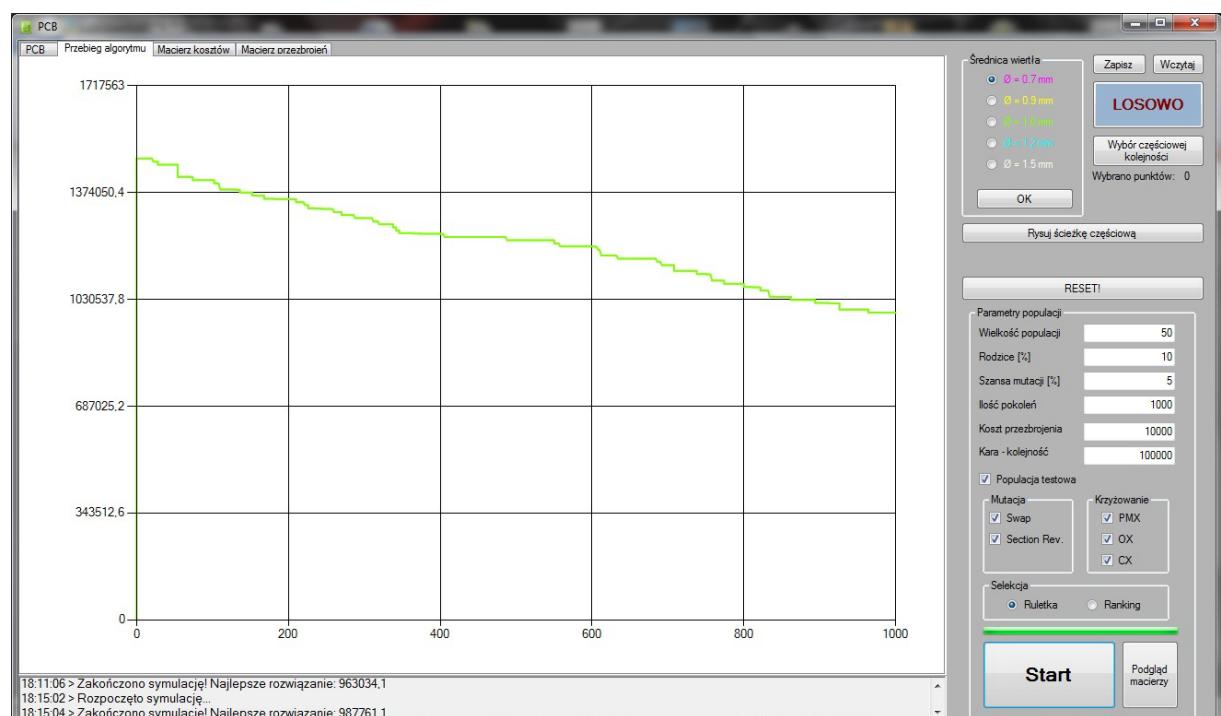
Dla losowej instancji początkowej składającej się z 200 otworów wygenerowane zostało przy użyciu parametrów domyślnych rozwiązanie, które posłuży nam jako model odniesienia, do porównania z innymi przebiegami.



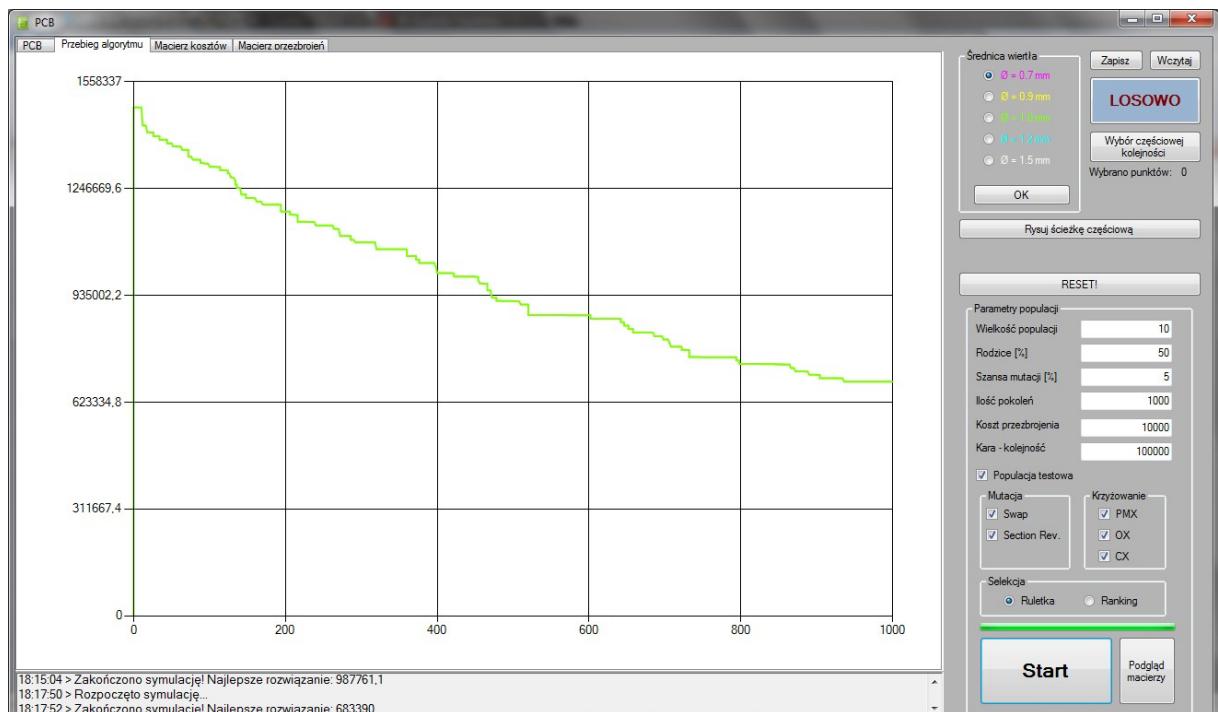
Przebieg funkcji celu:



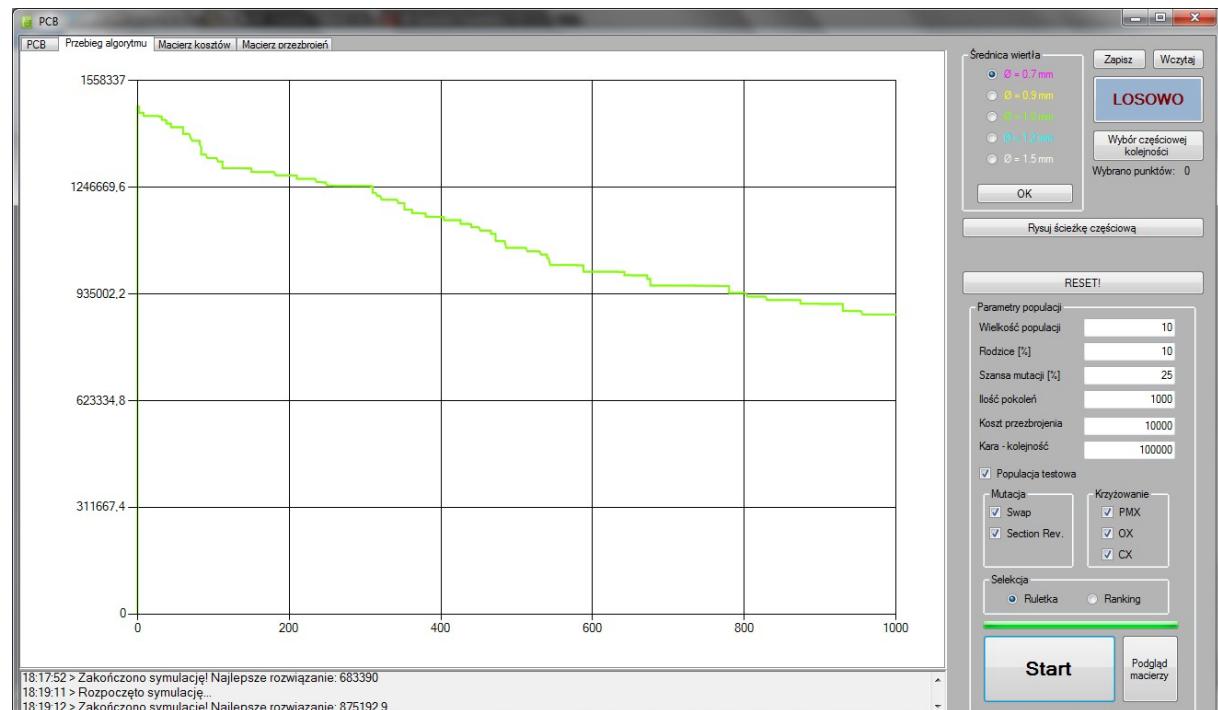
Przebieg funkcji celu dla zwiększonej populacji:



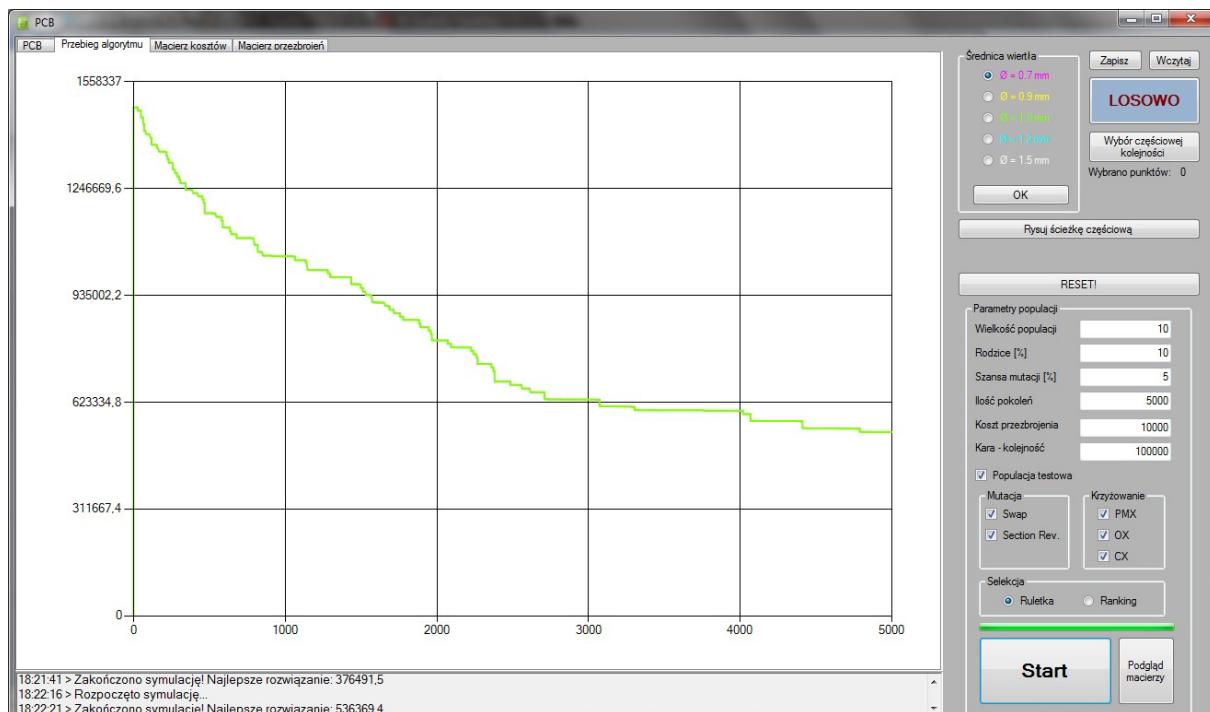
Przebieg funkcji celu dla zwiększonej ilości rodziców:



Przebieg funkcji celu dla zwiększonej szansy mutacji:



Przebieg funkcji celu dla zwiększonej ilości pokoleo:



Porównanie rozwiązań w zależności od parametrów	
Parametry domyślne	963034
Zwiększona populacja	987761
Zwiększona ilość rodziców	683390
Zwiększona szansa mutacji	875193
Zwiększona ilość pokoleń	536369
Koszt przebrojenia	268306

Wnioski

Jak wynika z porównania, największy wpływ na rozwiązanie miał koszt przebrojenia. Wynika to z bezpośredniego wpływu kosztu przebrojenia na rozwiązanie. Mając do dyspozycji 5 średnic wiertła, musimy wykonać co najmniej 4 przebrojenia. W praktyce, przy tworzeniu płyt PCBA redukcja czasu przebrojenia byłaby bardzo korzystna, przy produkcji na większą skalę.

Z pozostałych parametrów najbardziej znaczącym jest ilość pokoleń. Zwiększa ona złożoność czasową algorytmu, ale efekty są tego warte. Wynika to z podstawowej własności algorytmów ewolucyjnych; im więcej pokoleń tym więcej powstanie krzyżówek różnych osobników, populacja stale się poprawia.

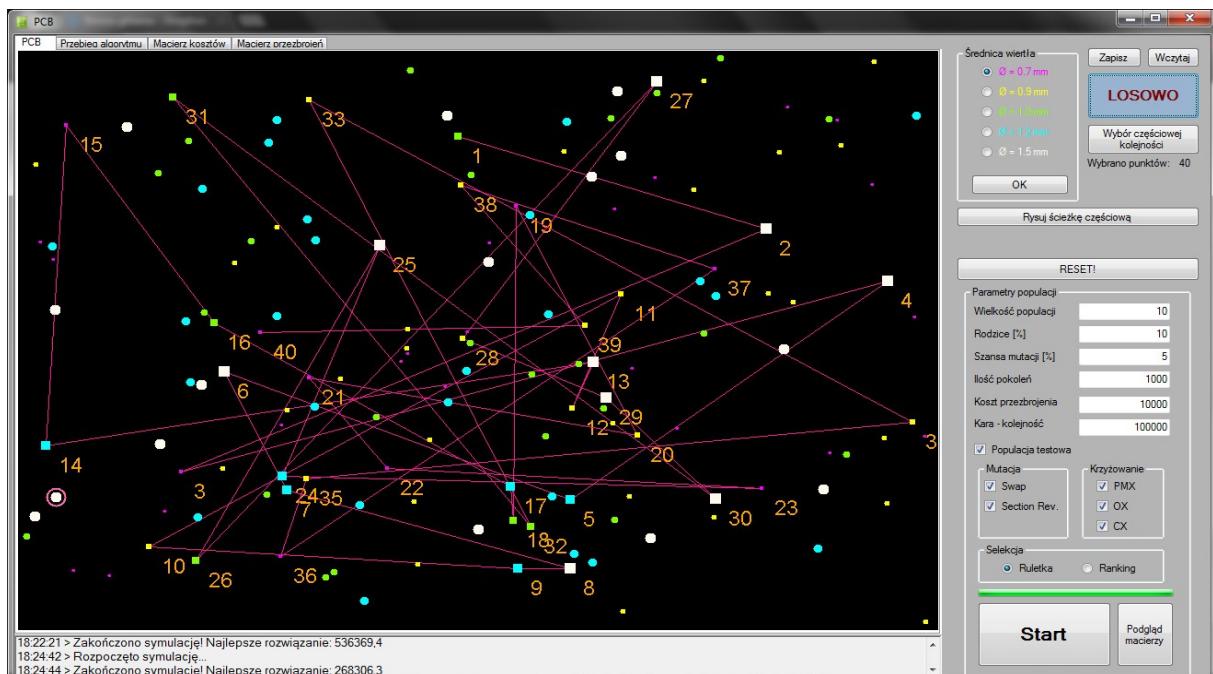
Kolejnym bardzo ważnym czynnikiem jest ilość rodziców biorących udział w tworzeniu potomków. Im więcej rodziców tym więcej nowych osobników, którzy mogą wnieść bardzo korzystne zmiany do populacji. Zwiększona ilość rodziców zwiększa złożoność pamięciową algorytmu, ale daje wspierające korzyści.

Mutacje mają wpływ na rozwiązanie najlepsze, ale jest on niewielki w stosunku do wcześniej omówionych parametrów, wielkość populacji wydaje się nie mieć wpływu w przeprowadzonym doświadczeniu, warto natomiast pamiętać, że zbytnia jej redukcja może doprowadzić do takiego zawężenia puli genetycznej, że nie uda się uzyskać krzyżówek, które spowodują znaczącą poprawę funkcji celu.

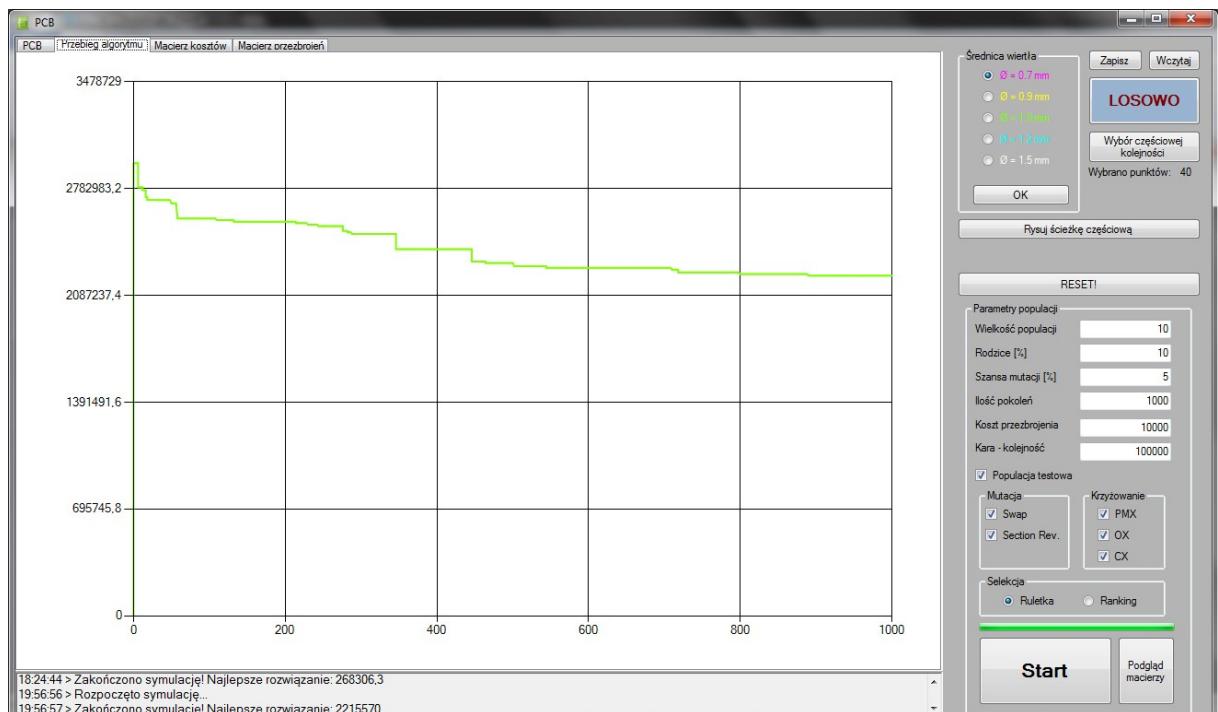
Funkcja kary

Funkcja kary została wprowadzona w celu zapewnienia częściowej kolejności wiercenia wybranych otworów, w przypadku zaburzenia tej kolejności wartość funkcji celu ulega znacznemu pogorszeniu.

Działanie funkcji celu na przykładzie (150 otworów, 40 punktowy podcykl), Instancja:

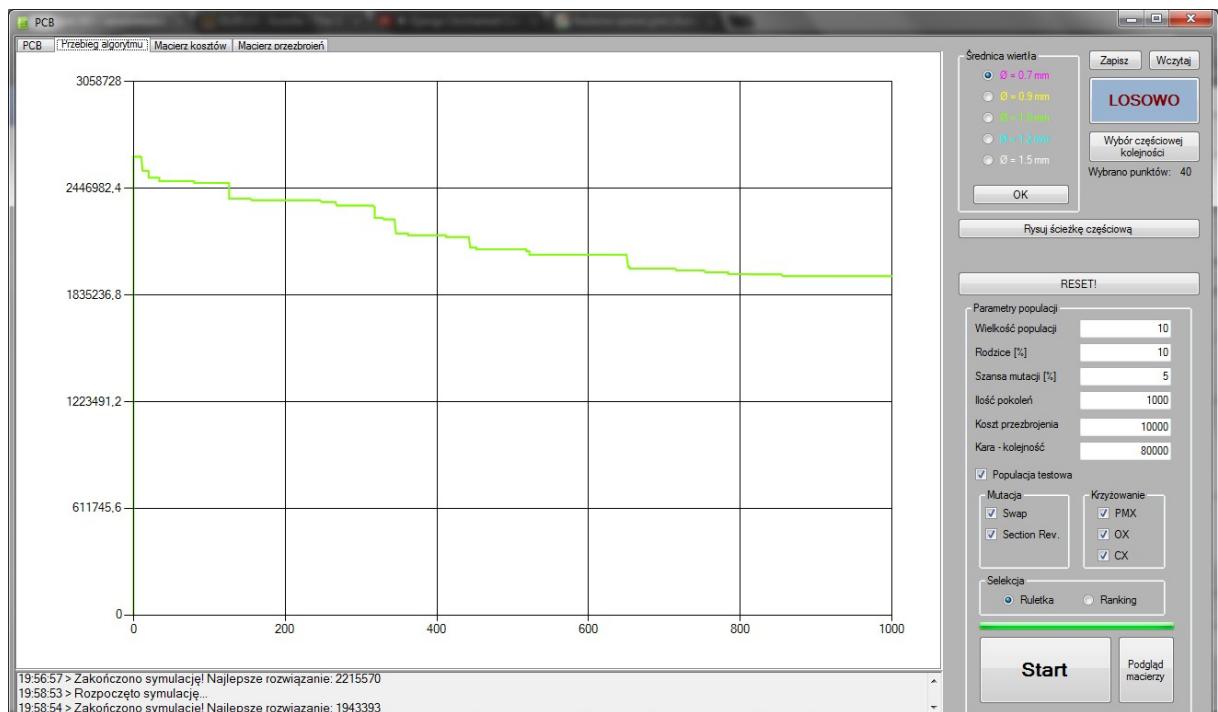


Dla wysokiego współczynnika kary (100000) algorytm wypromuje rozwiązania nie zakłócające kolejności częściowej, ponieważ jej zachowanie mimo kosztów drogi i przebrojenia będzie tańsze:



Przykładowy przebieg przedstawia konsekwencję takiego działania, wartość funkcji celu jest wysoka. Zmniejszając karę powodujemy, że w niektórych przypadkach algorytm złamie kolejność na rzecz zysku ze skrócenia drogi, lub ograniczenia ilości przezbrojeo.

Wartość funkcji celu osiągnęła wartość korzystniejszą:



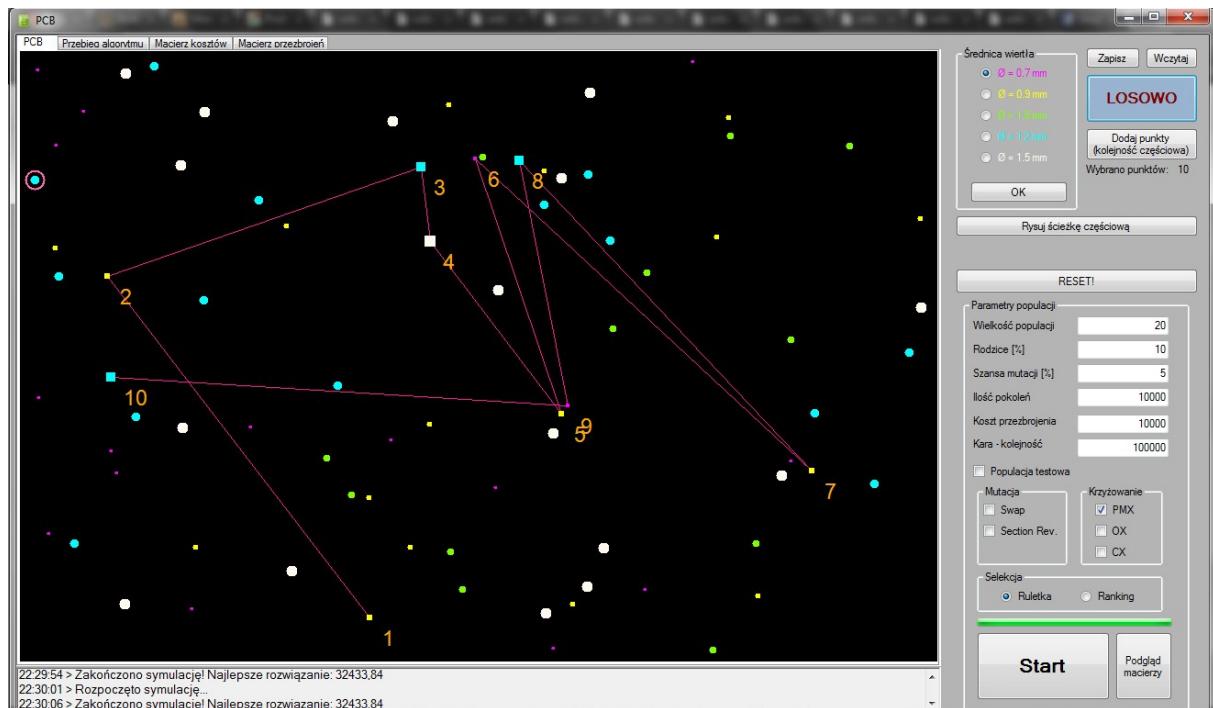
Należy jednak pamiętać, że nadmierne obniżenie kary spowoduje dużą redukcję czasu wykonania płytki, ale może nie mieścić się w wymogach technologicznych, o których nie możemy zapomnieć w przypadku zadań praktycznych.

Test operatorów krzyżowania

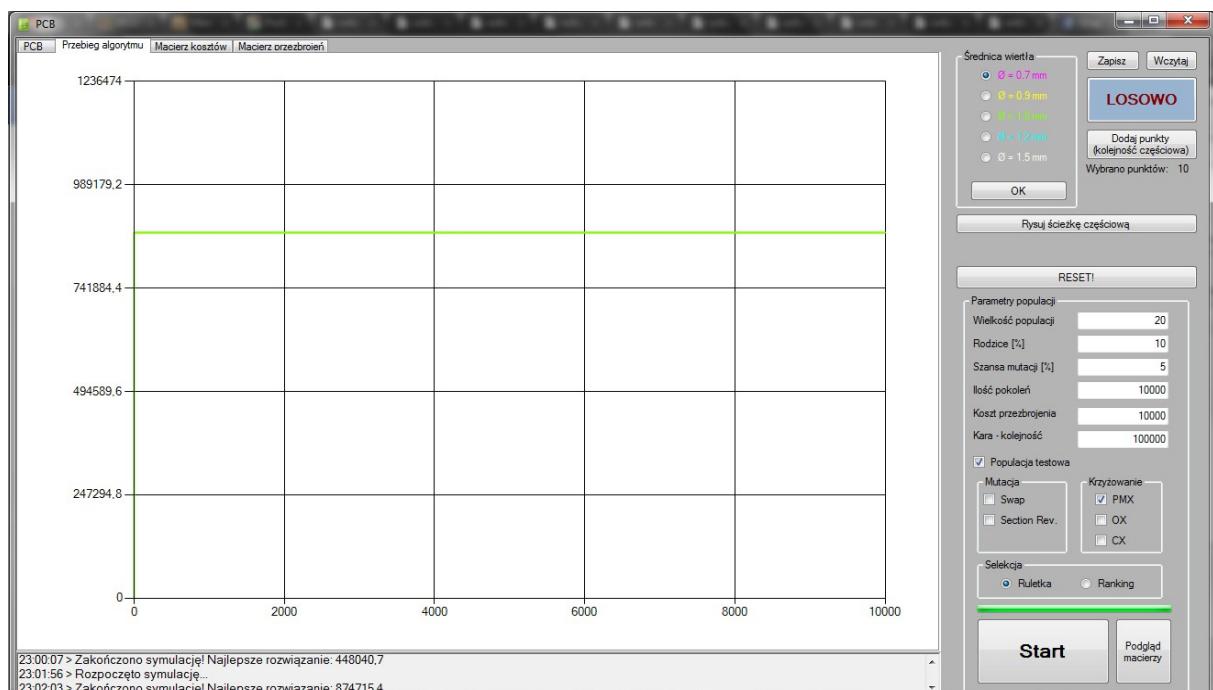
Na losowo utworzonej instancji (80 otworów, 10 punktowy podcykl) przetestowaliśmy operatory krzyżowania, najlepszym z nich okazał się operator OX. Operatory PMX oraz CX słabo sprawdziły się w tym problemie.

Testy przeprowadzone były dla identycznych rozwiązań początkowych.

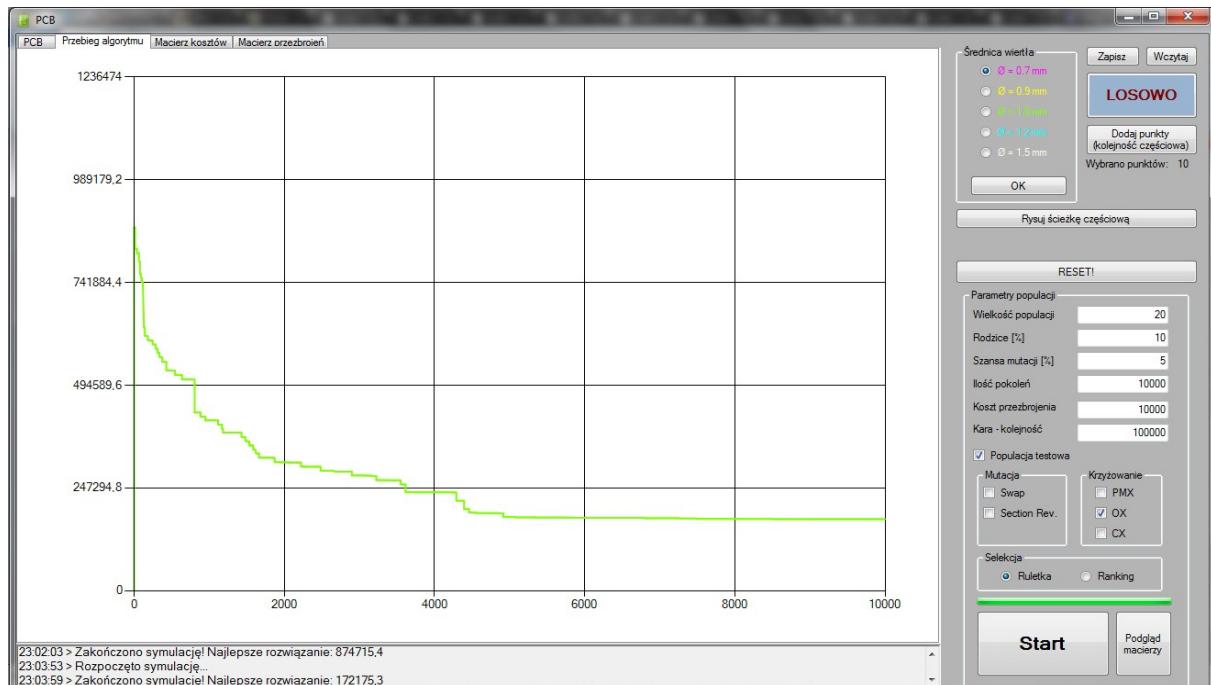
Instancja:



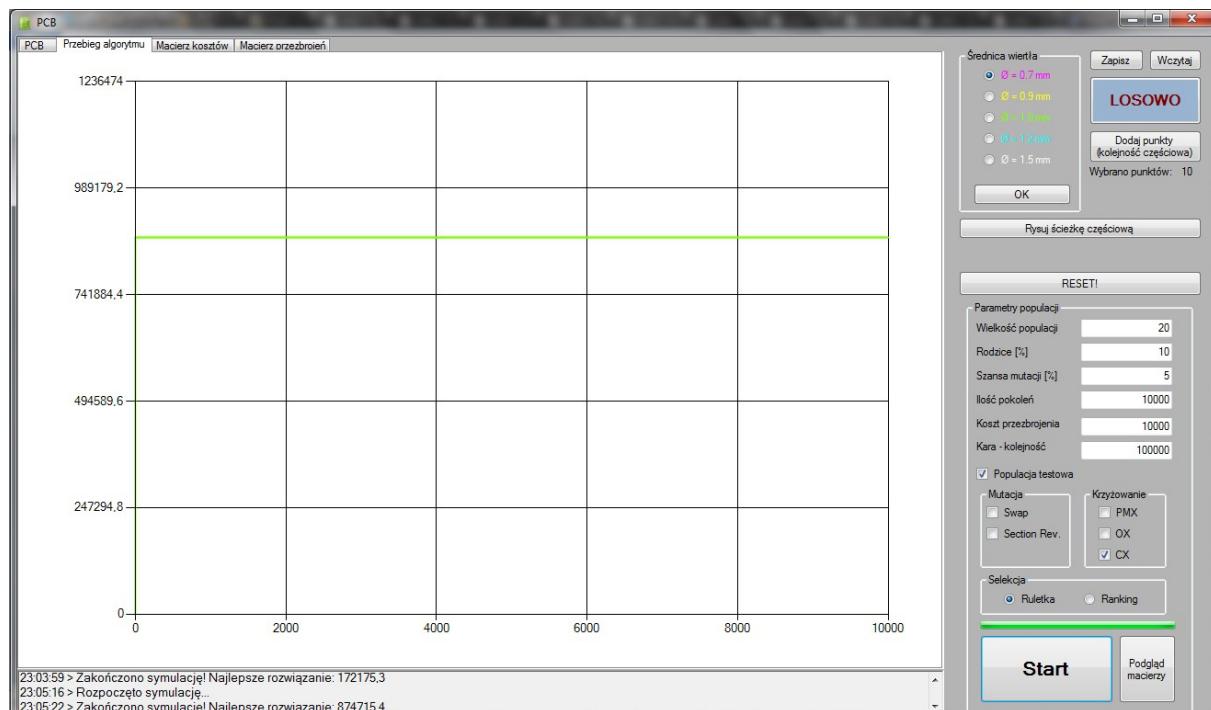
Przebieg pracy algorytmu przy użyciu tylko operatora PMX:



Przebieg pracy algorytmu przy użyciu tylko operatora OX:



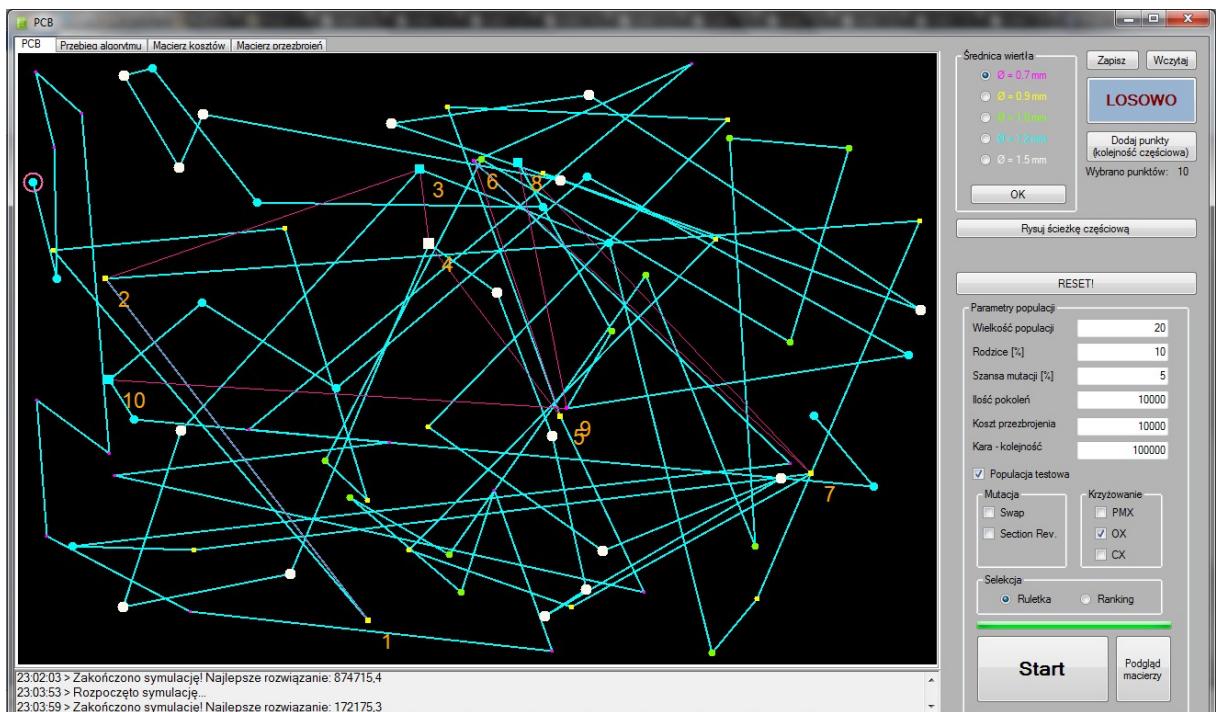
Przebieg pracy algorytmu przy użyciu tylko operatora CX:



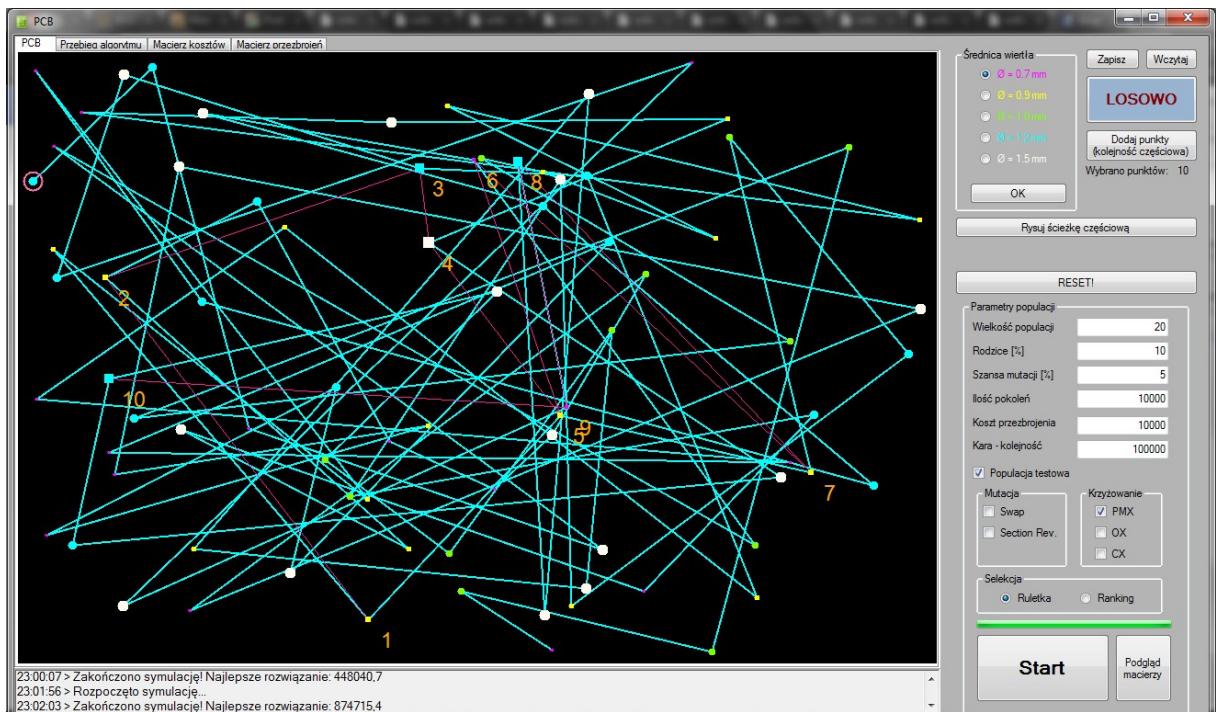
Jak widać z powyższych przebiegów najlepszy jest operator OX, zapewnia szybką zbieżność do rozwiązania najlepszego.

Rozwiązania uzyskane przez operatory PMX oraz CX w dużej mierze zależą od rozwiązania początkowego.

Rozwiązanie uzyskane przy użyciu metody OX:



Dla porównania rozwiązanie otrzymane dzięki metodzie PMX:



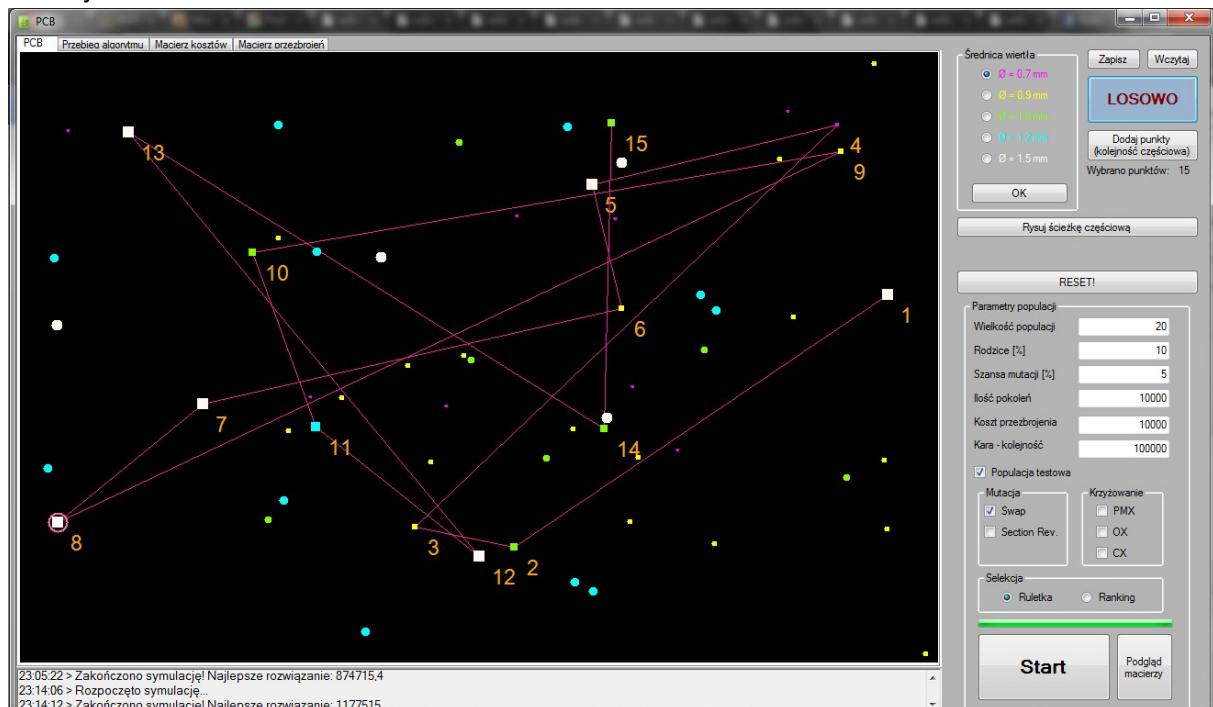
Rozwiązanie metody PMX jest bardziej „zagmatwane” niż rozwiązanie metody OX, co jest potwierdzone wynikami, 874715 dla PMX i tylko 172175 dla OX

Testy mutacji

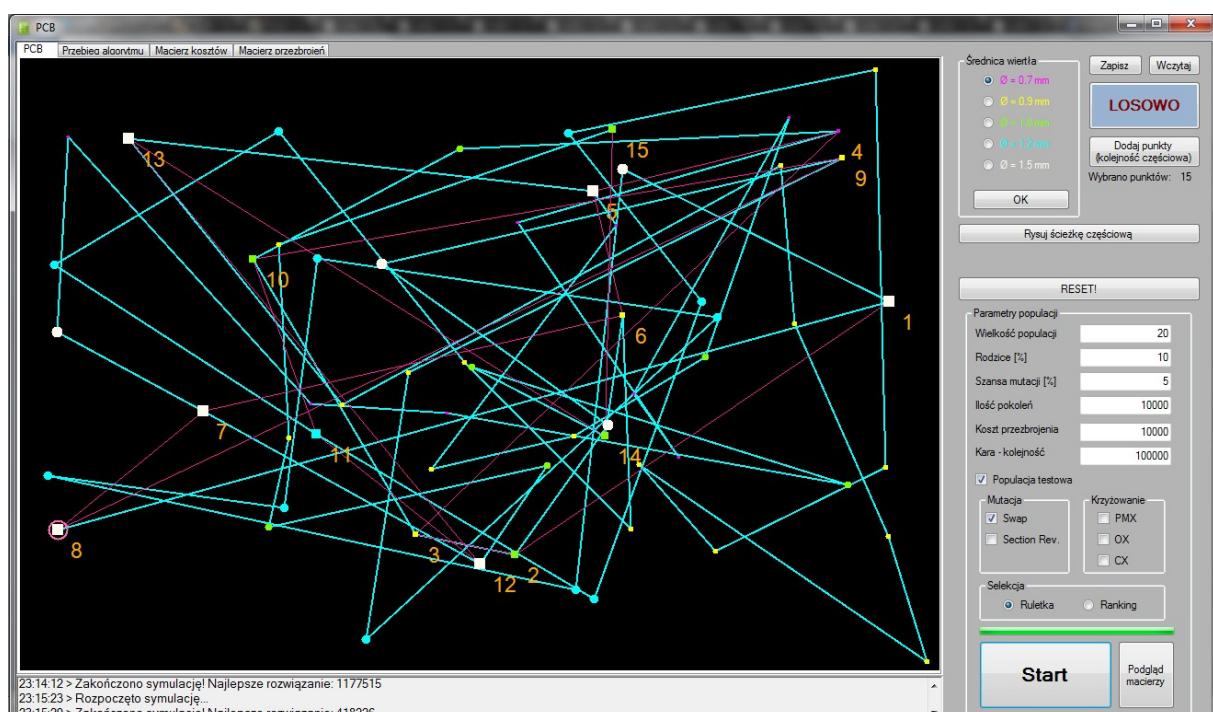
Na losowo utworzonej instancji (60 otworów, 15 punktowy podcykl) przetestowaliśmy operatory mutacji, spośród operatora Swap i mutacji poprzez inwersję, lepszy okazał się operator Swap.

Testy przeprowadzone były dla identycznych rozwiązań początkowych.

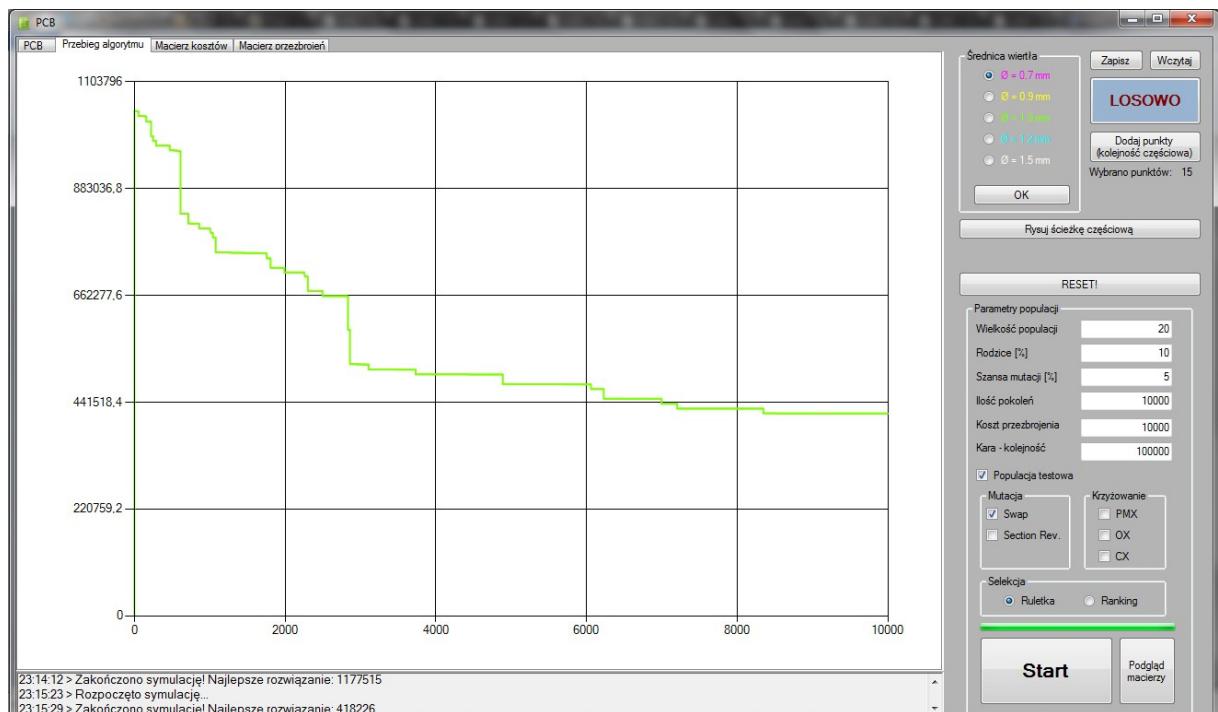
Instancja:



Rozwiązanie otrzymane metodą Swap:

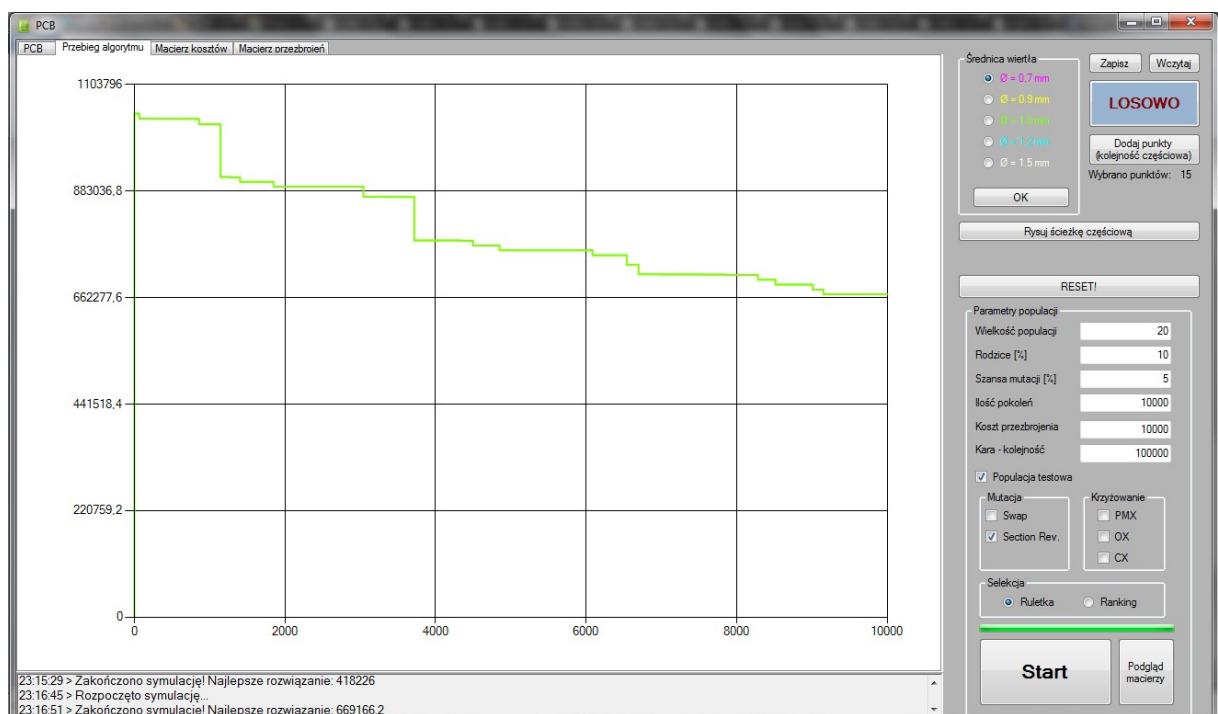


Przebieg pracy operatora Swap:



Rozwiązanie: 418226

Przebieg pracy operatora mutacji poprzez inwersję:



Rozwiązanie: 669166,2

Mimo tego, że efekty działania operatorów są podobne, to w tym problemie lepszy jest operator mutacji Swap.

Testy

W tabeli przedstawione są zebrane wyniki dla przykładowej instancji:

Testowe przebiegi dla 250 otworów, brak podcykli						
Selekcja:	Ruletka					Ranking
Krzyżowanie:	PMX, OX, CX	OX, CX	PMX, CX	PMX OX	OX	OX
1	713618,4	617322,3	1827830	563685,6	399695,3	251910,3
2	686903	615405,4	1904953	564090,2	405178,5	239332
3	672318	633776,8	1883713	523842,7	359523,9	300481,3
4	768765,8	655291,3	1860228	580988,5	402532,9	307225
5	615322,3	635792,2	1831766	485548,5	384993,2	234597,9
6	763004,9	555115,7	1903150	515229	473765,2	289189,1
7	636737,8	682502,7	1890083	623854,4	356666,7	258603,3
8	624295,6	581150,8	1816797	457452,2	357102,7	216875,5
9	716298,9	579864,3	1814714	470596,5	370577,8	296714,7
10	611724,3	613120,7	1936366	526406,1	378340,7	260354,6
minimum	611724,3	555115,7	1814714	457452,2	356666,7	216875,5
maximum	768765,8	682502,7	1936366	623854,4	473765,2	307225
średnia	680898,9	616934,2	1866960	531169,4	388837,7	265528,4

Parametry algorytmu:

Populacja: 60

Rodzice: 20%

Mutacja: 15%

Pokolenia: 5 000

Koszt przebrojenia: 10 000

Kara: -

Zebrane wyniki jednoznacznie wskazują metodę krzyżowania OX w połączeniu z rankingową selekcją, jako najlepszą. Najlepsze rozwiązanie uzyskane tą metodą jest prawie 3 razy tańsze od rozwiązania uzyskanego metodą kombinowaną wykorzystującą operatory PMX, OX oraz CX.

Najgorszą metodą okazała się kombinowana metoda, używająca PMX oraz CX. Jak już było wspomniane wcześniej metody te nie spisują się dobrze w tym przypadku.