Raport z badania Tabu Search

Ernest Przybył

1. Reprezentacja

Trasa:

Rozwiązanie problemu jest przechowywane w analogiczny sposób do rozwiązania z poprzedniego raportu z GA.

Jest to permutacja indeksów miast.

Tabu search będzie zastosowany tylko do wyboru trasy.

Plecak:

Problem plecakowy jest rozwiązywany w analogiczny sposób jak opisano w poprzednim raporcie. W sposób zachłanny dla danej trasy wybieramy przedmioty które mają największą wartość po odjęciu kosztu transportu.

2. Sąsiedztwo

Sąsiedztwo jest wyznaczane na dwa sposoby

- **Swap** – podmieniamy kolejnością losowe elementy permutacji reprezentującej rozwiązanie. W tym sposobie można sparametryzować jak bardzo sąsiedzi będą różni od osobnika stanowiącego punkt odniesienia. Jednak zależy nam na tym aby kolejne rozwiązania znajdował się blisko więc ilość podmian powinna stanowić niewielki odsetek całkowitej ilości miast.
- **Inverse** – wyznaczamy losowo dwa miejsca w permutacji względem których odwracamy kolejność elementów w permutacji.

Tabela 2. Przykłady sąsiedztwa opartego na operatorach SWAP i Inwersja

Swap – |5671234| → |5371264|
Inwersja |5671234| → |5321764|

3. Lista tabu

Lista tabu służy temu, zapobiec cyklicznemu przeszukiwaniu tych samych elementów. Rozpatrywane reprezentacje:

- Lista ostatnio odwiedzonych - Najprostszą formą listy tabu jest lista ostatnio odwiedzonych elementów. Po prostu ignorujemy sąsiadów w których już byliśmy. Lista jest reprezentowana Kolejką FIFO.

4. Eksperymenty

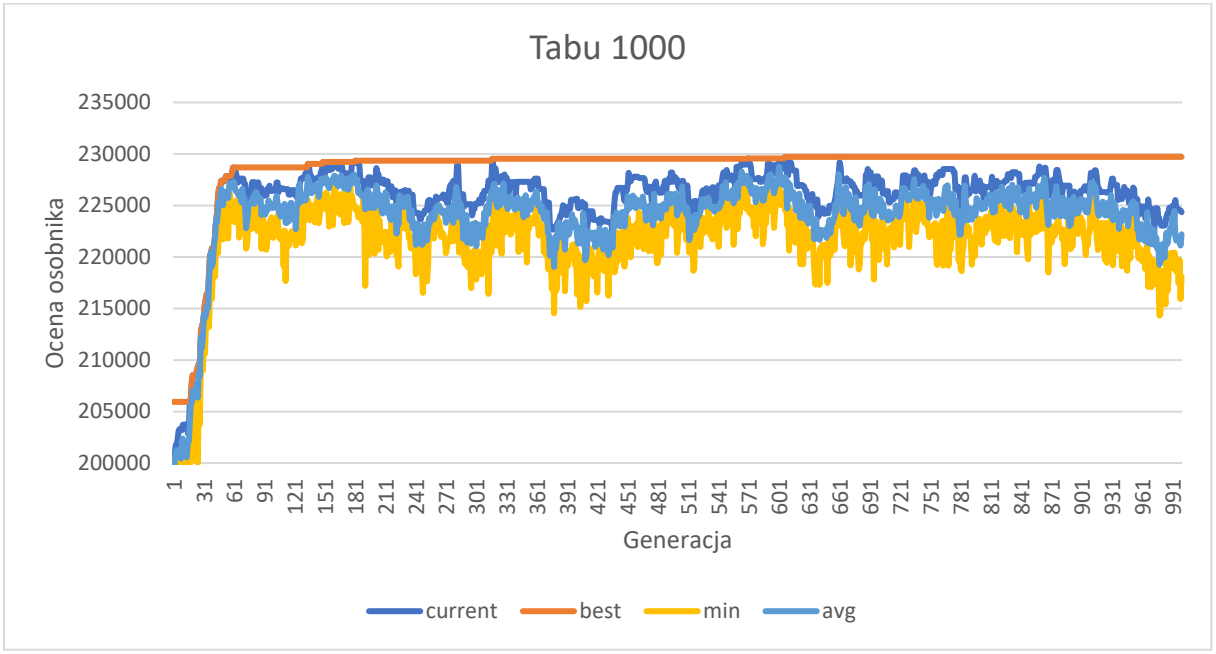
2. Wielkość tabu

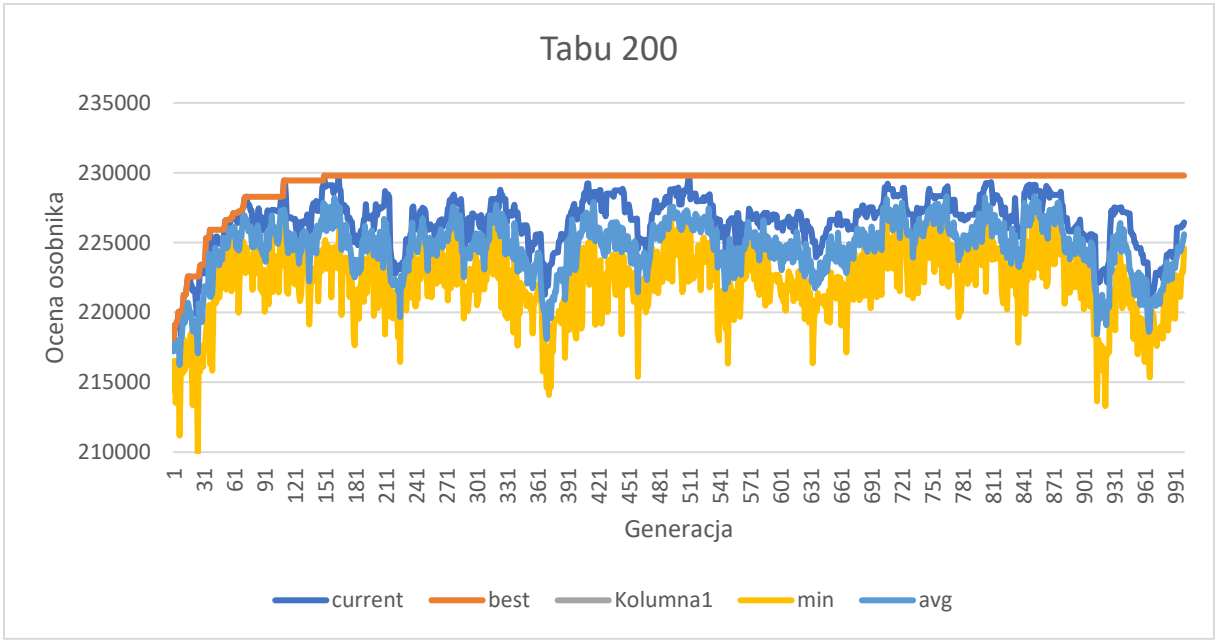
Stałe:

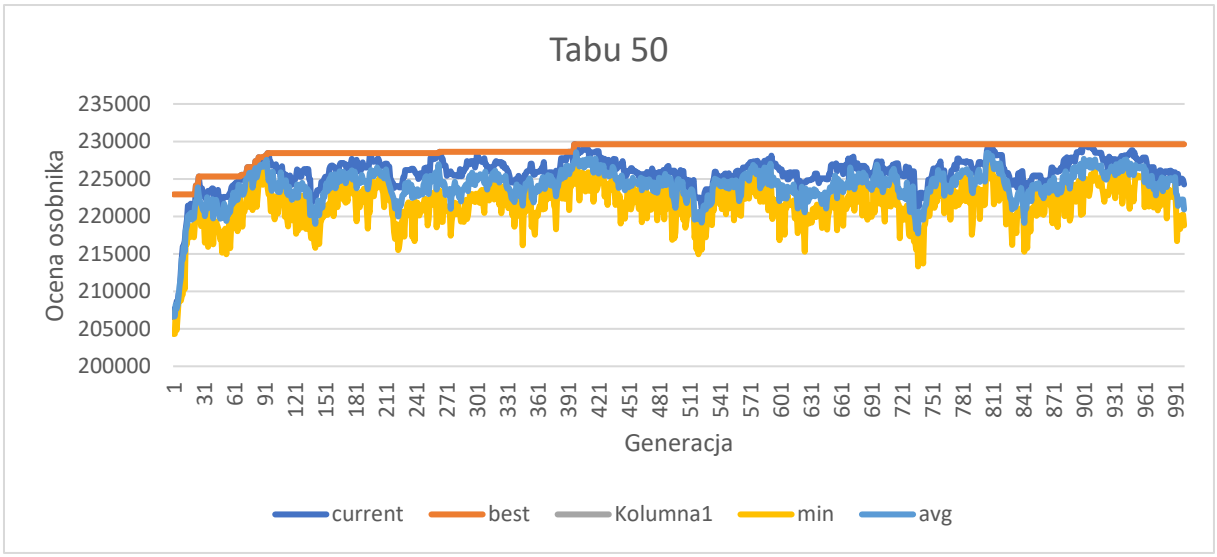
- Sąsiedztwo: Inverse; 7
- Zestaw: medium_4
- Iteracje: 1000

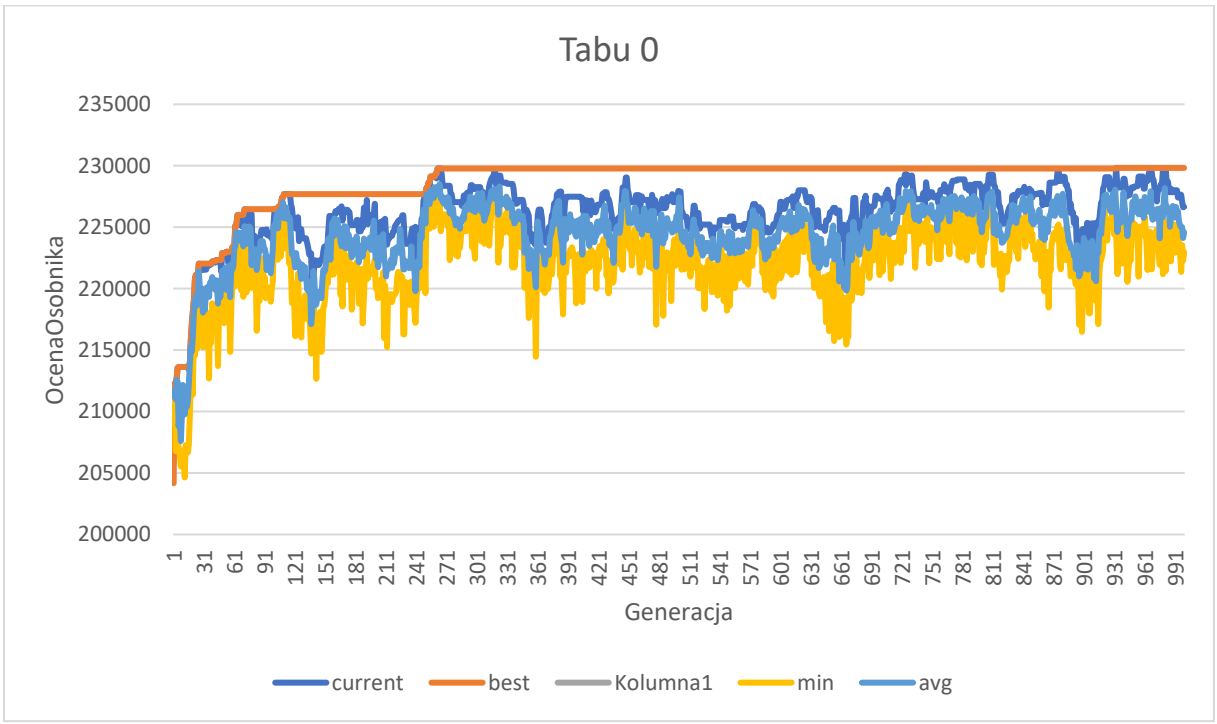
Zmienne:

- Wielkość tabu: [1000, 200, 50, 0]









Wnioski:

Spodziewałem się, że 0 wielkość tabu spowoduje, że osobni current utknie w jakimś optiumum i będzie pokrywał się przez większość przebiegu z best, nie dokonując, żadnej eksploracji. Prawdopodobnie eksploracja działa dalej ponieważ ilość sąsiadów jest stosunkowo niewielka do ilości możliwych kroków w porównaniu do swapa:

Ilość możliwych kroków dla:

- Swap: $n/2$; dla jednego swapa
- Inverse: $\frac{n^2}{2}$

$$n - liczba\ mist\ w\ problemie$$

Możemy też przypuścić, że optymalna wielkość tabu dla tej heurystyki jest zależna od wielkości problemu oraz ilości sąsiadów.

Dla większej ilości sąsiadów oraz większe ilości miast potrzeba większej ilości tabu.

Możliwe, że eksploracja działa dalej ponieważ metoda inwersji znajduje stosunkowo bardziej oddalonych sąsiadów od swapa. I w ten sposób nie utykamy w lokalnym optimum.

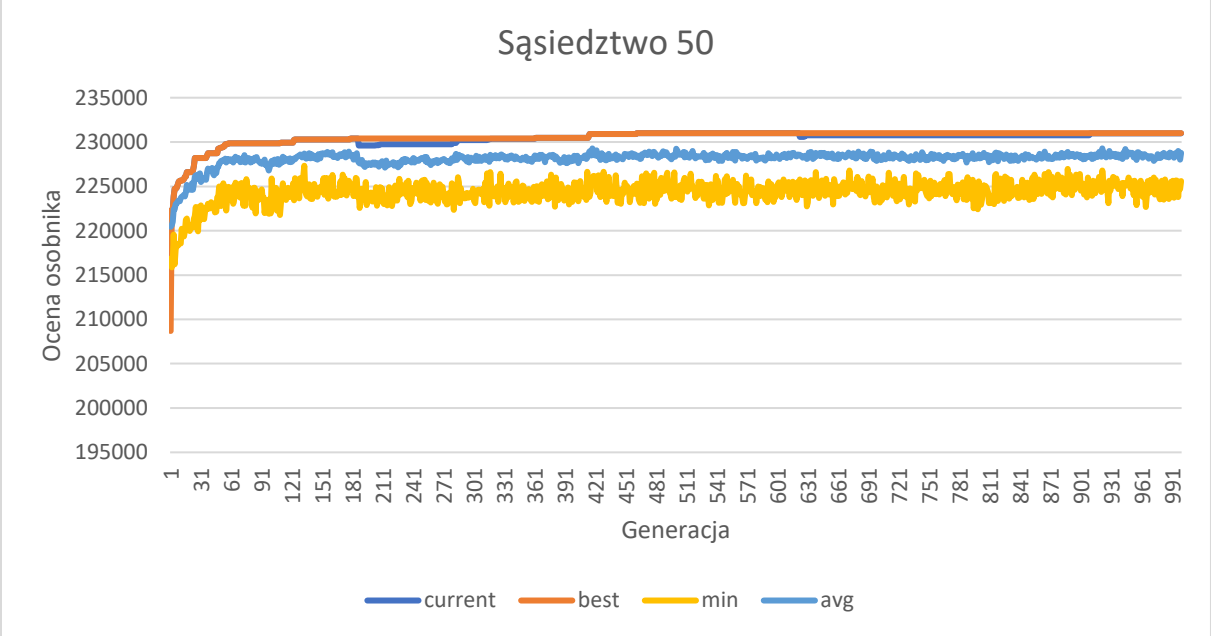
2. Ilość sąsiadów

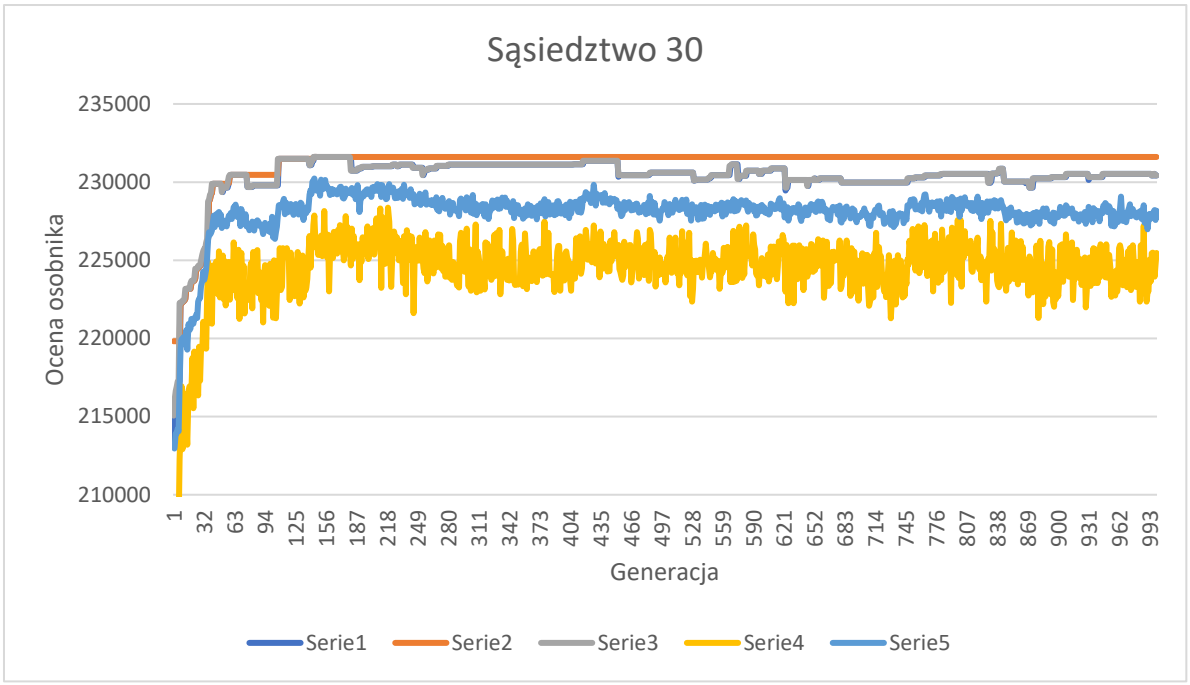
Stałe:

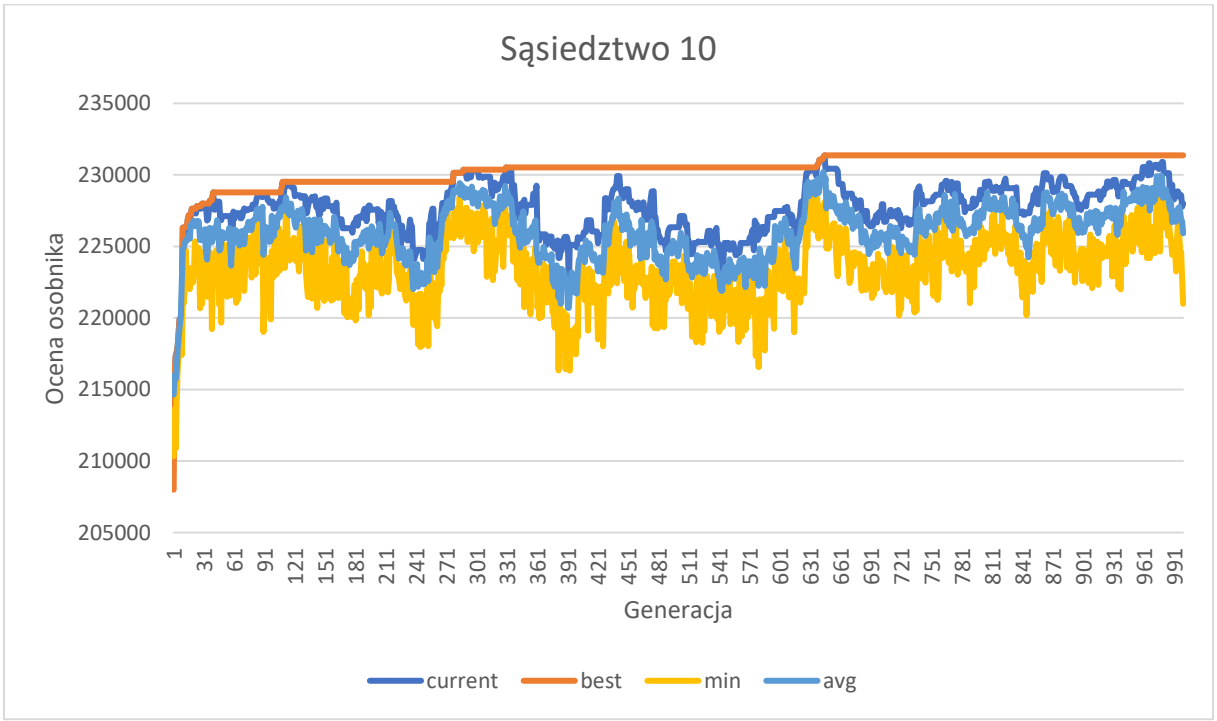
- Sąsiedztwo: Inverse;
- Zestaw: medium_4
- Iteracje: 1000
- Wielkość tabu: 200

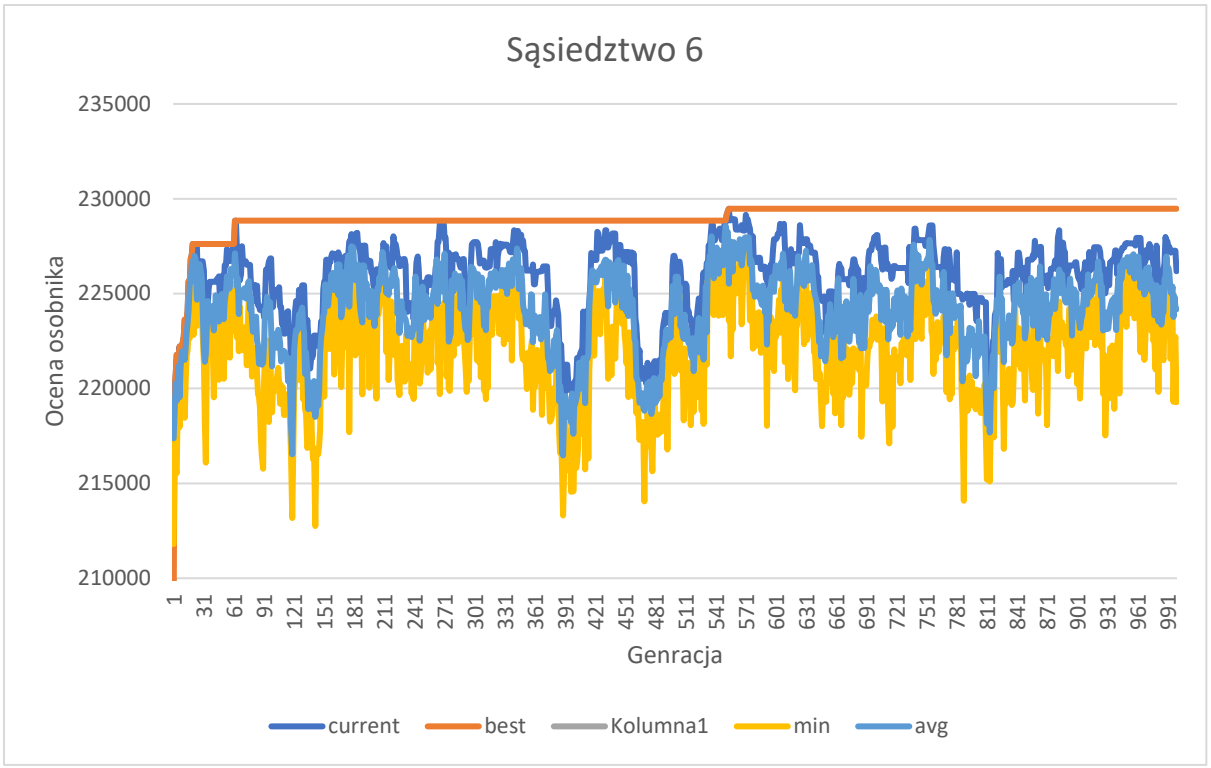
Zmienne:

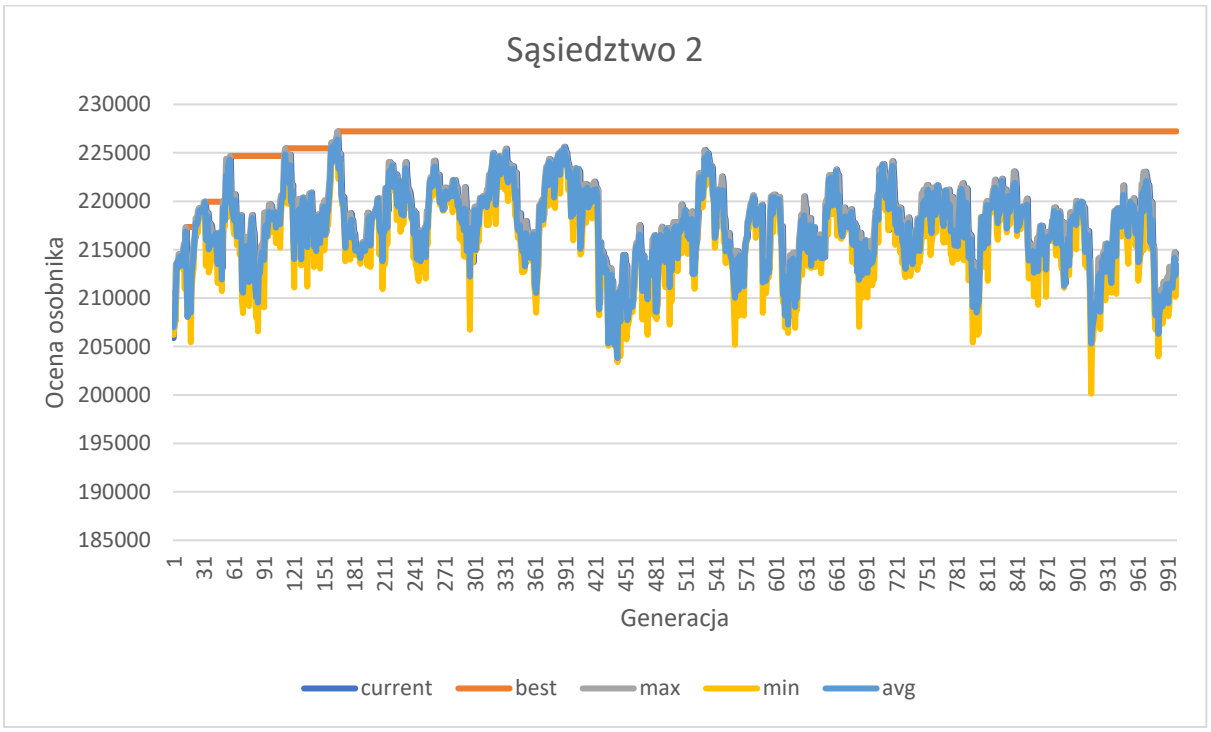
- Ilość sąsiadów: [2; 6; 10,30,50]




Sąsiedztwo 30
Ocena osobnika
235000
230000
225000
220000
215000
210000
1
32
63
94
125
156
187
218
249
280
311
342
373
404
435
466
497
528
559
590
621
652
683
714
745
776
807
838
869
900
931
962
993
Generacja
Serie1
Serie2
Serie3
Serie4
Serie5


Sąsiedztwo 10
Ocena osobnika
235000
230000
225000
220000
215000
210000
205000
1
31
61
91
121
151
181
211
241
271
301
331
361
391
421
451
481
511
541
571
601
631
661
691
721
751
781
811
841
871
901
931
961
991
Generacja
current
best
min
avg





Wnioski:

Przeglądanie zbyt wielkiej ilości sąsiadów powoduje, że algorytm gubi się w lokalnych optimach i nie szuka dynamicznie po innych wartościach.

Dla zbyt małych ilości sąsiadów. Mamy bardzo intensywną eksplorację. Jednak tręd osobnika curent nie jest wzrastający, co oznacza, że nie zmierzamy w kierunku lepszych rozwiązań.

Najlepsze wyniki otrzymaliśmy dla ilości sąsiadów z zakresu ~10. Tam przebieg osobnika current ma tręd wzrostowy, a jednocześnie w miarę dynamicznie przeszukuje pobliskie rozwiązania.

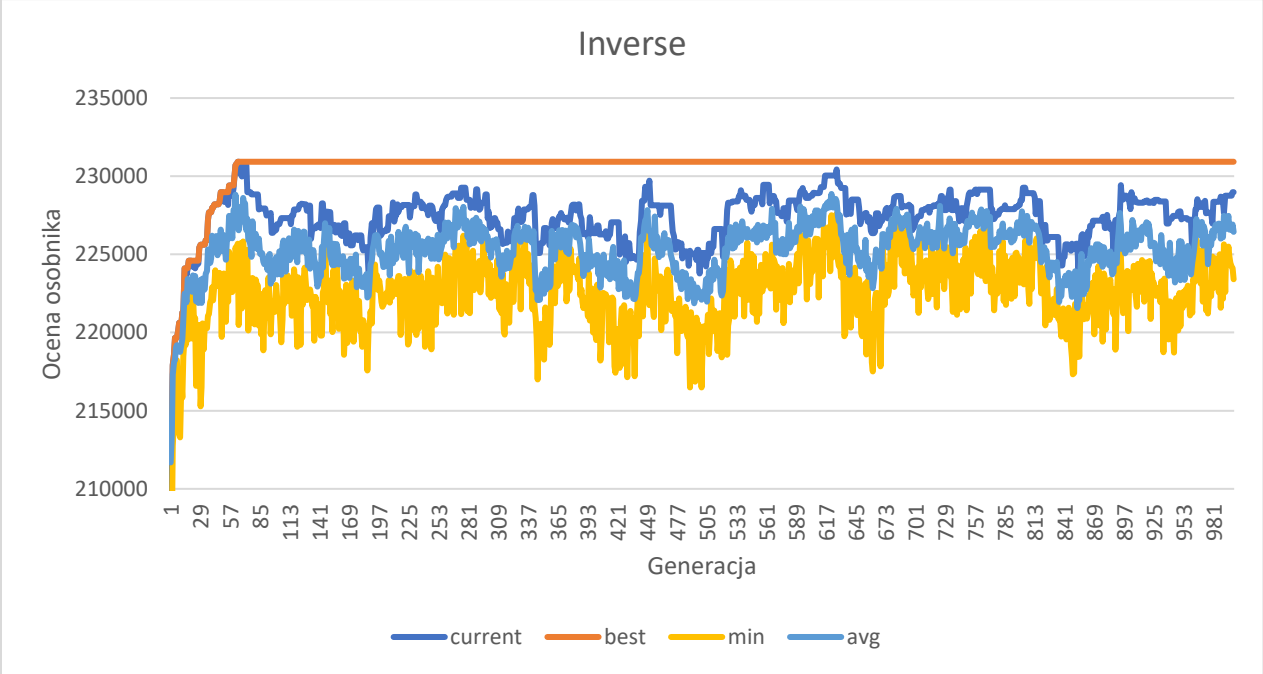
3. Porwanie sposobów wyboru sąsiada

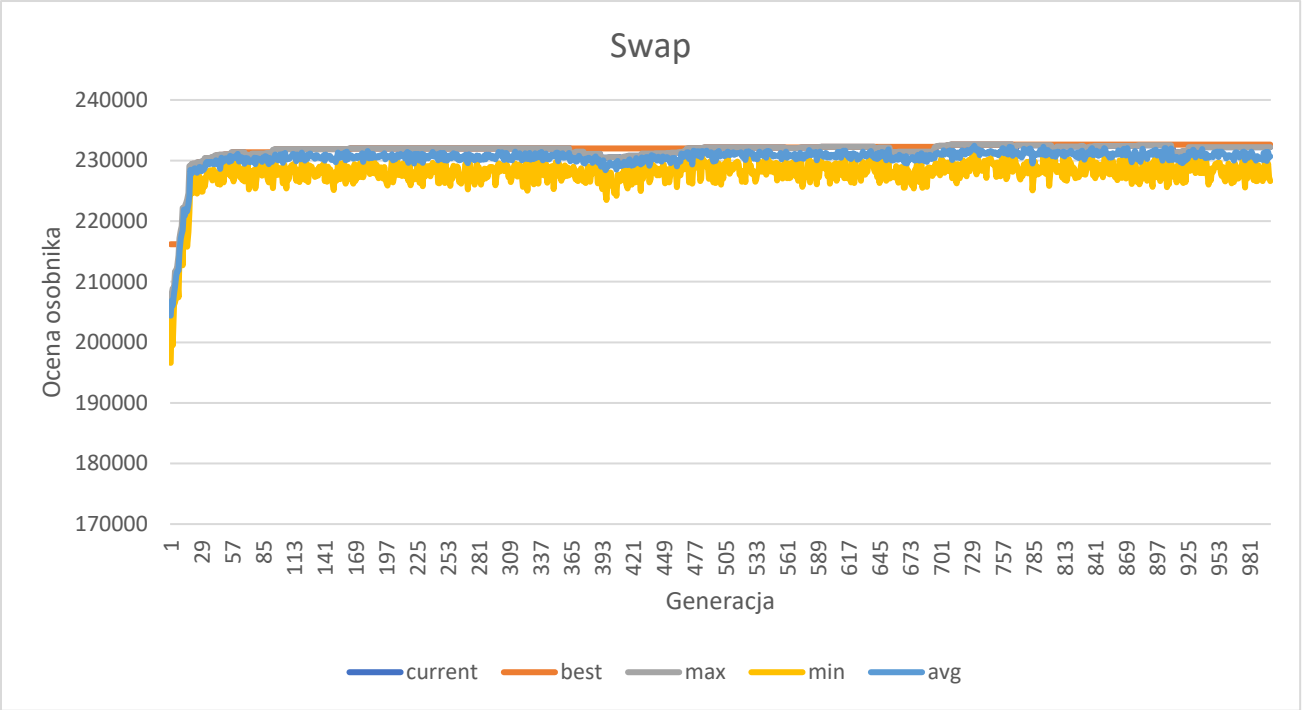
Stałe:

- Sąsiedztwo: ilość=10;
- Zestaw: medium_4
- Iteracje: 1000
- Wielkość tabu: 200

Zmienne:

- Sąsiedztwo: [„inverse”; „swap”]





Wnioski:

Ponieważ w swap każdorazowo dokonywaliśmy tylko jednej podmiany miast znajdywani sąsiedzi są naprawdę blisko siebie. Takie podejście ułatwia szybkie znajdywanie optimum, ale bardzo utrudnia eksplorację. Aby temu zapobiec można by zwiększyć tablicę

Wyniki:

Ustawienia:

- Selekcja sąsiadów: Inverse; 10 osobników
- Tabu: rozmiar 200
- Iteracje: 1000

Ustawienia(hard):

- Selekcja sąsiadów: Inverse; 30 osobników
- Tabu: rozmiar 1000
- Iteracje: 2000

Instancja	Opt. wynik	Alg. Losowy[10k]				Alg. Zachłanny[N]				Alg. Tabu Search [10x]			
		best	worst	avg	std	best	worst	avg	std	best	worst	avg	std
Easy_3		-26266	-85347	-45105	8449	-36928	-51398	-43653	3510	-20568	-21916	-21255	399
Easy_4		-23945	-74980	-39634	7123	-32480	-47833	-39137	4446	-18842	-19795	-19241	303
Medium_3		173703	117485	155768	7599	158848	144252	151807	4457	179916	179072	179565	261
Medium_4		225746	172537	210168	6352	211545	196919	204954	4189	230552	229583	230181	308
Hard_0		-502271	-1447402	-905929	143616	-193917	-2309747	-875340	718810	-195264	-254958	-220117	18045
Hard_1		-386036	-1097802	-685597	96394	-135734	-1666911	-486691	475441	-91351	-106741	-100272	5468

Wnioski i obserwacje:

Pomimo, iż wyniki w tabu search były zwykle tylko niewiele lepsze od metody losowej to charakteryzują się one niskim odchyleniem standardowym. Daje nam to dużo większą gwarancję, że otrzymamy wynik który będzie wynikiem loklanie optaymalnym.

Zaglądając od poprzedniego raportu widzimy, że AG radził sobie niewiele lepiej od tabu search. Jednak czas procesora dla AG był kilkukrotnie większy od tego dla Tabu serach.

Sytuacja odwraca się znacząco na korzyć AG w przypadku hard. Może to być związane z tym, że źle dobrałem parametry Tabu serach, albo dlatego, że TS prowadzi przeszukiwania tylko w jednym punkcie podczas gdy, AG działa na określonej populacji.

Ponieważ testy przeprowadzałem na instancjach o większej ilość przedmiotów do wyboru dla plecaka, różnice w wynikach mogą być słabo uwydatnione, ponieważ mocniej napracuje się tu algorytm zachłanny dla plecaka.