Sprawozdanie

Ćw. 10 – Algorytm optymalizacji rojem cząstek (PSO)

Filip Horst 311257

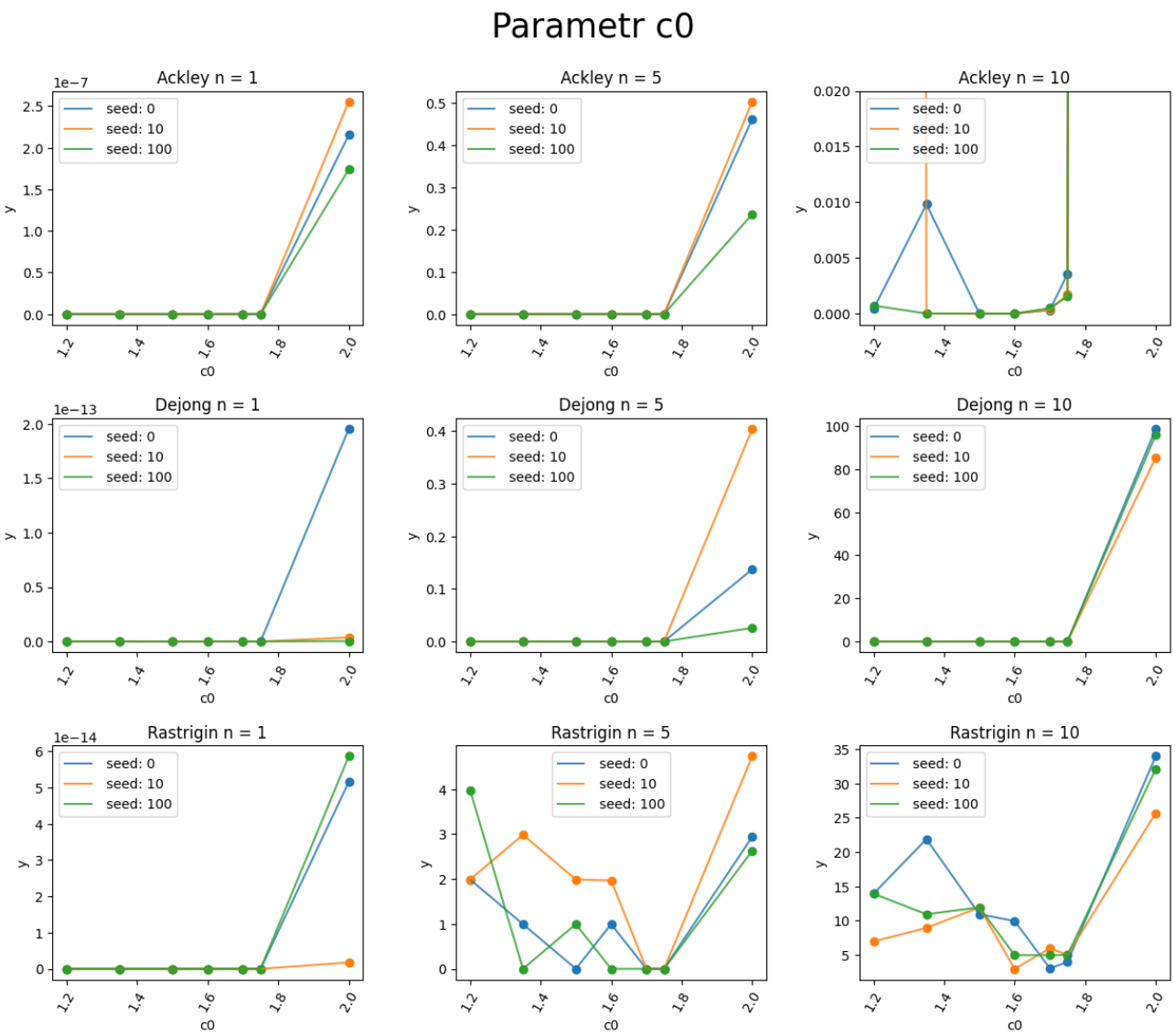
1 Badanie wpływu parametrów na wyniki

Pierwszym etapem podobnie jak w poprzednich ćwiczeniach jest analiza jednoczynnikowa. Domyślnymi ustawieniami były:

- C0 1.7
- C1 0.1
- C2 0.1
- C3 0.1
- Iter_danger 160
- Sąsiedztwo 3
- Popsize 50
- Niter 1000

Wszystkie testowane zadania posiadały optimum w punkcie 0, co pozwala na utożsamienie wyniku y z błędem (większe y = większy błąd).

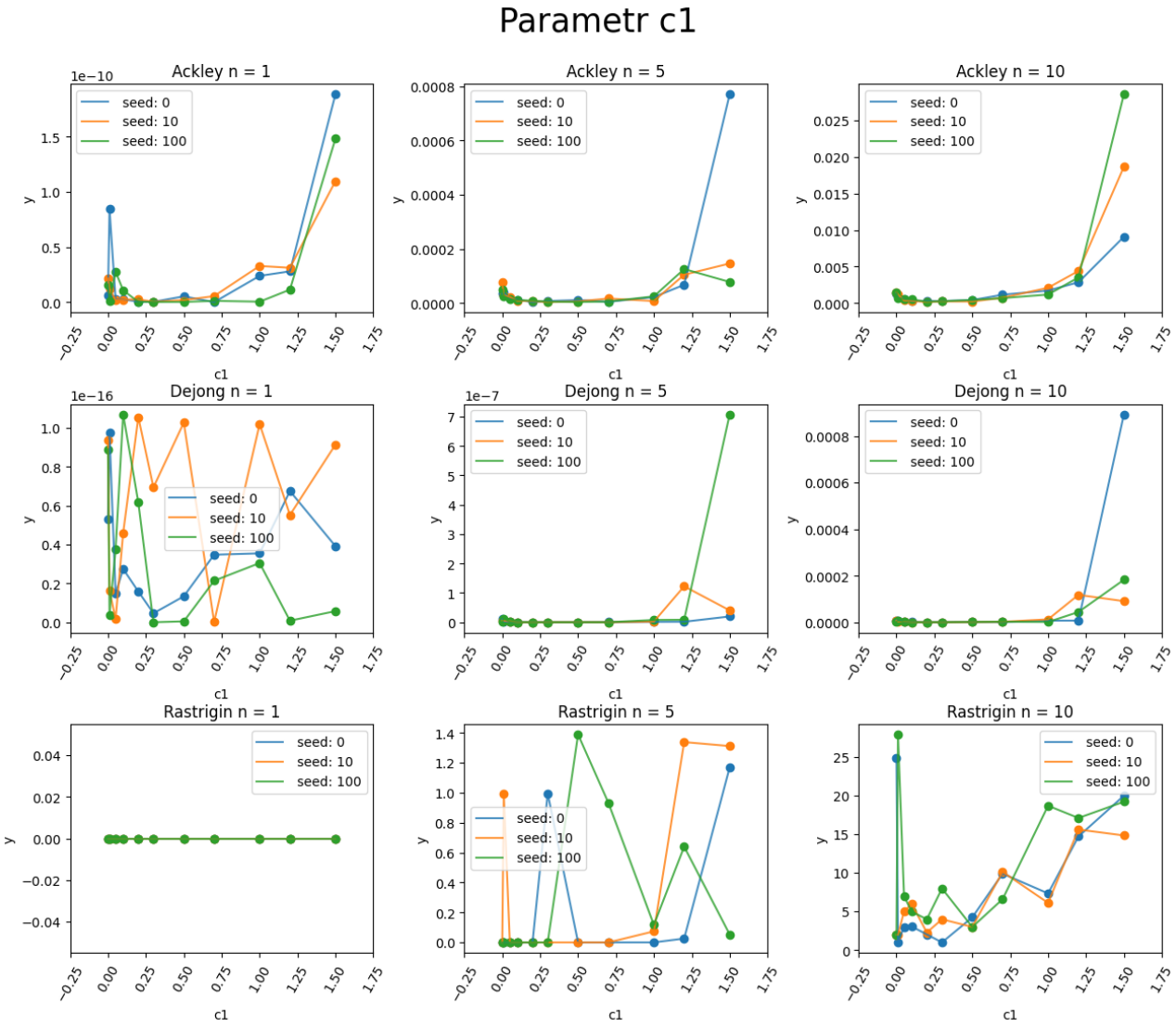
1.1 C0



Wpływ ustawień parametru na wyniki dla różnych zadań

Najważniejszym wnioskiem płynącym z eksperymentu jest to, że wartość c0 posiada pewne optimum, czyli może być zarówno zbyt wysokie, jak i zbyt niskie. Szczególnie dobrze widać to w przypadku funkcji Rastrigina o 10 wymiarach. Progi, których przekroczenie powoduje duży błąd są inne w zależności od zadania, ale dobrym przedziałem startowym wydaje się być 1.4 – 1.8.

1.2 C1



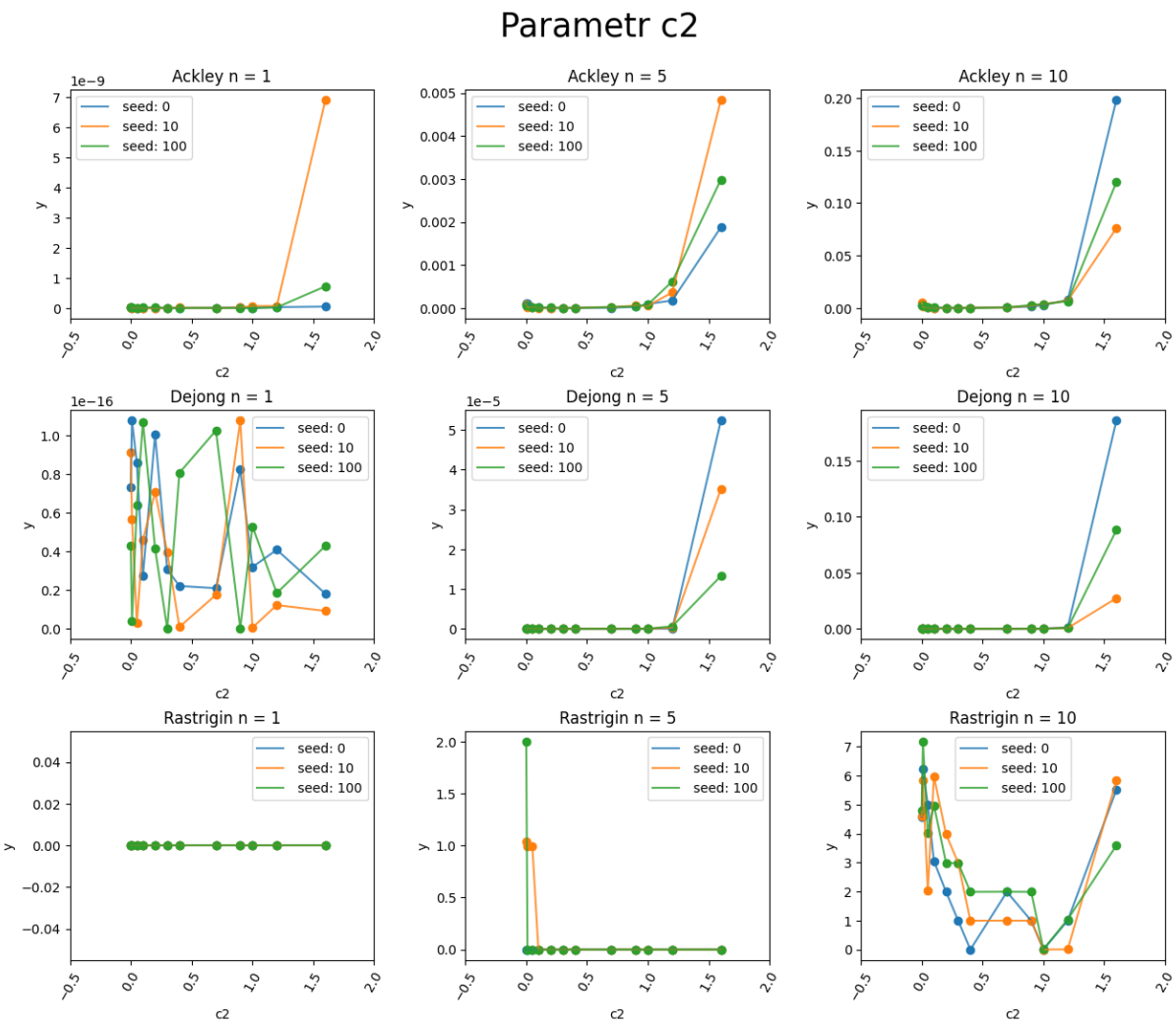
Wpływ ustawień parametru na wyniki dla różnych zadań

Dla parametru C1 zależność jest podobna do C0, co widać jednak tylko na 10 wymiarowej funkcji Rastrigina. Różnicą jest jednak przede wszystkim wartość – tym razem przedział startowy lepiej ustalić na 0.1 – 0.4.

Warto zaznaczyć, że losowe skoki dla funkcji jednowymiarowych są skutkiem bardzo małych wartości wyników (rząd e-10, e-16), gdzie zwykłe elementy losowości powodują skoki wartości – ostatecznie każdy z nich jest praktycznie idealny.

Wyniki dla f. Ratrigina są bardzo nietypowe, ponieważ zauważalne są losowe oscylacje również dla wersji 5- wymiarowej, natomiast dla funkcji 10 – wymiarowej najbardziej zauważalny jest charakter paraboli (oczywiście mówiąc bardzo ogólnie) zależności między c1, a wynikiem y.

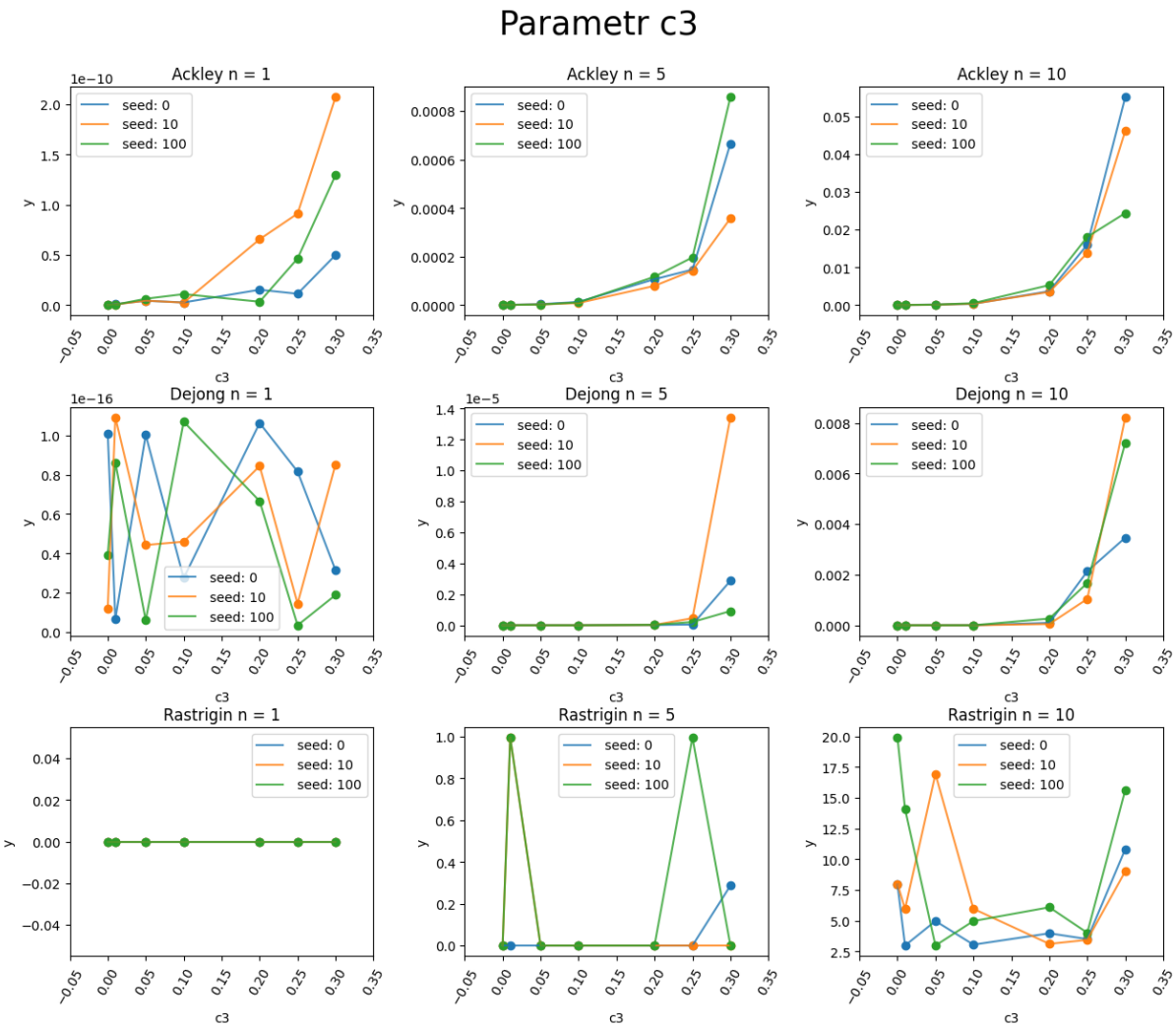
1.3 C2



Wpływ ustawień parametru na wyniki dla różnych zadań

Analogicznie do swoich poprzedników wyniki pokazują, że istnieje pewien przedział najlepszych ustawień dla problemu. Biorąc pod uwagę wyniki z różnych zadań dobrym przedziałem startowym wydaje się być 0.5-1.

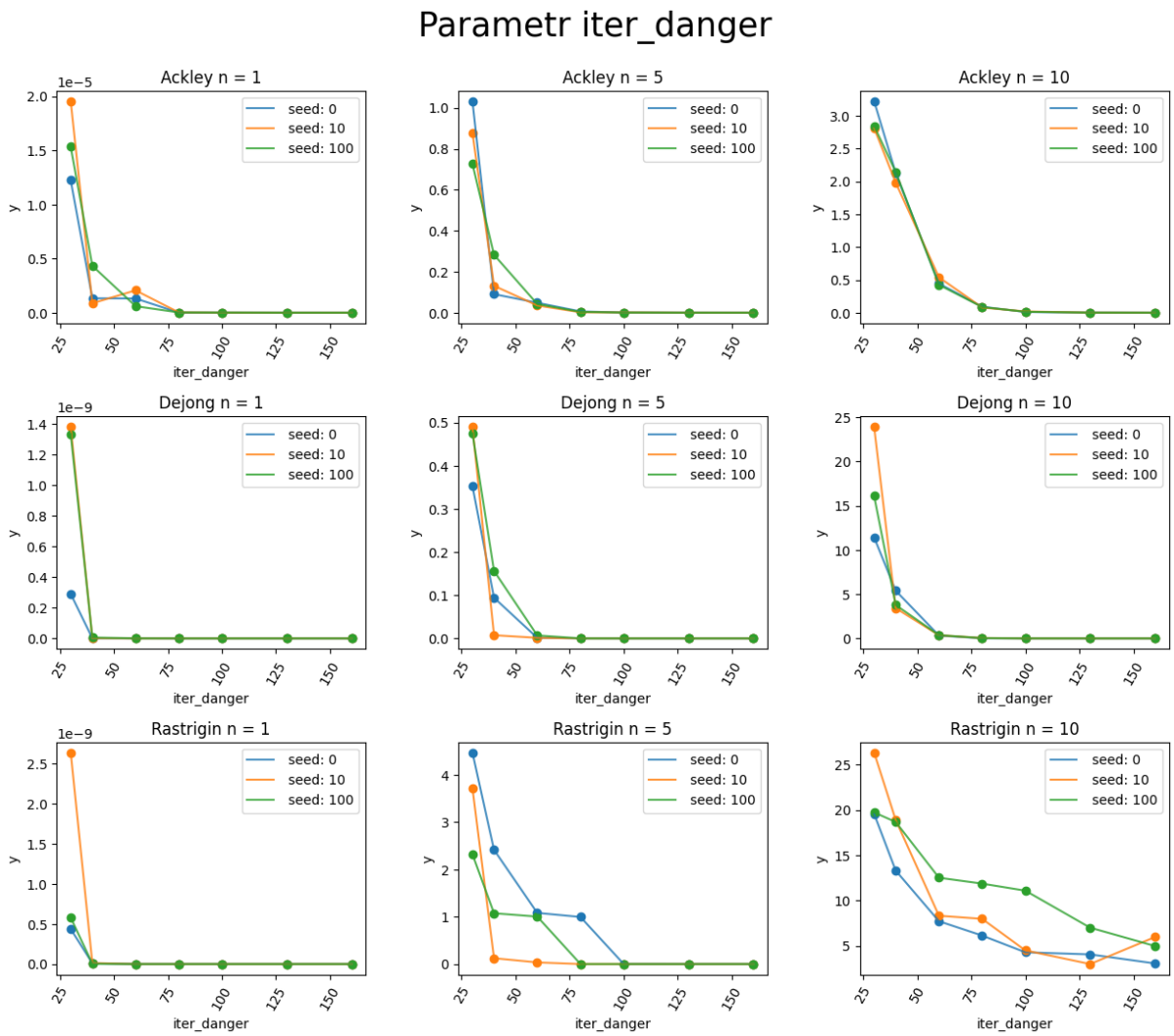
1.4 C3



Wpływ ustawień parametru na wyniki dla różnych zadań

Wrażenia analogiczne do poprzednich parametrów „c”. Dobry przedział wartości to 0.05 – 0.2.

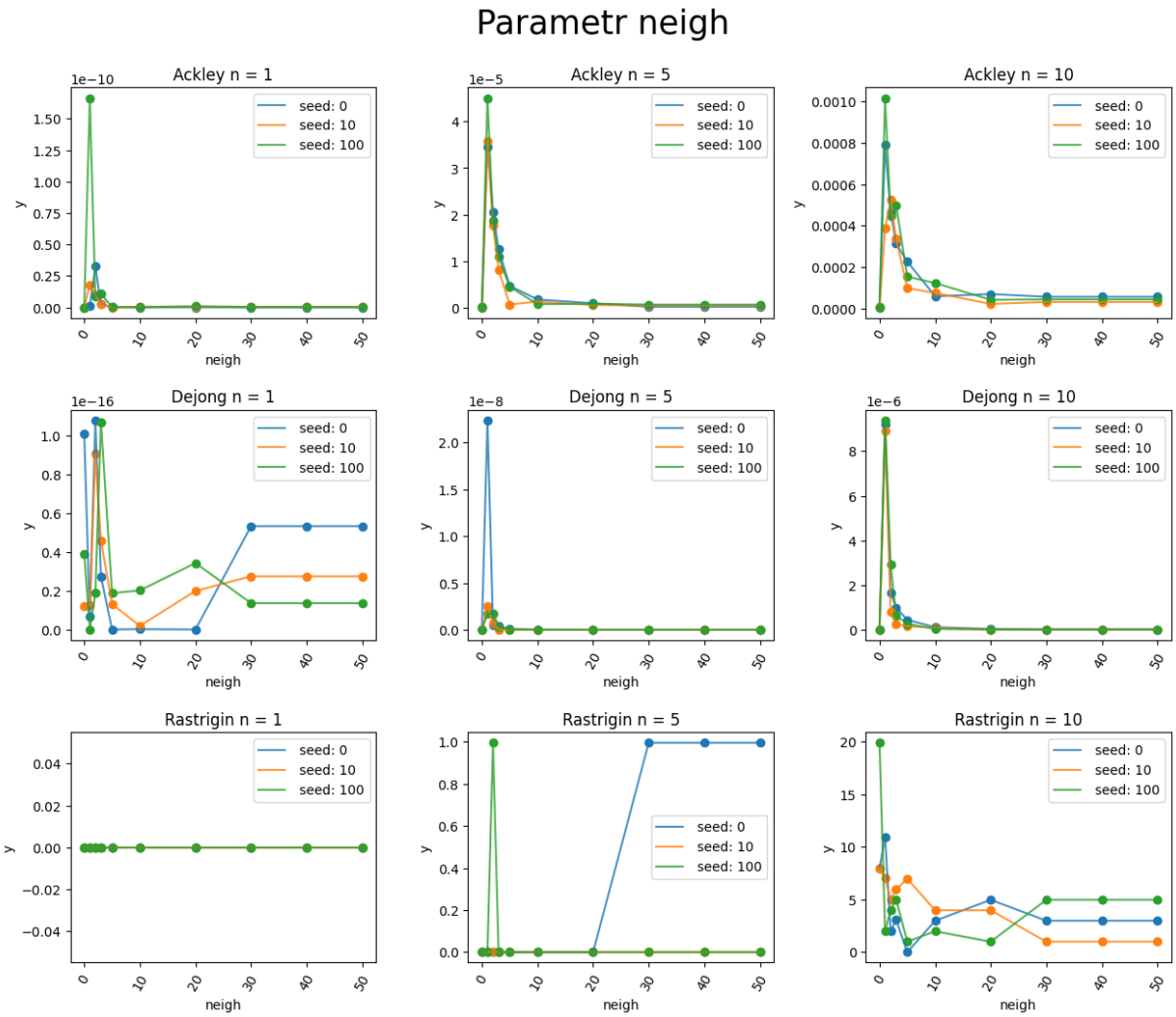
1.5 Iter_danger



Wpływ ustawień parametru na wyniki dla różnych zadań

Parametr iter_danger zdaje się nie mieć wartości „zbyt dużej”. W pewnym momencie ustawienie powoduje coś w stylu nasycenia i dalsze zwiększanie nie ma sensu. Na tym etapie nie ma zauważalnych wad dużych wartości parametru, więc można podejrzewać że po prostu resetowanie nie jest opłacalne w tym problemie, ponieważ potencjalna wada jaką jest zwiększona podatność na minima lokalne nie jest wystarczająco groźna. Mówiąc inaczej, brak zjawiska pogorszenia się wyników może być spowodowany zbyt łatwym zadaniem dla algorytmu (inne parametry radzą sobie z problemem minimów).

1.6 Sasiedztwo



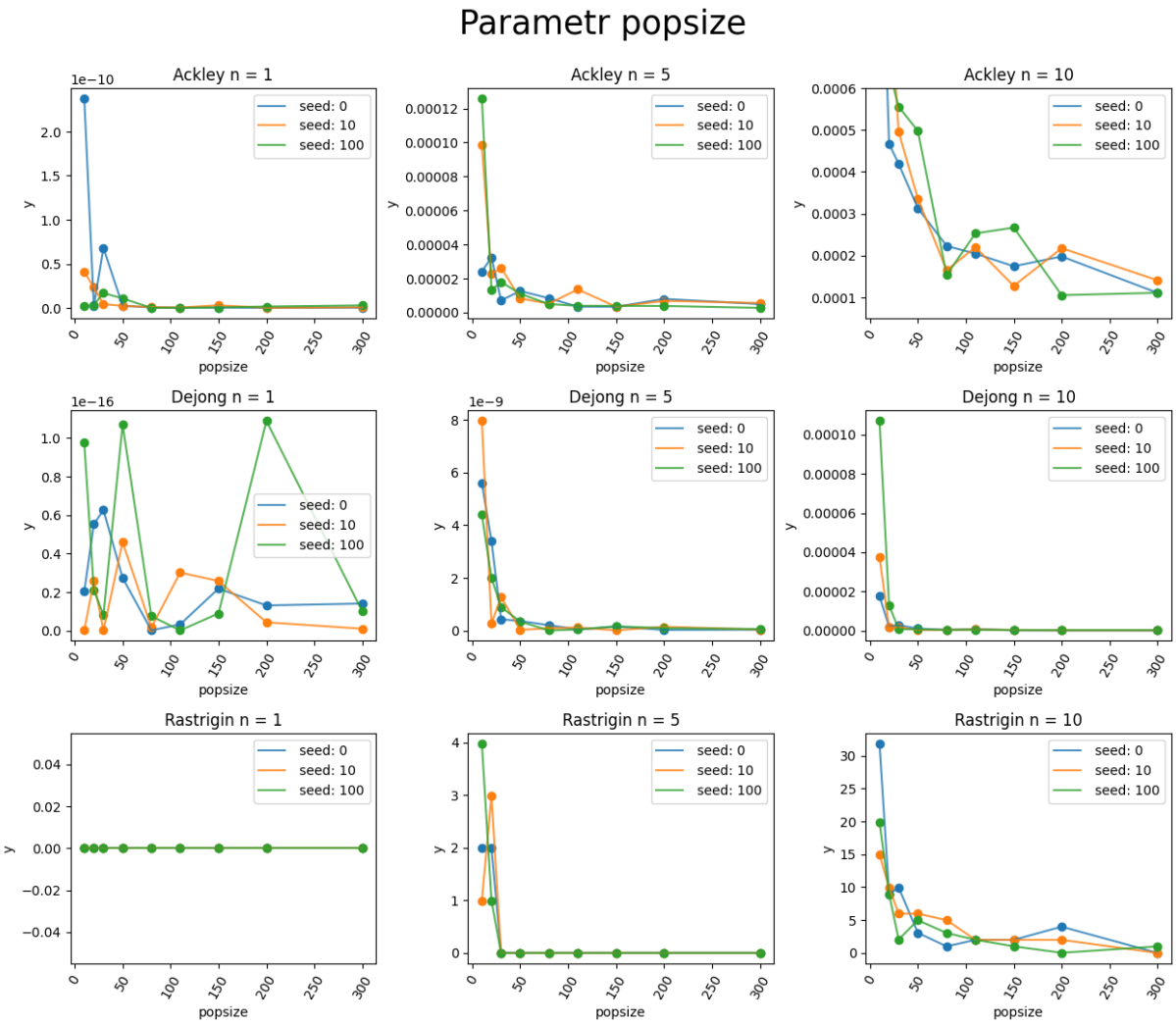
Wpływ ustawień parametru na wyniki dla różnych zadań

Sąsiedztwo podobnie do poprzedniego iter_danger wydaje się w pewnym momencie nasycać. Zauważalne jest jednak, że po przekroczeniu tego progu wyniki całkowicie się stabilizują, co wskazuje na to, że w pewnym momencie grupa sąsiadów staje się na tyle duża, że szansa na zawarcie się w niej najlepszego osobnika całej populacji staje się bardzo duża, a jeśli tak się stanie to mechanizm sąsiedztwa przekształca się we wzmocnienie czynnika społecznego.

Sam mechanizm sąsiedztwa jest jednak czasem opłacalny. Dla łatwiejszych zadań można zauważyć, że w ustawieniu 0 wartość jest bardzo niska, a często minimalna, jednak dla skomplikowanej funkcji 10 wymiarowej Rastrigina, algorytm lepiej sobie radzi z małym sąsiedztwem.

Ogólnie można powiedzieć, że sąsiedztwo ma sens tylko dla trudniejszych problemów, a nawet wtedy jest stosunkowo niewielkie – grupy kilku osobników.

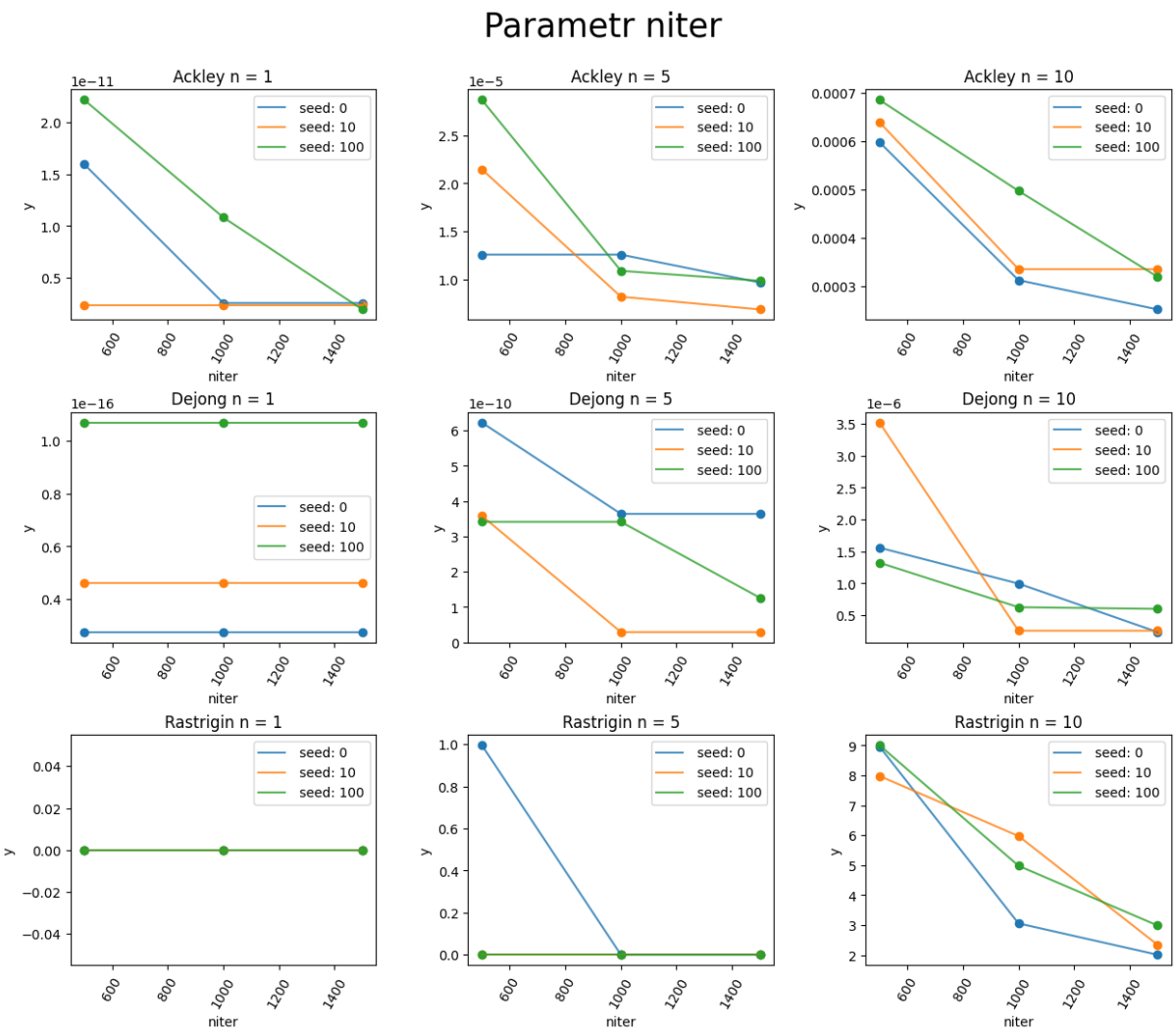
1.7 pop_size



Wpływ ustawień parametru na wyniki dla różnych zadań

Liczebność populacji na ogół wpływa pozytywnie na działanie algorytmu. Nie ma zauważalnego punktu zbyt dużej populacji tak jak w przypadku algorytmu ewolucyjnego nawet dla najprostszych funkcji. Jedyną zaobserwowaną wadą większych populacji jest czas trwania algorytmu.

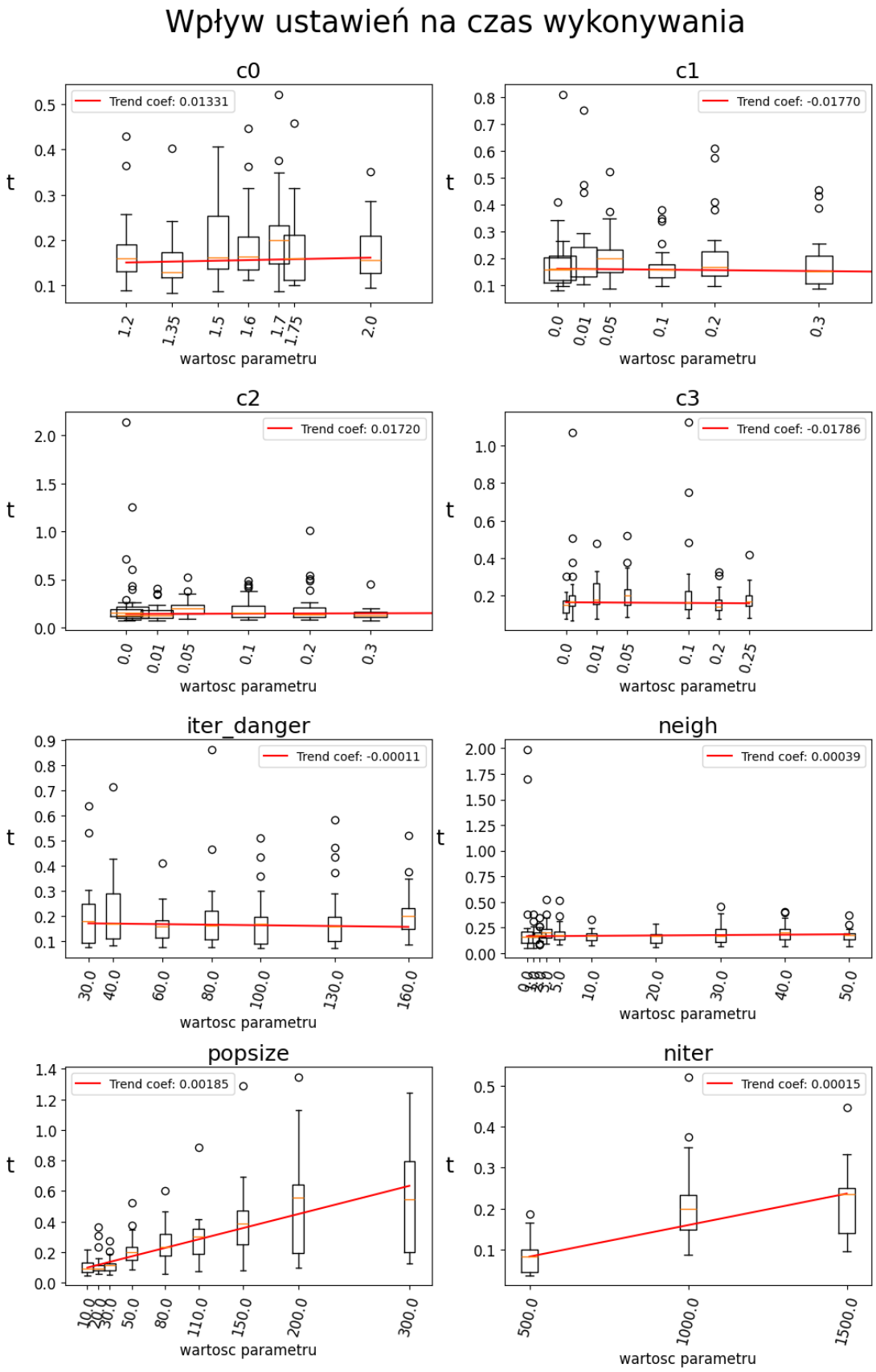
1.8 niter



Wpływ ustawień parametru na wyniki dla różnych zadań

Liczba iteracji jest oczywista – im więcej tym lepiej, choć czasami poprawy są na tyle małe (lub dosłownie zerowe), że nie ma sensu kontynuowanie obliczeń. Wadą jest też wydłużenie czasu pracy.

2 Wpływ parametrów na szybkość wykonania



Wpływ różnych ustawień parametrów na czas wykonania algorytmu

Realnie, tylko parametry popsize oraz niter mają wpływ na czas wykonania.

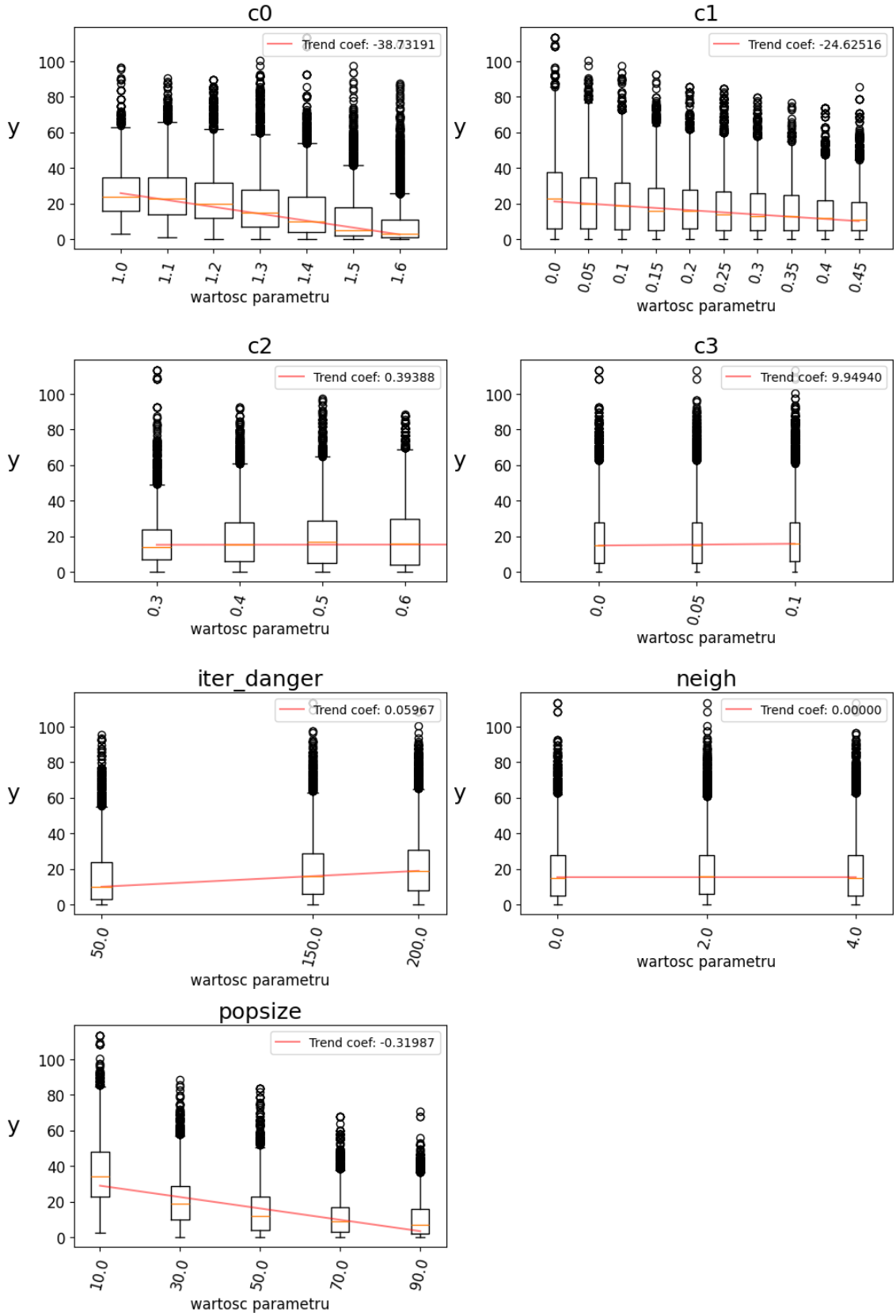
W pozostałych przypadkach, mimo teoretycznie większego nachylenia prostej trendu, realne zmiany w czasie nawet między skrajnymi ustawieniami nie mają zupełnie żadnego znaczenia z perspektywy użytkownika.

Ciekawym przypadkiem jest iter_danger. Zwiększanie liczby iteracji przed restartem jest bardzo lekko proporcjonalne do spadku czasu trwania.

3 Analiza wieloczynnikowa

Analiza wieloczynnikowa, czyli obszerniejszy „grid search” została wykonana dla 10-wymiarowej funkcji Rastrigina i objęła ponad 47 000 kombinacji. Badane były przede wszystkim parametry C - ustawione zakresy odpowiadały tym, które wydawały się najlepsze w analizie pojedynczych parametrów, aby odpowiedzieć na pytanie, czy optymalizacja parametrów po kolei daje odmiennie wyniki od optymalizacji jednoczesnej.

Wpływ ustawień na wynik przy przeszukaniu typu grid search



Wyniki wielu ustawień przy danej wartości wybranego parametru przedstawione w postaci wykresów pudełkowych.

Parametr c0 zachowywał się podobnie jak w przypadku analizy jednoskładnikowej malejąc do wartości bliskiej 0 (zarówno minimum, jak i mediana) dla ustawienia 1.6.

Parametr c1 wykazuje inną zależność niż wcześniej. W poprzednim badaniu wynik był optymalny w ustawieniu 0.3, a w kolejnym zbadanym 0.5 był znacznie gorszy (ok. 4-krotnie). W tej analizie dla ustawienia 0.45 błąd wciąż maleje, więc można podejrzewać że dla ustawienia 0.5 nie skoczył by nagle do ogromnego błędu. Oznacza to, że wartość parametru c1 zależy bardziej od innych ustawień. Trzeba zaznaczyć, że poprawa między 0.3, a 0.45 to zaledwie kilka procent.

Parametr c2 w tym badaniu jest praktycznie stały. W badaniu pojedynczym w miejscu przeszukanego zakresu występowały pewne skoki wartości wyniku.

Parametr c3 wygląda jakby nie miał wpływu na wynik jednak współczynnik dopasowanego trendu wynosi aż prawie 10. Oznacza to, że zwiększanie wartości parametru lekko pogarsza wyniki. Nie jest to duże pogorszenie i duży współczynnik prawdopodobnie wynika z faktu, że badane było ustawienie mające bardzo małą wartość w stosunku do szerokości zakresu.

Wartą podkreślenia obserwacją jest to, że w tym przeszukaniu wyniki pogorszyły się dla dużej wartości iter_danger. To znaczy, że przy innym ustawieniu pozostałych parametrów istnieje szansa wystąpienia problemów z minimami lokalnymi, przy których zbyt mała liczba restartów skutkuje gorszym wynikiem końcowym.

Parametry neigh i popsize wykazały podobne zależności co analiza pojedynczo.

Dodatkowo warto zauważyć, że niezależnie od ustawień każdego z parametrów istniała kombinacja pozostałych, która dawała praktycznie idealne wyniki. Po zgrupowaniu wyników po wartości każdego z parametrów i wybraniu minimalnego wyniku dla każdego ustawienia wybrano najgorszy przypadek (max) oraz medianę i zawarto je w tabeli jako dowód tego zjawiska:

Parametr	Maksymalny wynik spośród minimalnych dla różnych ustawień	Mediana wyników minimalnych dla różnych ustawień
C0	2.984877	8.001848e-09
C1	1.082299e-10	7.105425e-15
C2	0	0
C3	0	0
Iter_danger	8.001848e-09	0
Neigh	0	0
Popsize	2.685423	7.958079e-13

Aby zwiększyć zrozumiałość wartości w tabelce poniżej dołączony jest kod je generujący. df to tabela zawierająca wyniki symulacji.

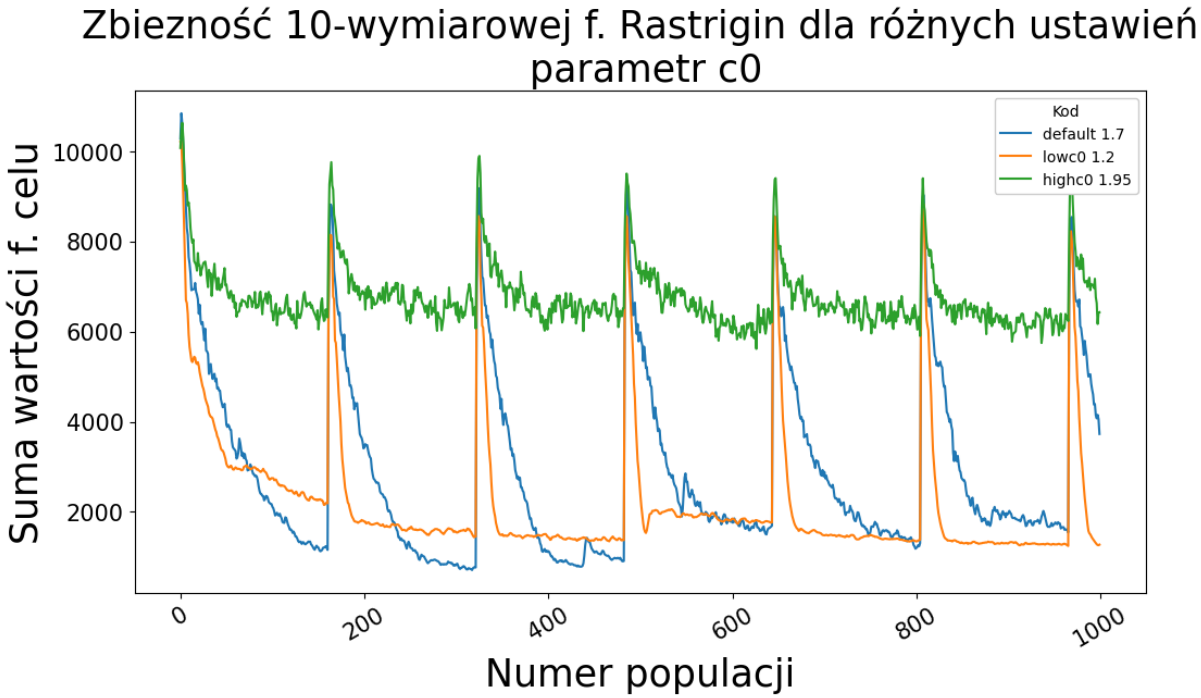
```
for param in ['c0','c1','c2','c3','iter_danger','neigh','popsize','niter']
    gb = df.groupby(by=param)
    gbdf = gb.min(numeric_only=True)[y]
    print(param, np.max(gbdf), np.median(gbdf))
```

Najważniejszym wnioskiem, którym można podsumować tę tabelę jest to, że algorytm PSO jest w tym zadaniu tak skuteczny, że istnieje bardzo dużo różnych kombinacji, dla których da się uzyskać wyniki bardzo bliskie optimum – wystarczy, że ustawienia będą w ogólnie dobrym przedziale, który bezproblemowo można znaleźć badając parametry pojedynczo.

Podsumowując analiza wieloskładnikowa pozwoliła uzyskać inne zależności dla niektórych parametrów, jednak nawet jeśli analiza pojedynczo była gorsza to nieznacznie. Biorąc pod uwagę, że przeszukanie automatyczne trwało ponad 2 godziny (a przy okazji spowodowało bluescreen) to potencjalny zysk jest nieopłacalny, tym bardziej biorąc pod uwagę, że wyniki i tak są dostatecznie bliskie zeru.

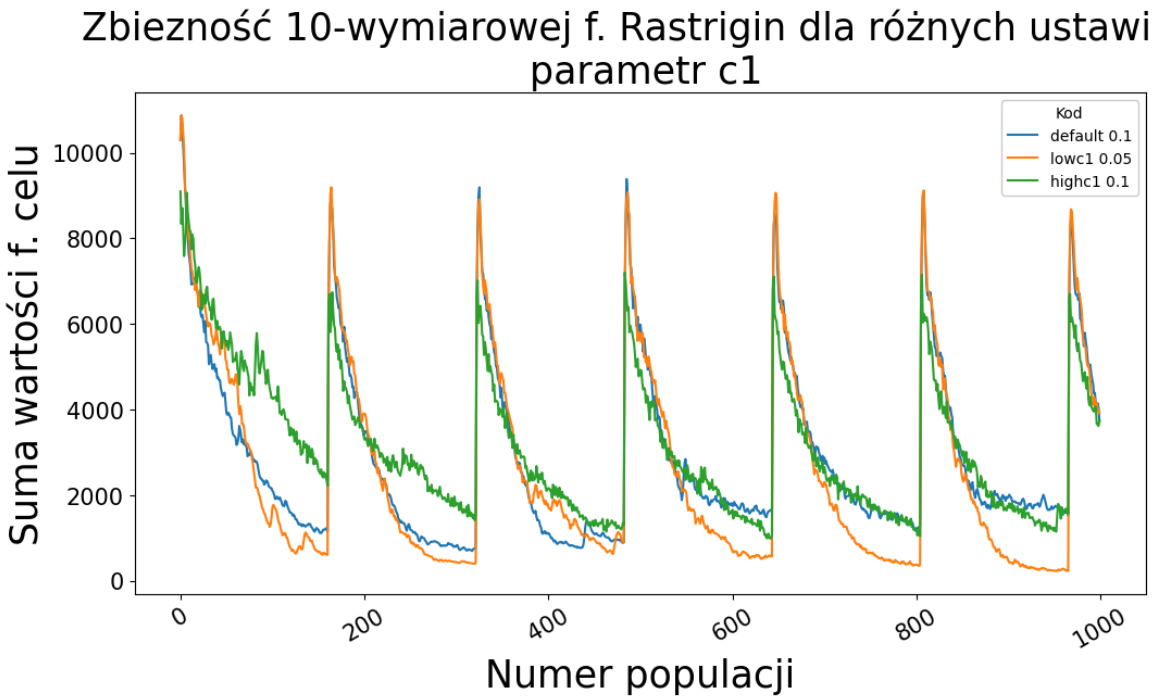
Przeszukania z dokładniejszymi i szerszymi zakresami nie zostały przeprowadzone z powodu czasochłonności oraz niewielkich różnic w wynikach końcowych porównaniu do badania jednoczynnikowego.

4 Analiza zbieżności dla różnych ustawień



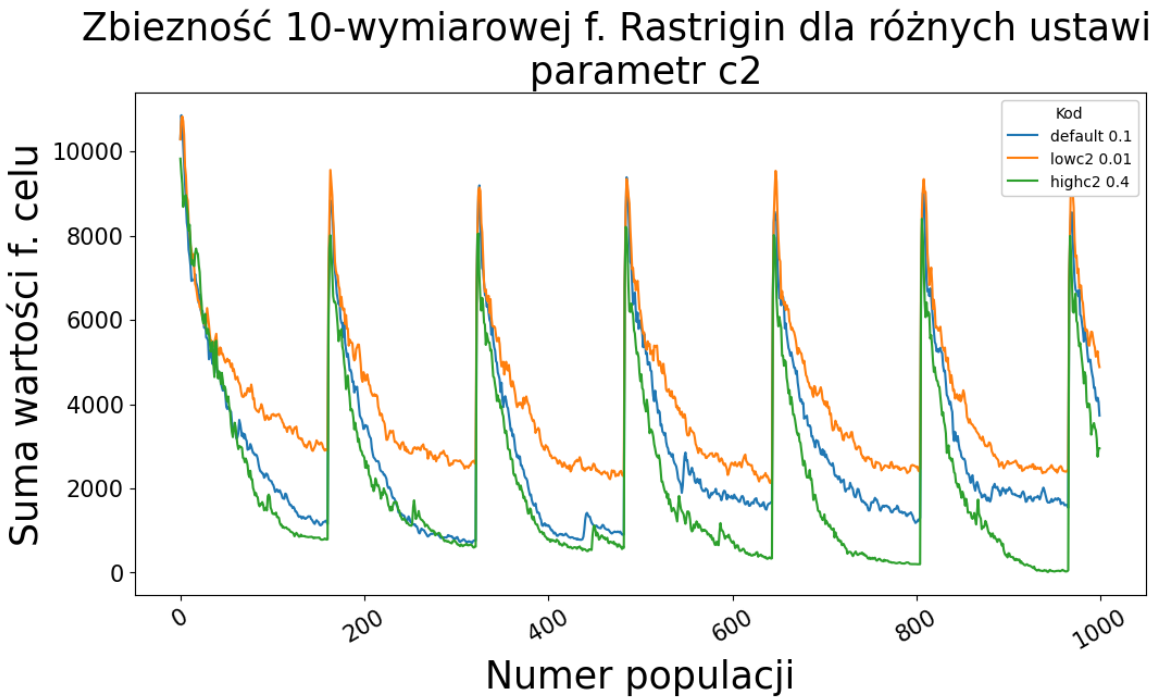
Zbieżność populacji w zależności od ustawień jednego z parametrów

Parametr c0, czyli bezwładność. Dla zbyt dużej wartości bardzo trudno zmienić kierunek cząstki – skutkuje to brakiem zbieżności i dużą losowością. Zbyt niska wartość wręcz przeciwnie – sprawia, że cząstki często zmieniają kierunek. Jak się okazuje prowadzi to do bardzo stromego zbiegu na początku, które później zatrzymuje się w miejscu. Oznacza to, że brak bezwładności powoduje zbyt słabą eksplorację i cząstki zbyt mocno upodabniają się do najlepszych. Optymalny jest kompromis.



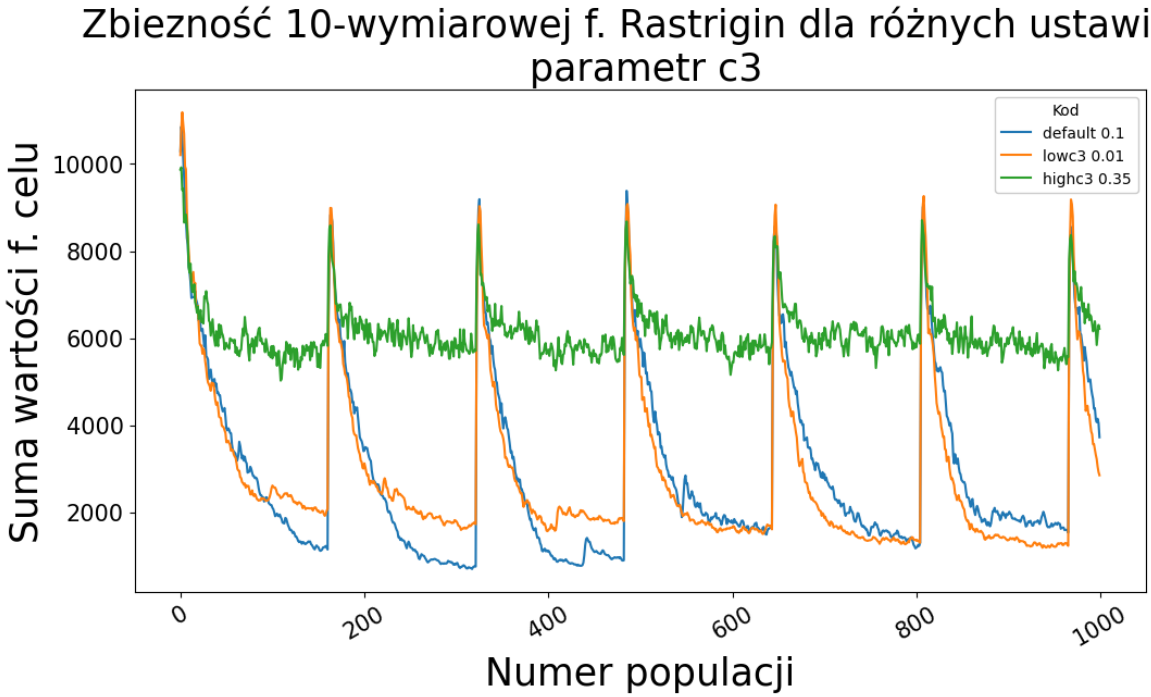
Zbieżność populacji w zależności od ustawień jednego z parametrów

Parametr c1, czyli współczynnik upodabniania się do własnego rekordu. Wysoka wartość działa analogicznie do zbyt dużej bezwładności, choć nie tak drastycznie – słaba eksploracja prowadzi do słabej zbieżności. Niska wartość c1 okazała się w tym zadaniu lepsza od domyślnej. Można zaobserwować, że dla niskiego czynnika poznawczego cząstka szybciej zbiega do najlepszych osobników (chociaż nie zawsze najszybciej!). Jak pokazały poprzednio opisane badania algorytm nie miał problemów z minimami lokalnymi w badanej funkcji, a to jest potencjalnie największe ryzyko niskiego czynnika poznawczego – wpadanie w minima lokalne trafione przez inne osobniki.



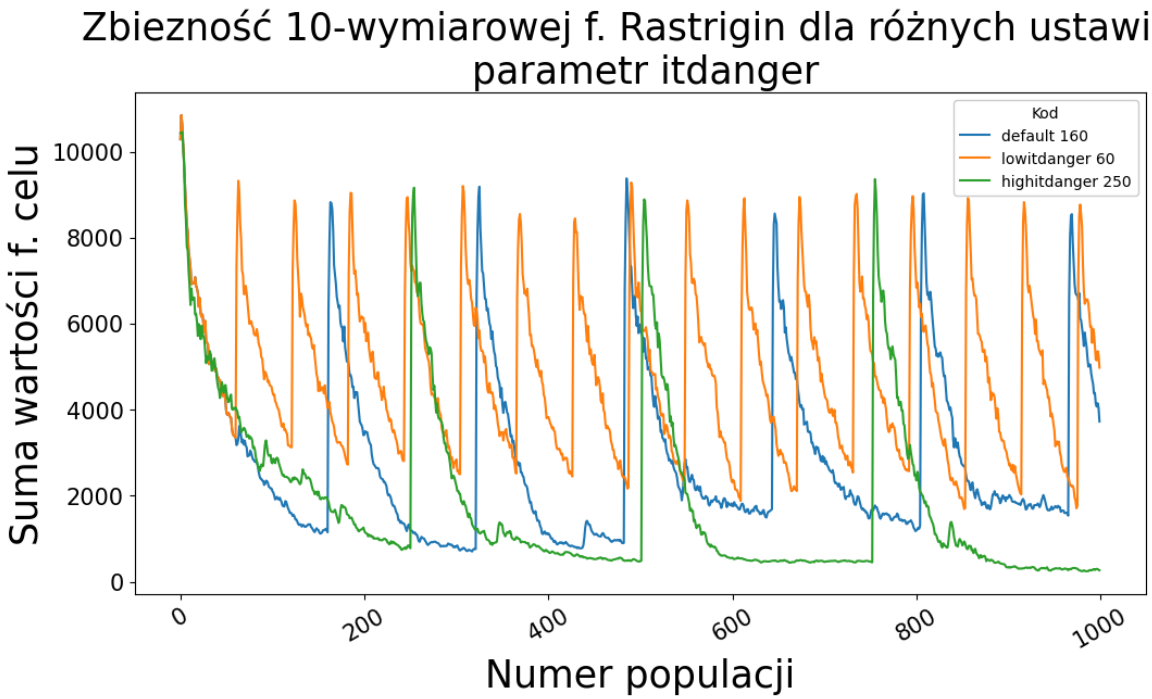
Zbieżność populacji w zależności od ustawień jednego z parametrów

Parametr c2, czyli czynnik społeczny. Zauważalne zjawiska są dokładnie odwrotne od tych dla c1. Za niska wartość = brak społeczności = słaba zbieżność. Zbyt wysoka wartość = ślepe gonienie najlepszych = szybsza zbieżność, ale potencjalnie ryzyko wpadki przy trudnych minimach lokalnych.



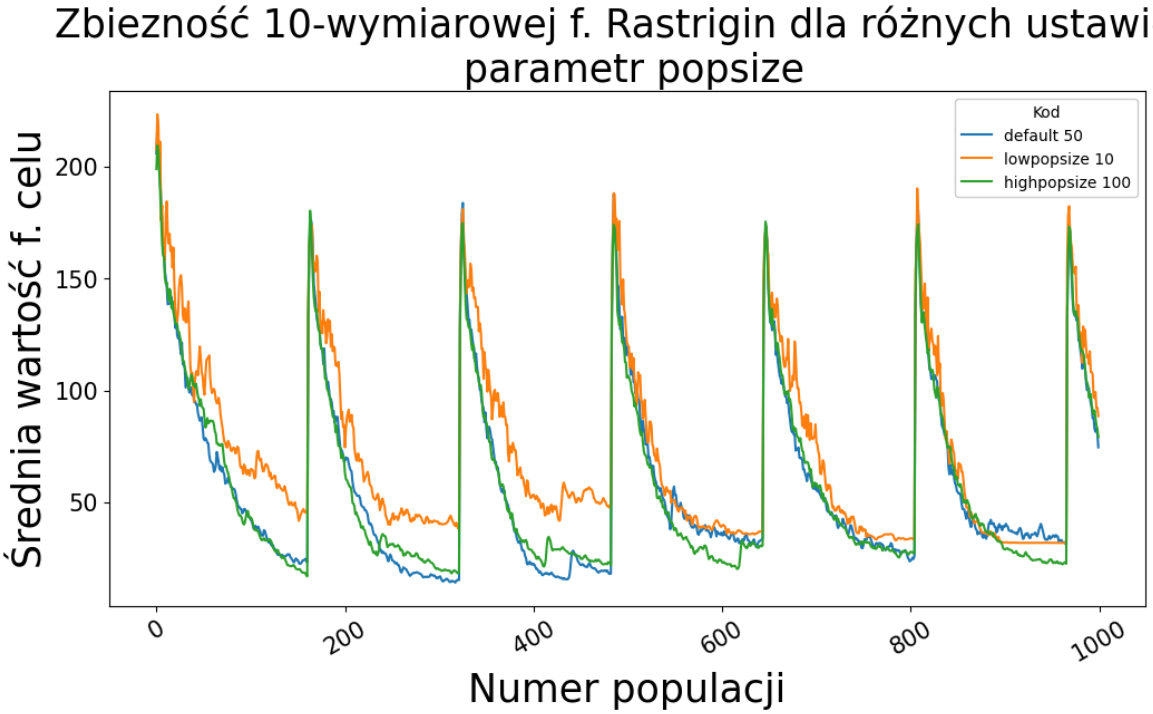
Zbieżność populacji w zależności od ustawień jednego z parametrów

Warto wspomnieć, że liczba sąsiadów w badaniu wynosiła 3. Zbyt duża wartość upodabniania do najlepszego sąsiada skutkuje słabą zbieżnością i dużymi skokami. Różny skład grup sąsiedzkich i ich losowe wartości początkowe sprawiają, że cząstki zbytnio upodabniając się do różnej jakości osobnika powodują wręcz losowo wyglądające zmiany. Za niska wartość, czyli ignorowanie sąsiadów przyśpiesza początkową zbieżność (bo jest bliższe ślepemu dążenia do najlepszego) jednak po pewnym czasie zaczyna tracić w porównaniu do podejścia uwzględniającego sąsiadów. Potencjalnym powodem jest zbyt mały czynnik eksploracji w wersji bez sąsiadów, co sprawia że populacja utyka w minimach lokalnych. Hipoteza zgadza się z wykresem, ponieważ pomarańczowy wykres niskiego wsp. sąsiedztwa co reset trafia do podobnego miejsca, podczas gdy ustawienie niebieskie (kompromis) raz jest lepsze, raz gorsze, a raz podobne – większa eksploracja.



Zbieżność populacji w zależności od ustawień jednego z parametrów

W przypadku it_danger łatwo zauważyć, że zbyt niska wartość oznacza częste restarty, co skutkuje brakiem możliwości zbiegu do niższej wartości – podobnie jak by było w przypadku zbyt niskiej łącznej liczby iteracji. Wysokie ustawienie sprawia, że algorytm ma więcej czasu na przeszukiwanie okolic optimum kosztem czasu wykonywania. W niektórych przypadkach takich jak okolice 600 iteracji występuje też sytuacja, gdy małe poprawki są nieopłacalne. Dodatkowo, jak pokazało badanie wieloczynnikowe większe wartości zmniejszają eksplorację i zwiększają ryzyko pogorszenia wyników przez minima lokalne, dlatego warto zachować umiar.



Zbieżność populacji w zależności od ustawień jednego z parametrów

Ostatnim parametrem jest popsize – z oczywistych powodów tutaj osią Y nie jest suma tylko średnie przystosowanie. Zgodnie z oczekiwaniami niższa liczebność powoduje wolniejszy zbieg w większości przypadków, ponieważ szansa na szczęśliwy traf jest niższa. Duża populacja z drugiej strony nie wykazuje żadnej stałej zależności w porównaniu do ustawienia domyślnego – raz jest lepiej, raz gorzej. To co jest pewne, to że dla większej populacji czas wykonania się wydłuża, dlatego warto podejść do ustawienia z umiarem.

5 Porównanie

Funkcja	Wymiar	C0	C1	C2	C3	Iter_danger	sasiedztwo	popsize	niter	seed	y
Ackley	1	1.5	0.1	0.1	0.1	160	3	50	1000	0	4.440892e-16
Ackley	5	1.5	0.1	0.1	0.1	160	3	50	1000	0	3.996803e-15
Ackley	10	1.2	0.1	0.1	0.1	160	3	50	1000	10	5.817672e-08
DeJong	1	1.7	0.1	0.3	0.1	160	3	50	1000	100	6.559724e-21
DeJong	5	1.35	0.1	0.1	0.1	160	3	50	1000	10	5.592148e-17
DeJong	10	1.35	0.1	0.1	0.1	160	3	50	1000	100	1.076982e-16
Rastrigin	1	1.7	0.1	0.1	0.1	160	3	50	1000	0	0.0
Rastrigin	5	1.35	0.1	0.1	0.1	160	3	50	1000	100	0
Rastrigin	10	1.7	0.1	0.1	0.1	160	3	300	1000	10	4.432029e-07

Tabela przedstawiająca ustawienia i wyniki PSO dla różnych zadań

Na podstawie wniosków z poprzednich sekcji uznano za nieopłacalne wykonywanie dalszych optymalizacji i do zdobycia wartości porównawczych zostały wykonane wyniki zbiorcze z analiz jednoczynnikowych dla wszystkich funkcji. Zgodnie z oczekiwaniami okazało się, że wyniki są świetne – nie raz błąd wynika głównie z niedokładności reprezentacji, czas wykonania nie przekracza paru dziesiątych sekundy, a najlepsza wartość dla danej funkcji jest osiągana przy wielu kombinacjach parametrów. Na przykład 9 ustawień dla 1-wymiarowej f. Ackley'a i aż 163 kombinacje dla f. Rastrigina z n = 1 (wynika to z niedokładności reprezentacji i obliczania tej funkcji – parametry miały rząd e-10 i się zmieniały, ale wynik był zwracany jako 0.0).

6 Wnioski ogólne

Porównując algorytm PSO do poprzednio zbadanych SA oraz AE można powiedzieć, że jest to algorytm bardzo wysokiej jakości szczególnie skuteczny dla badanych zadań ze względu na ich ciągły charakter. Pozwolił osiągnąć bliskie optimum wyniki szybciej i łatwiej niż poprzednicy, chociaż przewaga nad AE nie jest bardzo duża – może to jednak wynikać ze stosunkowej prostoty rozwiązywanych zadań.

W przeciwieństwie do poprzedników PSO w tej implementacji wykazuje się większą odpornością na niedokładne ustawienia, ponieważ posiada wiele możliwości sterowania balansem między eksploracją

i eksploatacją, a już dla domyślnych ustawień ten kompromis znajduje się w dobrym miejscu. Utrudnia to automatyczne przeszukiwanie, ale potencjalnie oznacza większą możliwość dopasowania się do skomplikowanych problemów i znalezienia lepszych rozwiązań.