

POP projekt

Adrian Murawski, Bogumił Stoma

29 stycznia 2025

1 Treść zadania

Zmodyfikuj algorytm ewolucji różnicowej w wariancie rand/1/bin tak, aby zasięg mutacji F był strojony zgodnie z metodą MSR (ang. Median Success Rule) oraz PSR (ang. Population Success Rule). Zbadaj działanie zmodyfikowanych wersji algorytmu, używając benchmarku CEC 2017, dla wymiarowości D = 10,30.

2 Opis algorytmu

Algorytm ewolucji różnicowej działa następująco: po inicjalizacji populacji dla każdego punktu wybiera się punkt roboczy oraz dwa losowe punkty, które służą do modyfikacji. Tworzy się nowy punkt na podstawie wyskalowanej różnicy tych punktów, a następnie krzyżuje go z punktem roboczym. Jeśli nowy punkt jest lepszy, zastępuje on i-ty punkt w populacji. Proces trwa do spełnienia warunku stopu (np. maksymalna liczba iteracji).

W wariancie rand/1/bin punkt roboczy wybierany jest losowo, modyfikacja korzysta z jednej pary punktów, a krzyżowanie jest binarne.

Czynnik skalujący F jest strojonym parametrem. Metoda Median Success Rule dostosowuje F w zależności od różnicy median wartości funkcji celu dla populacji – większa liczba sukcesów zwiększa F (eksploracja), a mniejsza go zmniejsza (eksploatacja). Metoda Population Success Rule zmienia F w zależności od stosunku sukcesów w populacji do ustalonego poziomu docelowego – większy stosunek zwiększa F, mniejszy go zmniejsza.

3 Przeprowadzone eksperymenty

W ramach pracy przeprowadzono testy na benchmarku CEC 2017, obejmującym zestaw funkcji testowych o zróżnicowanym stopniu trudności, charakteryzujących się wielowymiarowością i obecnością wielu minimów lokalnych. Algorytm ewolucji różnicowej został przetestowany dla wymiarów 10 i 30.

3.1 Kryteria oceny wyników

Wyniki eksperymentów oceniano na podstawie następujących kryteriów:

- Średnia wartość funkcji celu po określonej liczbie iteracji,

- **Wariancja wyników** w ramach wielu uruchomień algorytmu,
- **Szybkość zbieżności** do rozwiązania.

Dodatkowo przeanalizowano wpływ wymiarowości na skuteczność algorytmu.

3.2 Analiza modyfikacji algorytmu

Po dokumentacji wstępnej zmieniono zakres testów - parametr CR nie był przedmiotem zainteresowania testów - ustalono go na 0.9 dla wszystkich uruchomień. Analogicznie wielkość populacji oraz maksymalna ilość iteracji były zależne jedynie od wymiaru. Szczegółowo zbadano wpływ wprowadzonych modyfikacji, takich jak:

- **Median Success Rule (MSR)** – dostosowanie współczynnika mutacji F na podstawie mediany różnic wartości funkcji celu,
- **Population Success Rule (PSR)** – strojenie F w zależności od stosunku sukcesów do liczby osobników w populacji.

3.3 Badanie parametrów algorytmu

Przeprowadzono eksperymenty mające na celu ocenę zależności wyników od kluczowych parametrów, takich jak:

- Współczynnik mutacji F ,
- Amplifier/reductor ,
- target ratio (PSR) ,
- threshold (MSR) .

4 Eksperymenty i dyskusja wyników

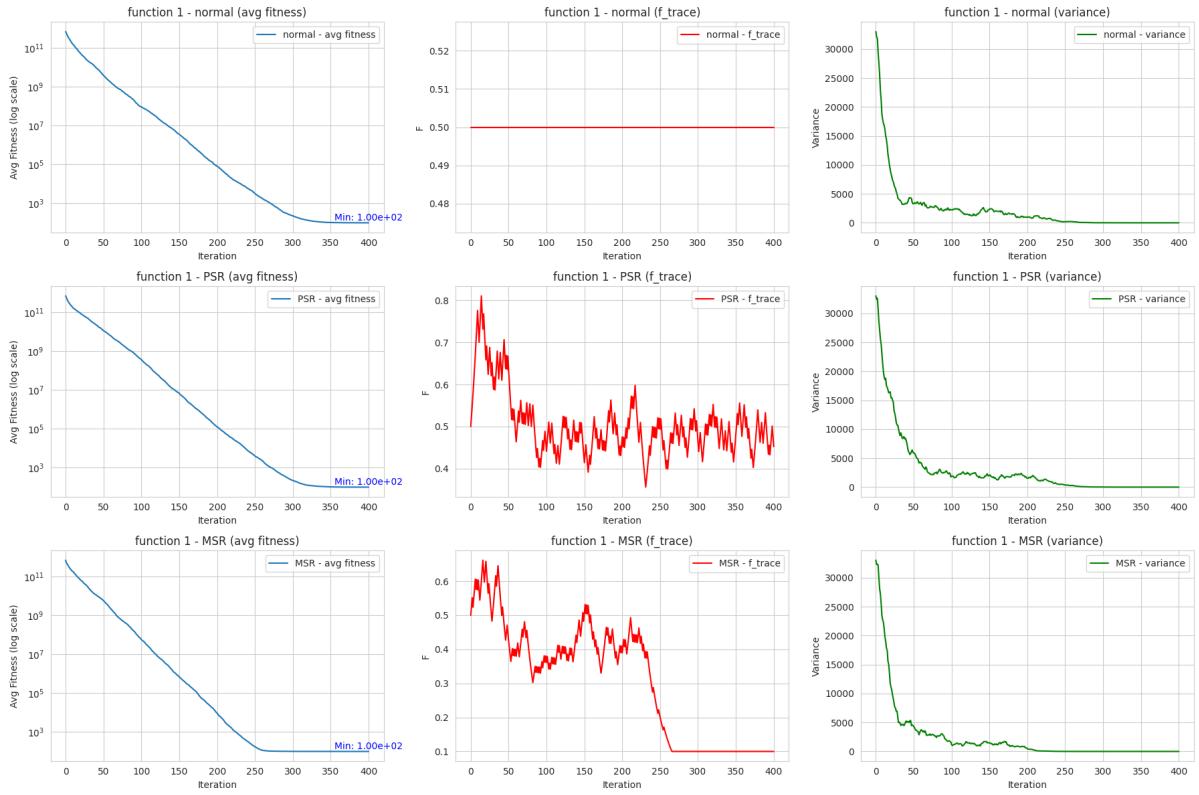
4.1 Uwaga

Dla lepszej czytelności pracy ukazujemy wyniki dla pewnych arbitralnie wybranych funkcji, reszta wykresów znajduje się w repozytorium. Dodatkowo na końcu agregujemy wyniki dla pokazania wykresu dla wszystkich funkcji w benchmarku jednocześnie.

4.2 Demonstracja działania zmodyfikowanych wersji

Pomyślnie zaimplementowano metody MSR i PSR do aktualizowania wartości F w trakcie działania algorytmu. Przeprowadzono eksperiment polegający na uruchomieniu każdej z trzech wersji algorytmu dla każdej z funkcji oraz obu wymiarów. Zapisywano średnie wartości funkcji celu, wariancję populacji oraz wartość F w czasie trwania algorytmu. Na tej podstawie stworzono wykresy przedstawiające te wielkości dla każdego trybu działania algorytmu - normalnego, z PSR oraz z MSR.

Wyniki dla przykładowo wybranej funkcji i wymiaru prezentujemy poniżej.



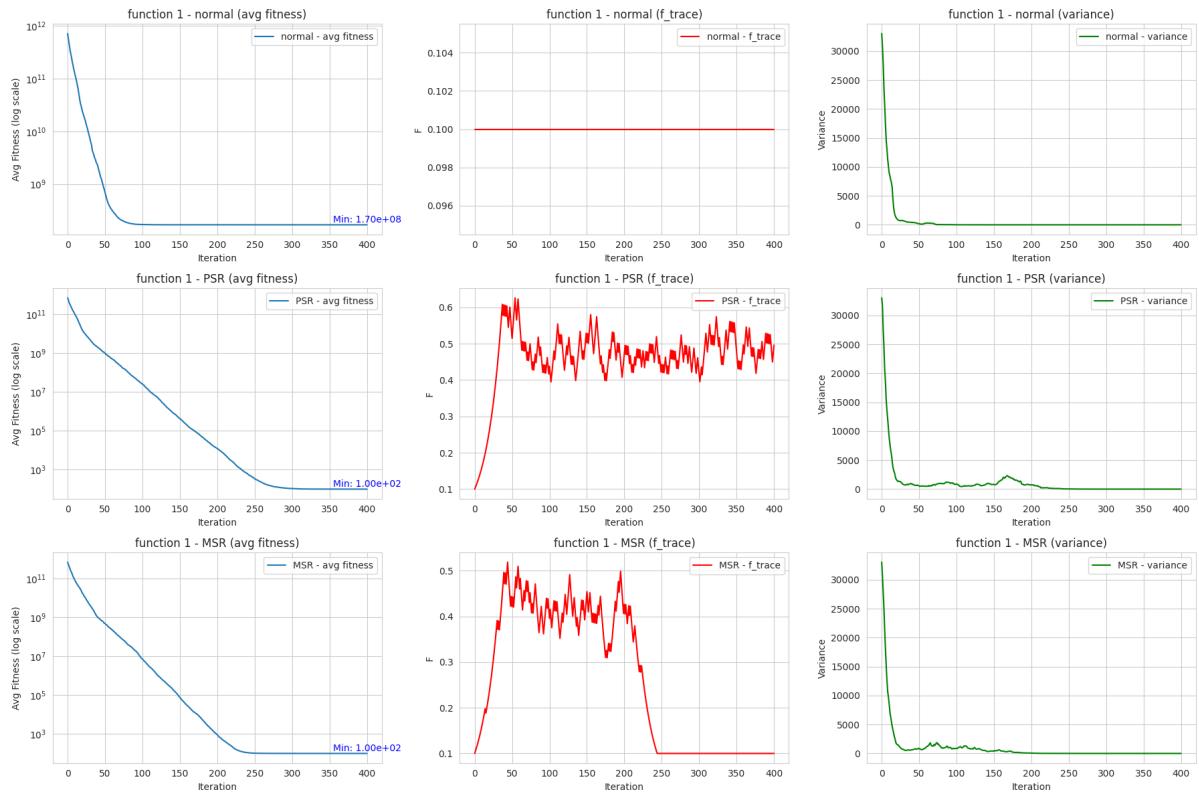
Rysunek 1: Funkcja nr 1, wymiar = 10, początkowe F = 0.5

Można zauważyć, że wartość F dla normalnego uruchomienia jest stała, a jeśli włączymy metodę PSR lub MSR, to jest ona aktualizowana w trakcie trwania algorytmu, co odpowiada oczekiwanej zachowania algorytmu.

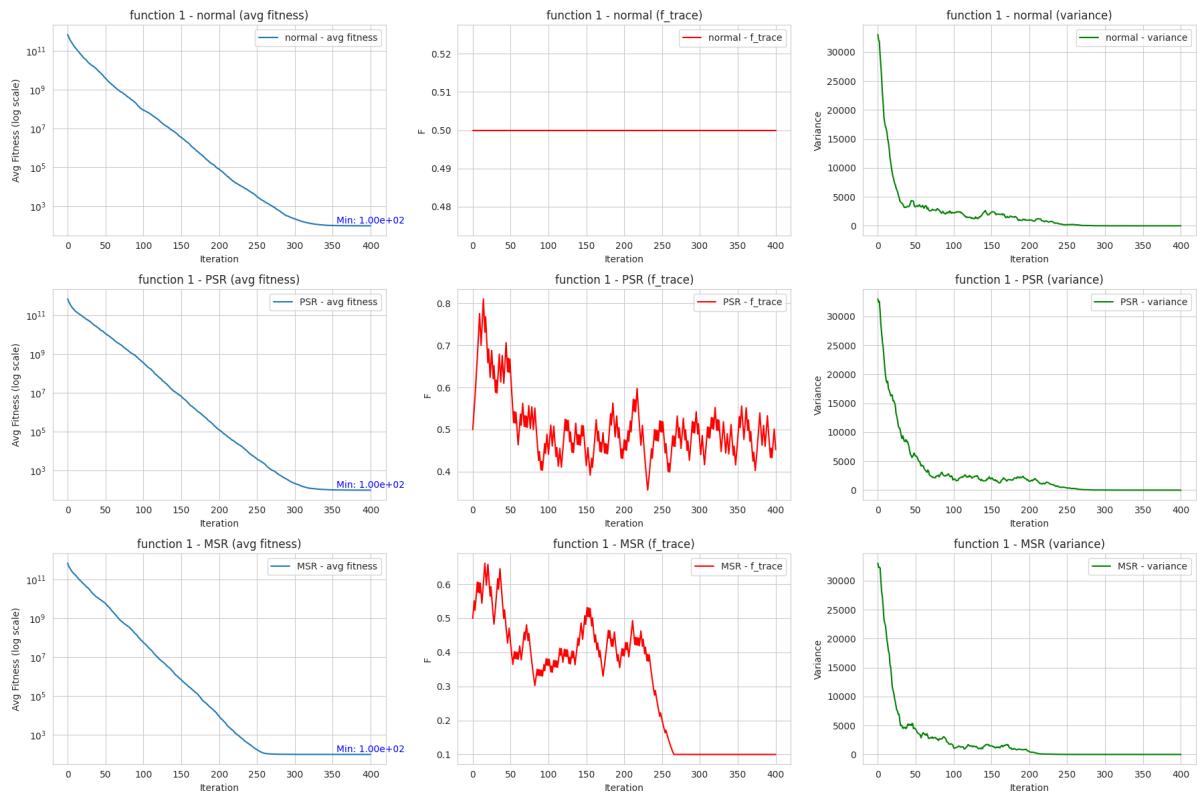
4.3 Parametr F

Zbadano także działanie funkcji dla różnych początkowych wartości F. Przyjęto wartości 0,1; 0,5 oraz 0,9.

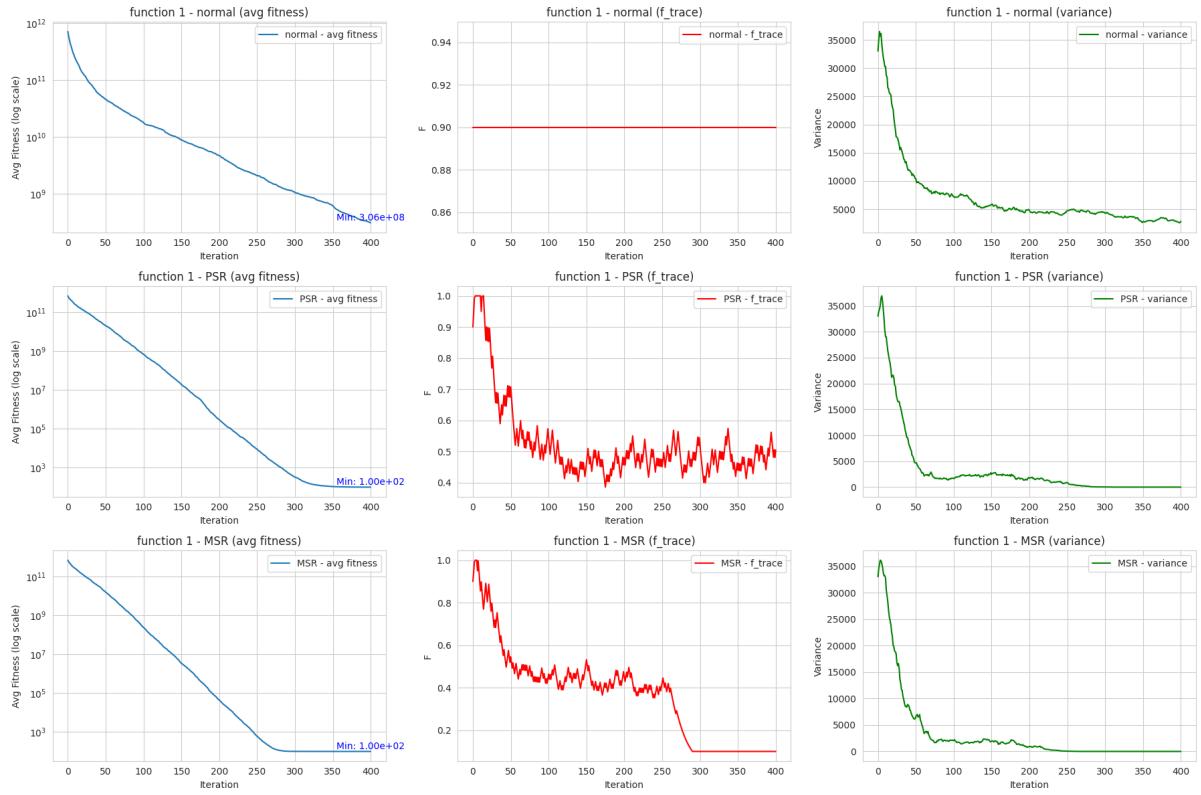
Na przykładzie funkcji nr 1 można zademonstrować jak początkowy parametr F może negatywnie wpływać na optymalizację funkcji w przypadku zwykłego algorytmu DE.



Rysunek 2: funkcja nr 1, wymiar = 10, $F=0.1$



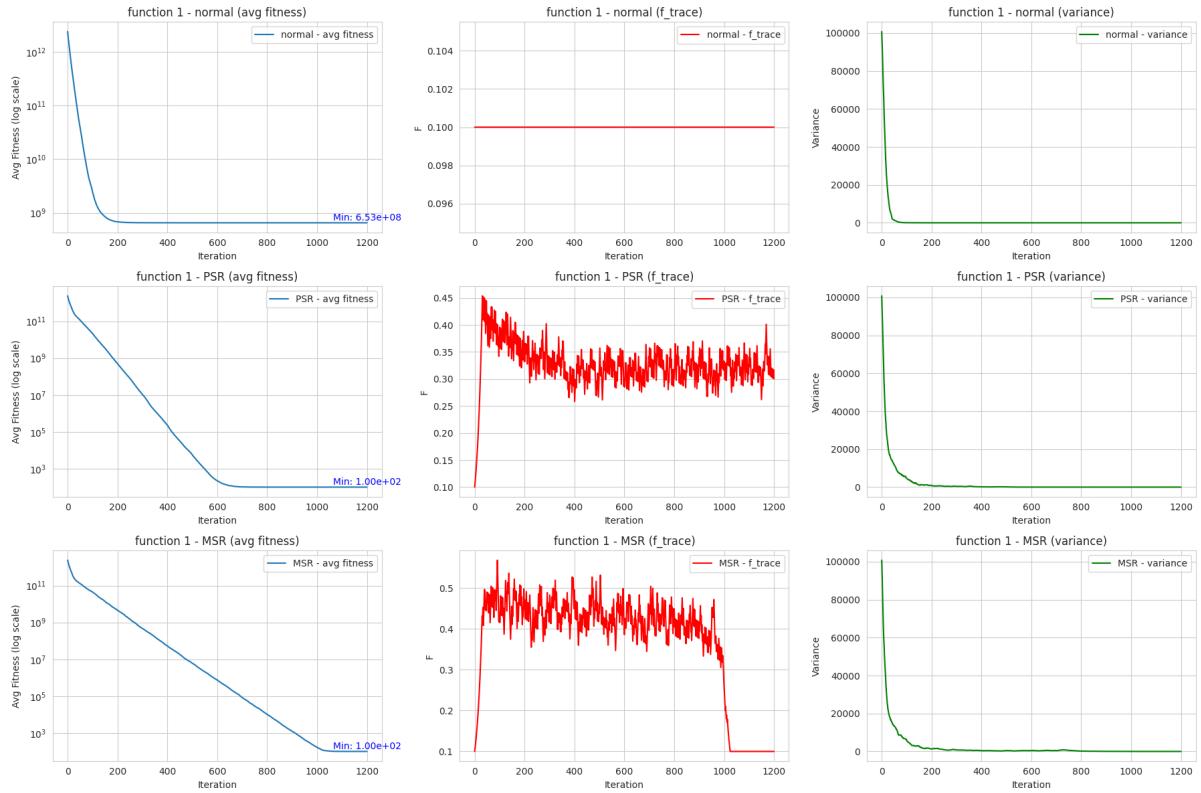
Rysunek 3: funkcja nr 1, wymiar = 10, $F=0.5$



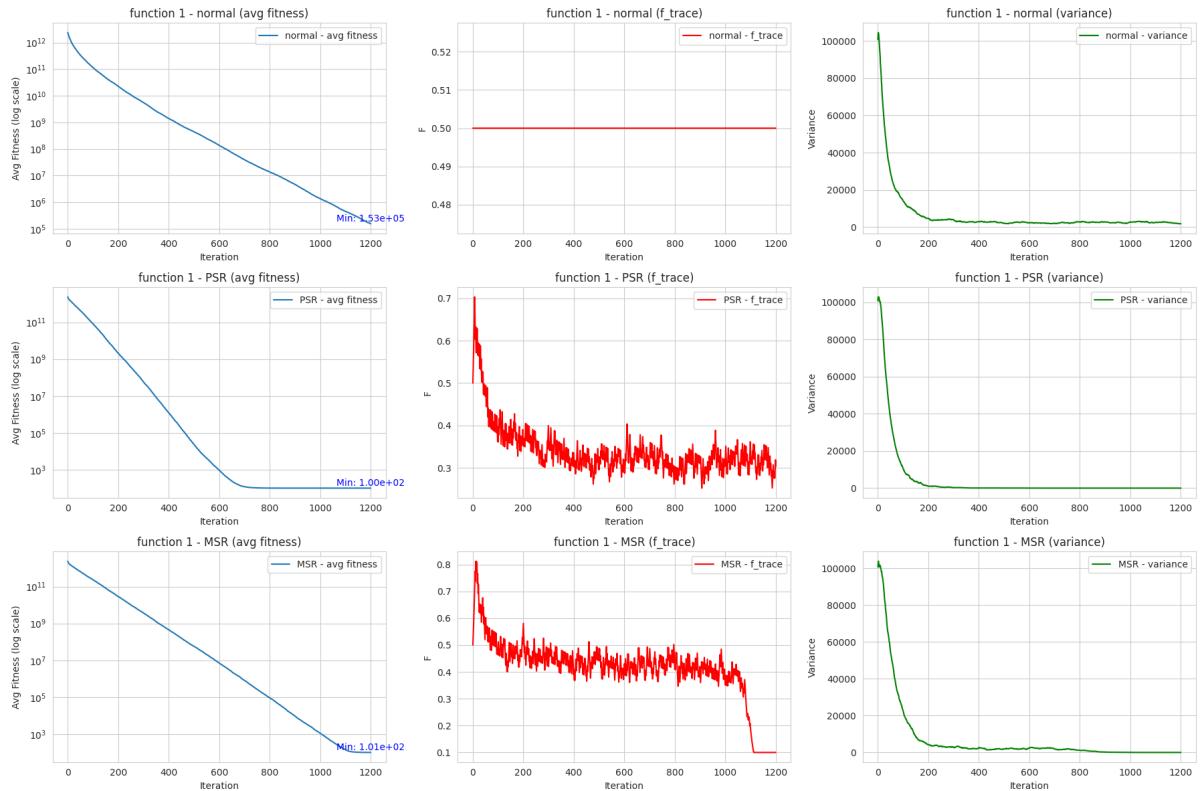
Rysunek 4: funkcja nr 1, wymiar = 10, F=0.9

Dla wymiarowości 10, zwykły algorytm DE nie radzi sobie, jeśli parametr F zostanie ustawiony skrajnie - mały lub duży. Dla F = 0,5 algorytm radzi sobie ze znalezieniem optimum. Zmodyfikowane wersje algorytmu (PSR i MSR) dzięki bieżącemu dopasowywaniu wartości parametru F, radzą sobie ze znalezieniem optimum niezależnie od wartości początkowej tego atrybutu.

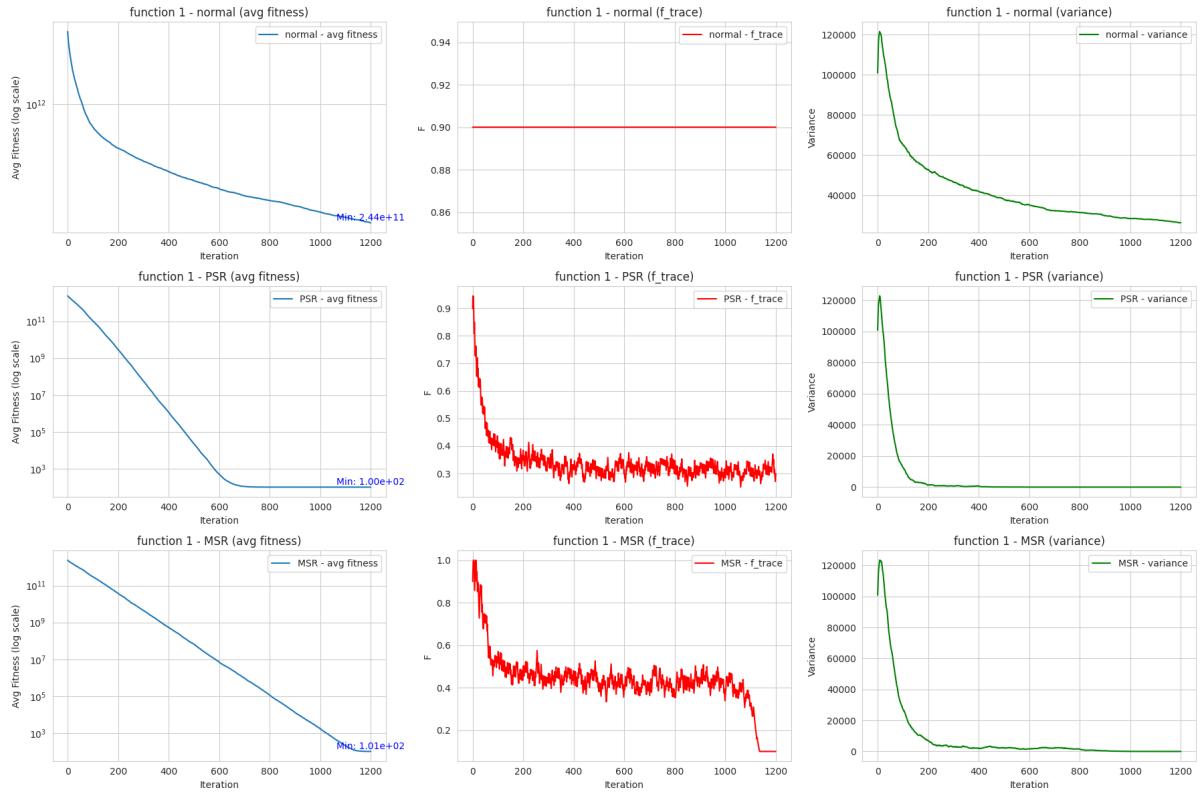
Przeprowadzone eksperymenty dla wymiarowości 30 pokazują, że w bardziej skomplikowanym przypadku, dobranie wartości F może być trudne.



Rysunek 5: funkcja nr 1, wymiar = 30, $F=0.1$



Rysunek 6: funkcja nr 1, wymiar = 30, $F=0.5$



Rysunek 7: funkcja nr 1, wymiar = 30, F=0.9

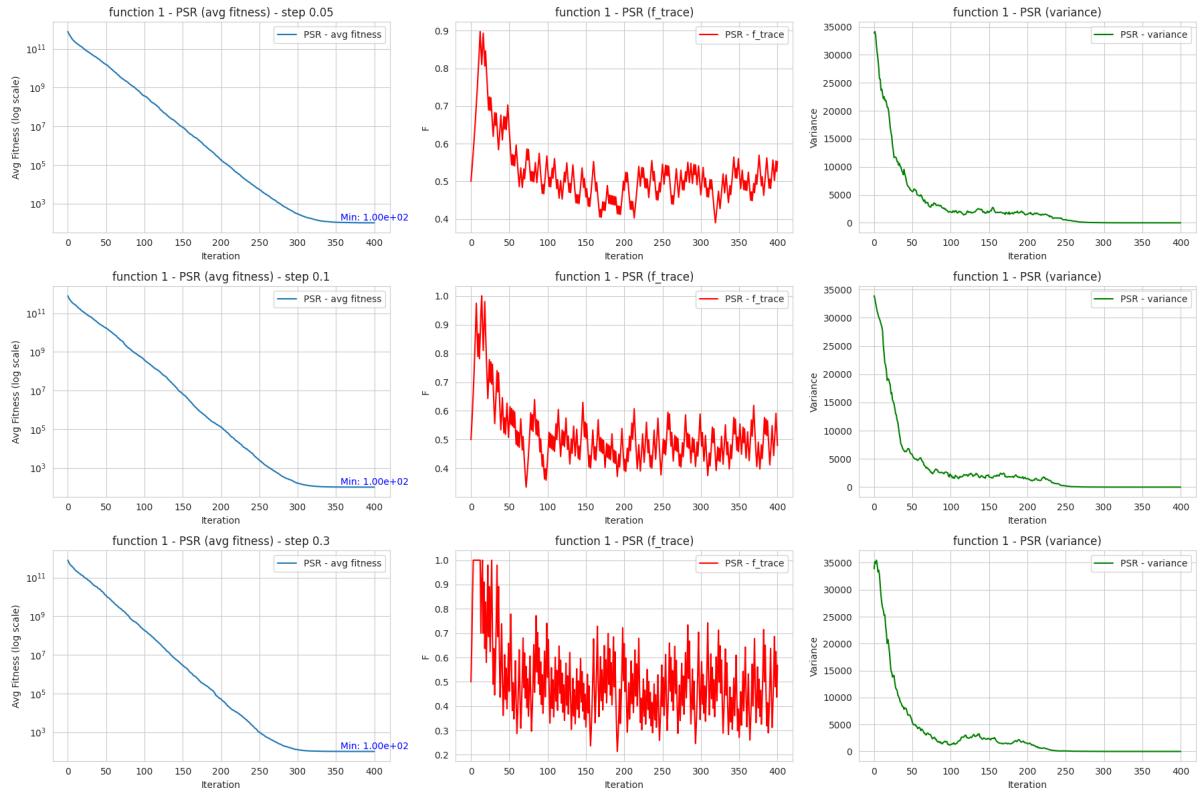
W tym przypadku algorytm DE w podstawowej wersji nie znajduje optimum także dla F = 0,5 (w podanym zakresie iteracji; widać, że tendencja jest dobra, wartość średniej funkcji celu maleje, ale pokazuje to większe zapotrzebowanie czasowe tego algorytmu na znalezienie optimum).

Zmiany F w algorytmach PSR i MSR skutkowały niewielkim wpływem tego parametru na końcowe wyniki. Pokazuje to, jak dużą zaletą zmodyfikowanych wersji jest zdolność do adaptacji parametru F.

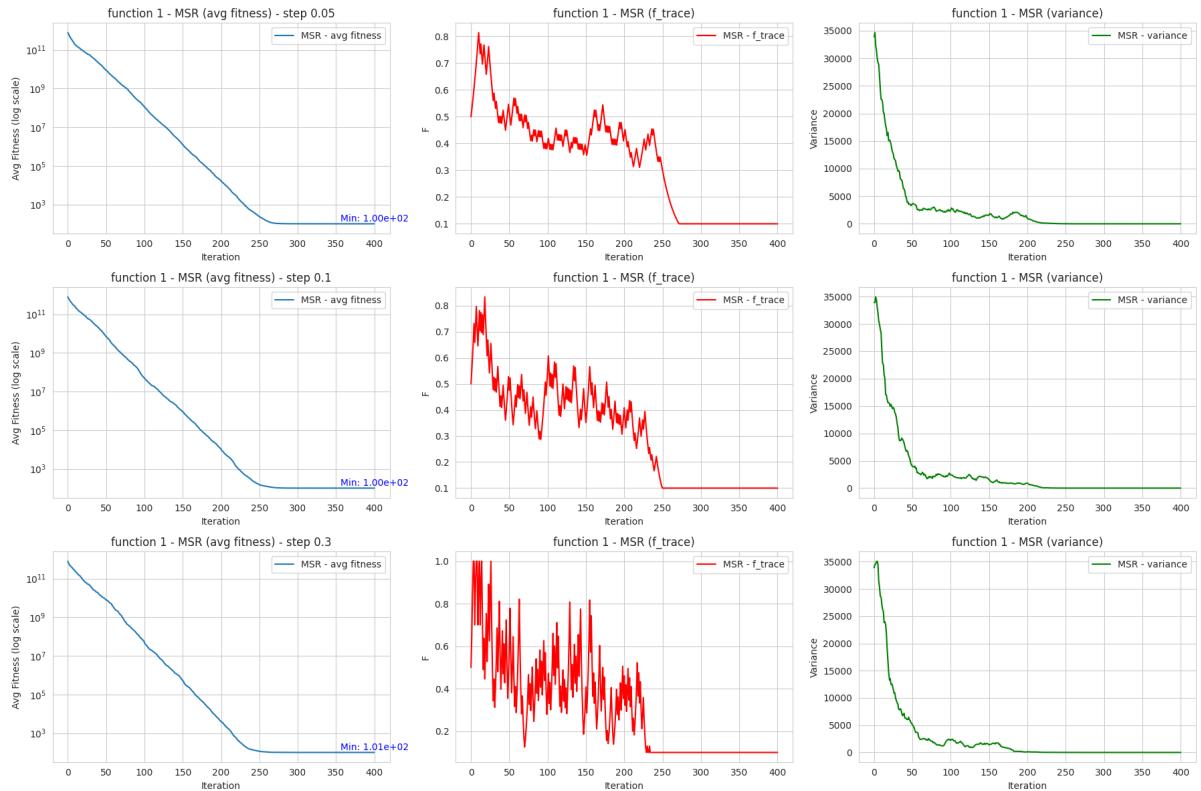
4.4 Parametr amplifier/reductor

Przeprowadzono eksperymenty polegające na sprawdzeniu jak różne wartości parametrów amplifier / reductor wpływają na zachowanie zmodyfikowanych wersji algorytmu.

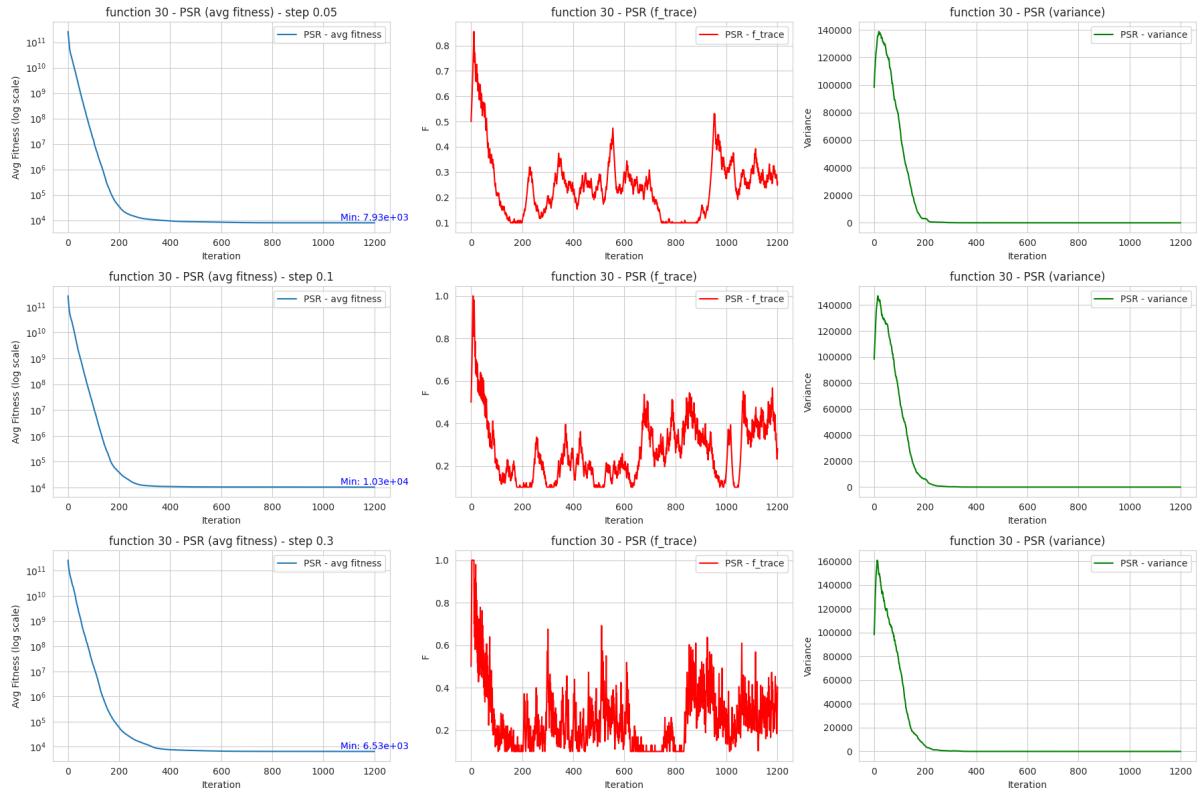
Zmiana zachowania algorytmu jest bardzo dobrze widoczna dla funkcji nr 1 lub nr 30 z benchmarku CEC2017. Prezentujemy wyniki poniżej.



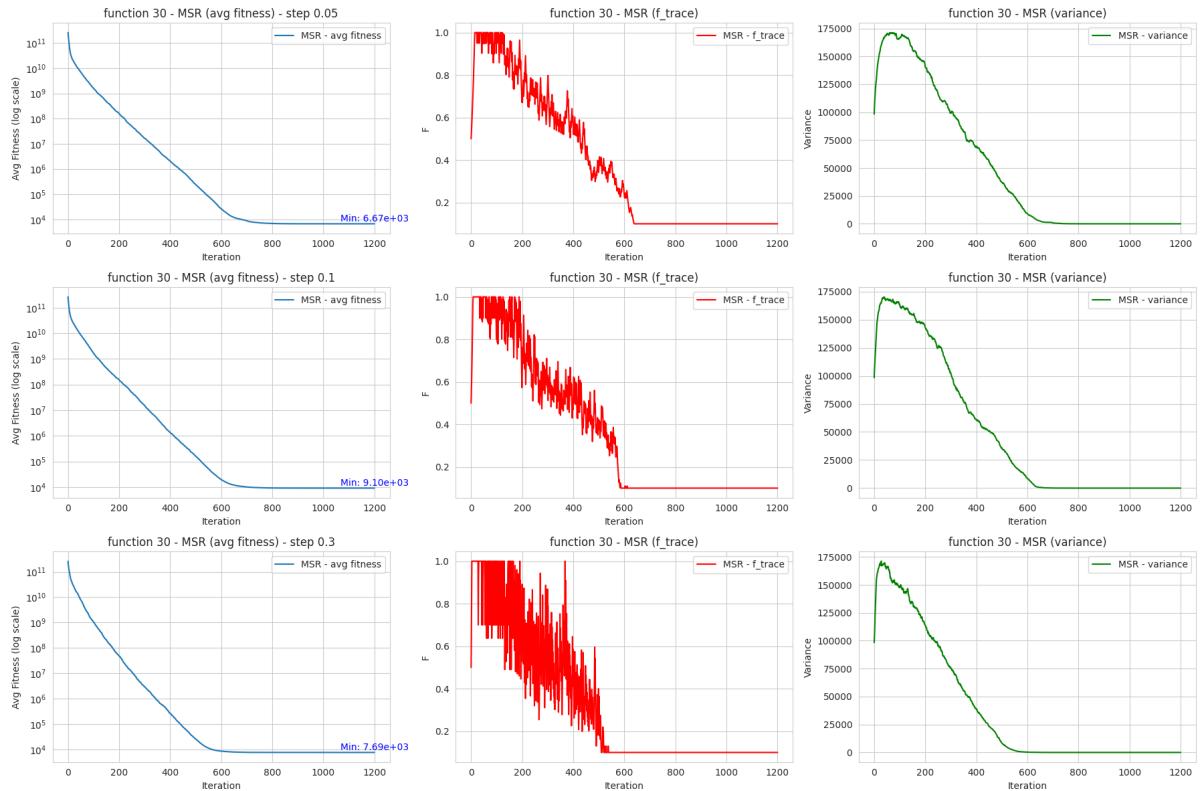
Rysunek 8: Funkcja nr 1, wymiar = 10, PSR



Rysunek 9: Funkcja nr 1, wymiar = 10, MSR



Rysunek 10: Funkcja nr 30, wymiar = 30, PSR



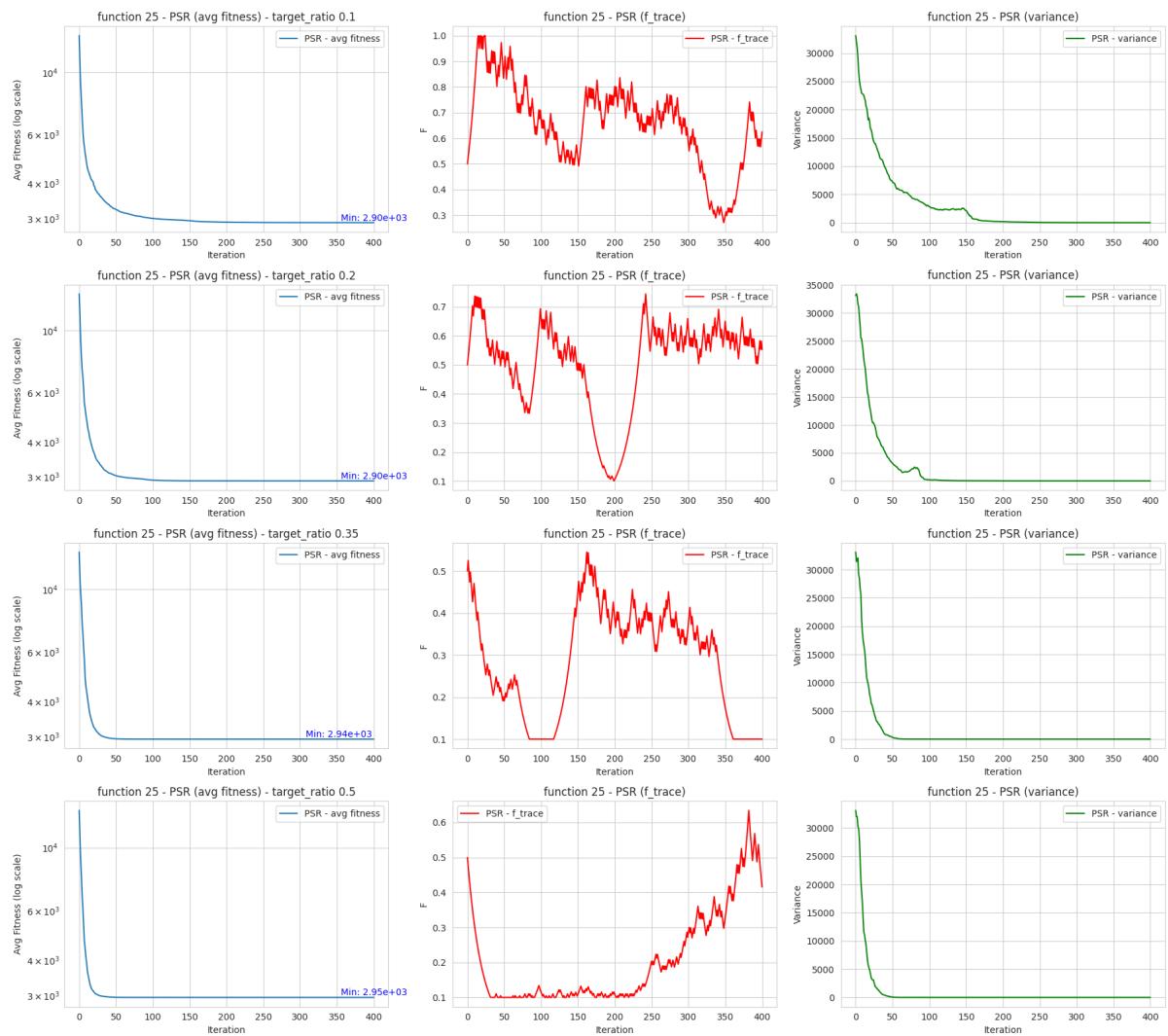
Rysunek 11: Funkcja nr 30, wymiar = 30, MSR

Bardzo dobrze widać, że im większa wartość step - równa różnicy (co do wartości bezwzględnej) między amplifier a wartością 1 oraz analogicznie między reductor a wartością 1, tym występują większe wahania wartości F.

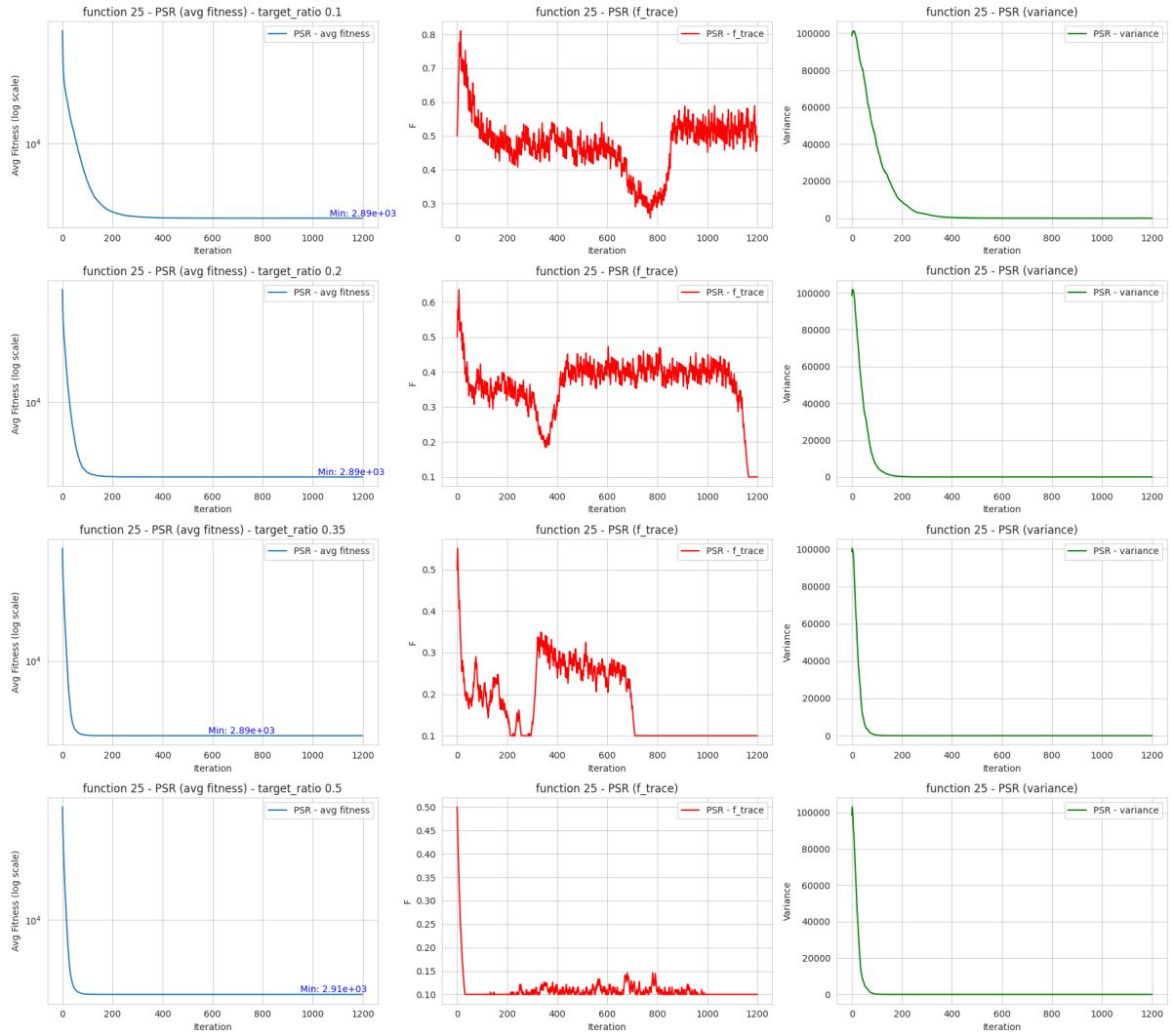
4.5 Parametr target ratio

Następnym z przeprowadzonych eksperymentów było sprawdzenie wpływu parametru target ratio dla metody PSR - które jest oczekiwany stosunkiem liczby sukcesów do wielkości populacji.

Zauważono, że dla pewnych funkcji parametr ten ma wpływ na dynamikę zmian parametru F oraz szybkość zbieżności do optimum, natomiast nie wpływa na końcową skuteczność optymalizacji - najmniejsza średnia wartość funkcji celu w populacji jest dla wszystkich uruchomień podobna. Poniżej przykład takiej funkcji.

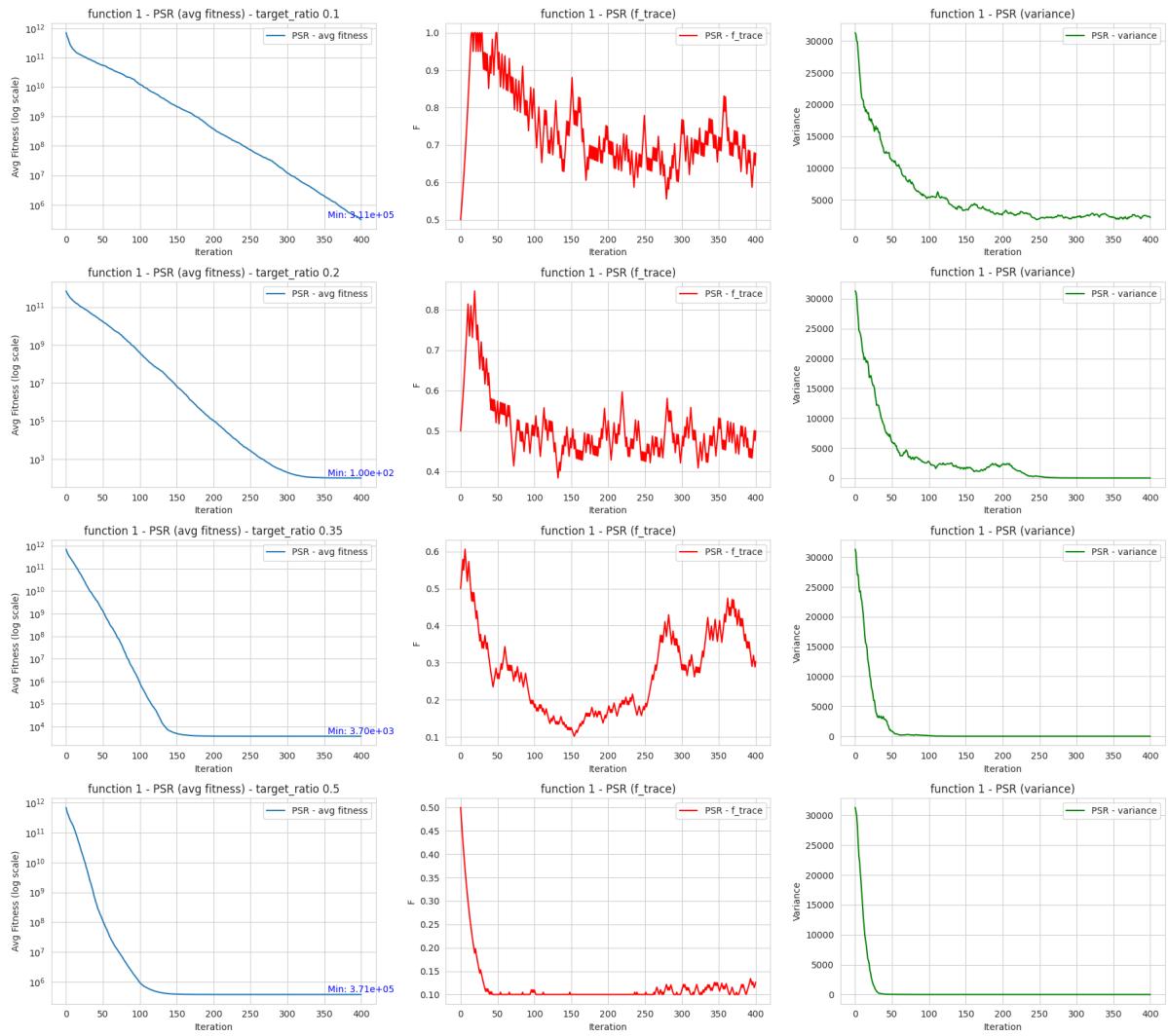


Rysunek 12: funkcja nr 30, wymiar = 10

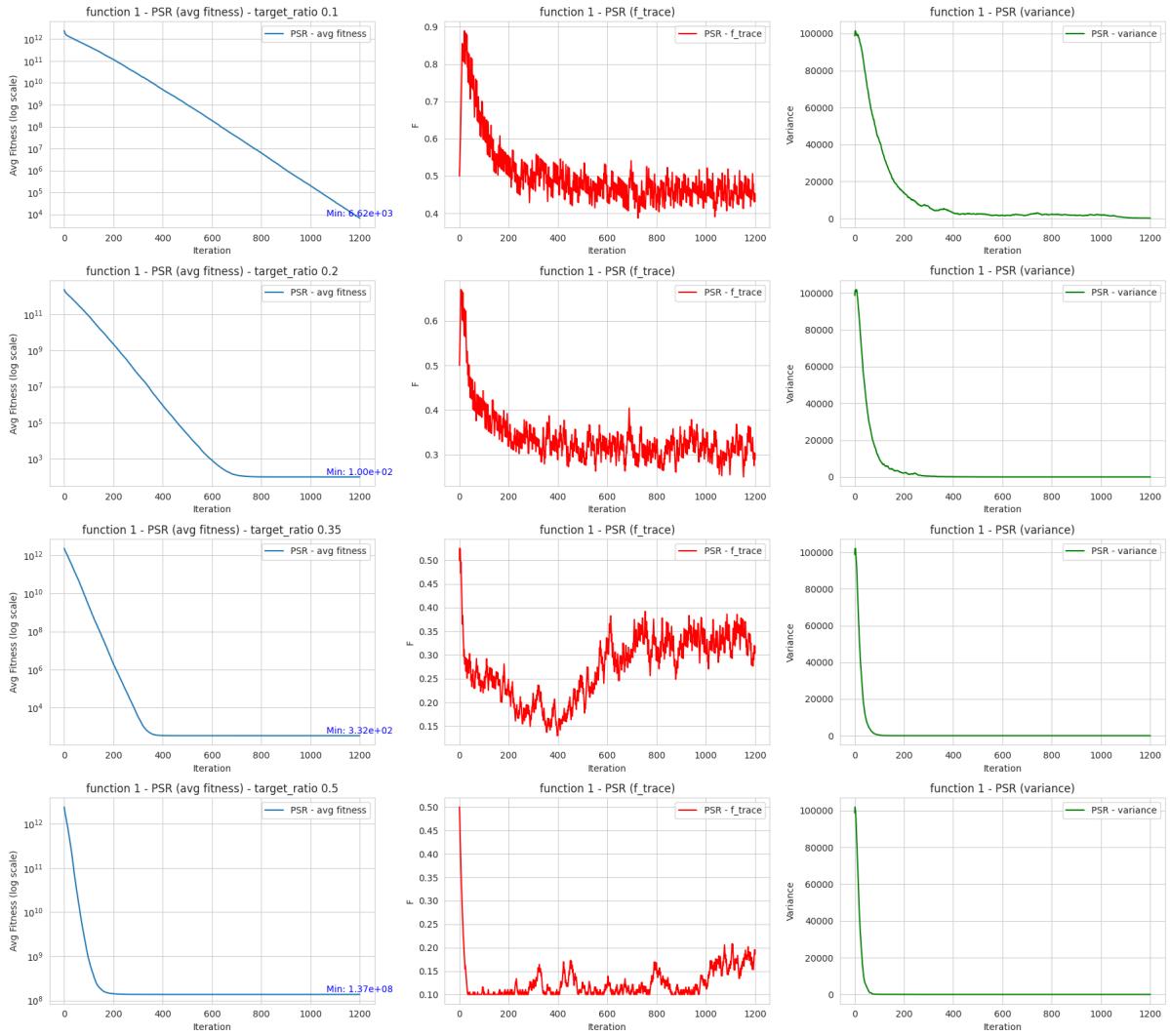


Rysunek 13: funkcja nr 25, wymiar = 30

Z kolei dla innych funkcji, wartość tego parametru mogła, poza dynamiką, zdecydować o optymalności znalezionych rozwiązań. Taka charakterystyka wystąpiła dla funkcji nr 1.



Rysunek 14: funkcja nr 1, wymiar = 10



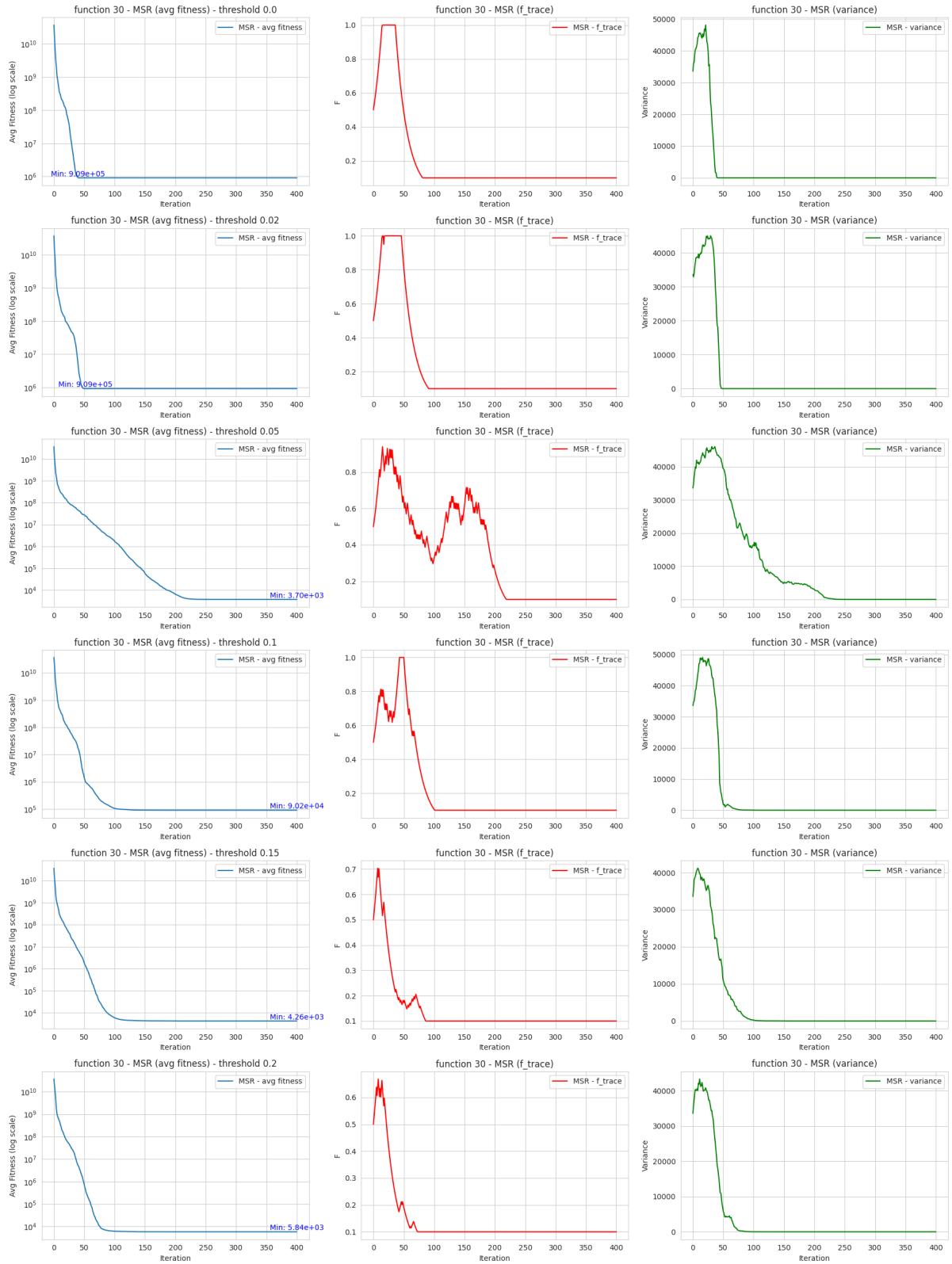
Rysunek 15: funkcja nr 1, wymiar = 30

Na powyższych wykresach widać, że dla target ratio równemu około 0,2 algorytm znajduje optymalne rozwiązanie, a dla wartości mniejszych bądź większych optymalne rozwiązanie nie jest osiągane.

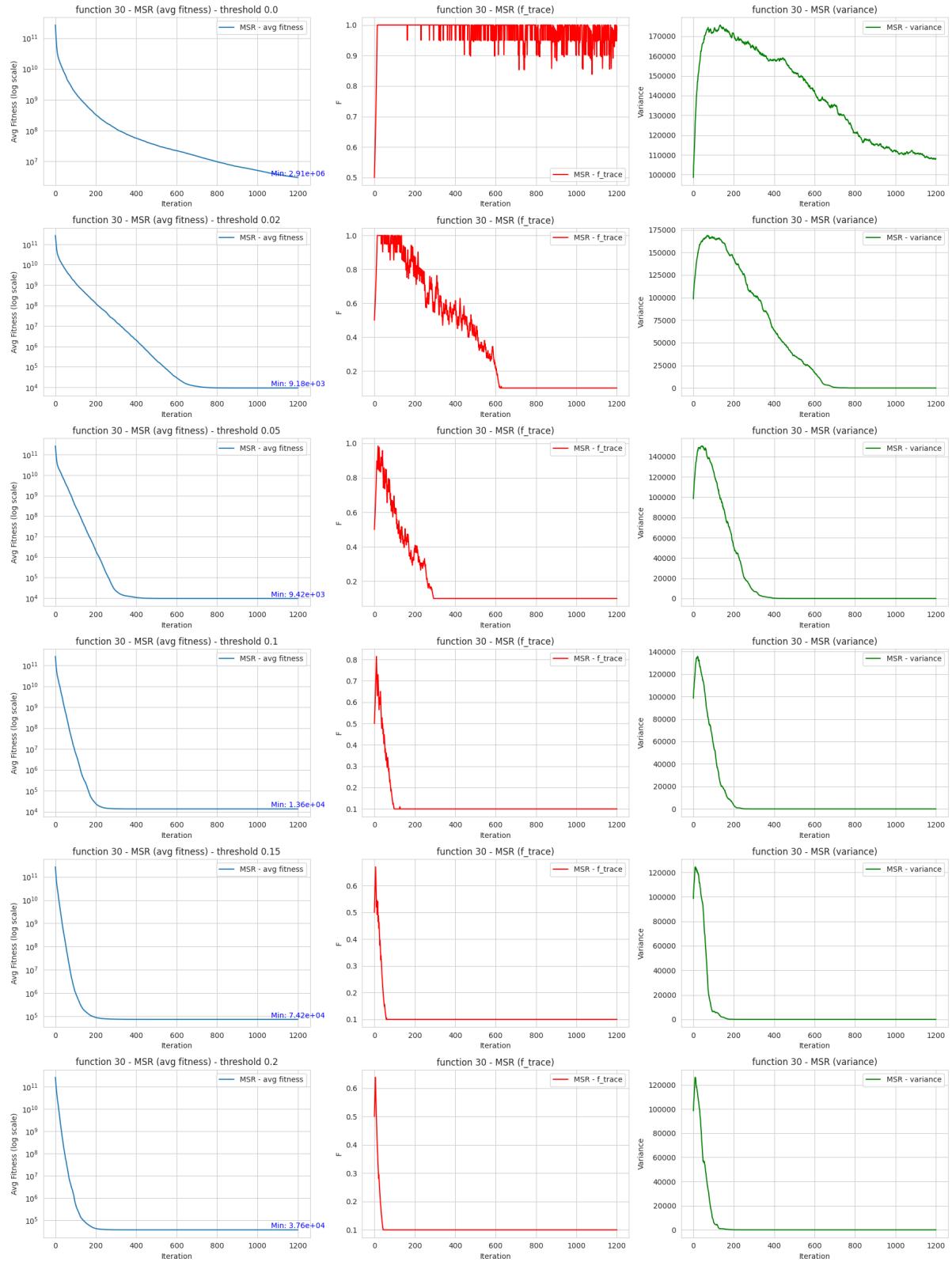
4.6 Parametr threshold

Parametr threshold wpływa na aktualizowanie wartości F w metodzie MSR. Jest on pewnym marginesem, mówiący o tym, o ile procent mediana nowej populacji musi się poprawić względem mediany poprzedniej populacji, aby można było zwiększyć parametr F i tym samym zwiększyć eksplorację.

W trakcie eksperymentów zauważono, że dla niektórych funkcji ten parametr nie wpływa na rozwiązanie, ale dla pewnych złożonych funkcji jest on kluczowy, aby algorytm mógł dojść do optymalnego rozwiązania. Przykładem takiej funkcji może być funkcja nr 30.



Rysunek 16: funkcja nr 30, wymiar = 10



Rysunek 17: funkcja nr 30 wymiar = 30

Brak takiego marginesu (threshold = 0.0) często skutkował nadmiernym zwiększeniem parametru F, co prowadziło do nieoptimalnych wyników. Większa wartość tego parametru wpływała na dynamikę zmian wartości F oraz szybkość utraty różnorodności populacji - zmniejszanie się wariancji.

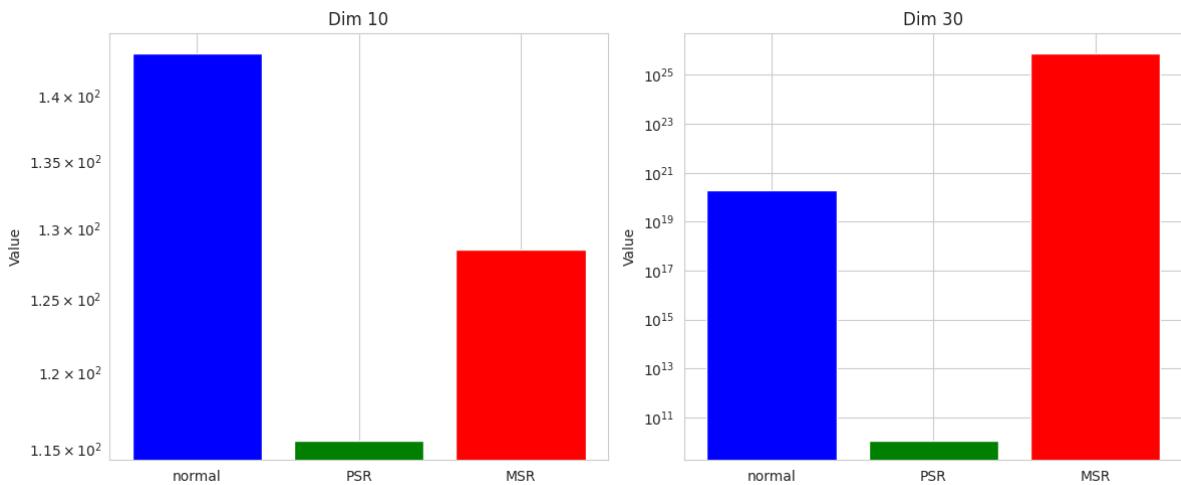
5 Sumaryczny test na benchmarku CEC2017 i wnioski

Na podstawie eksperymentów wybrano początkową wartość parametru F jako 0.5, target ratio jako 0.2 (dla PSR) i threshold percentage jako 0.08 dla wymiaru 10 i 0.02 dla wymiaru 30 (dla MSR).

Na wszystkich funkcjach z benchmarku przeprowadzono test, który polegał na pięciokrotnym uruchomieniu danego algorytmu na danej funkcji i wymiarze. Zapisywano najmniejszą (spośród tych 5 uruchomień) średnią wartość funkcji celu z całej populacji w dowolnym momencie (dla dowolnej iteracji). Następnie, dla każdego z algorytmów, oddziennie dla wymiaru 10 i 30, zsumowano różnice tych minimalnych średnich i optimów dla kolejnych funkcji, którą ostatecznie podzielono na ilość wszystkich funkcji.

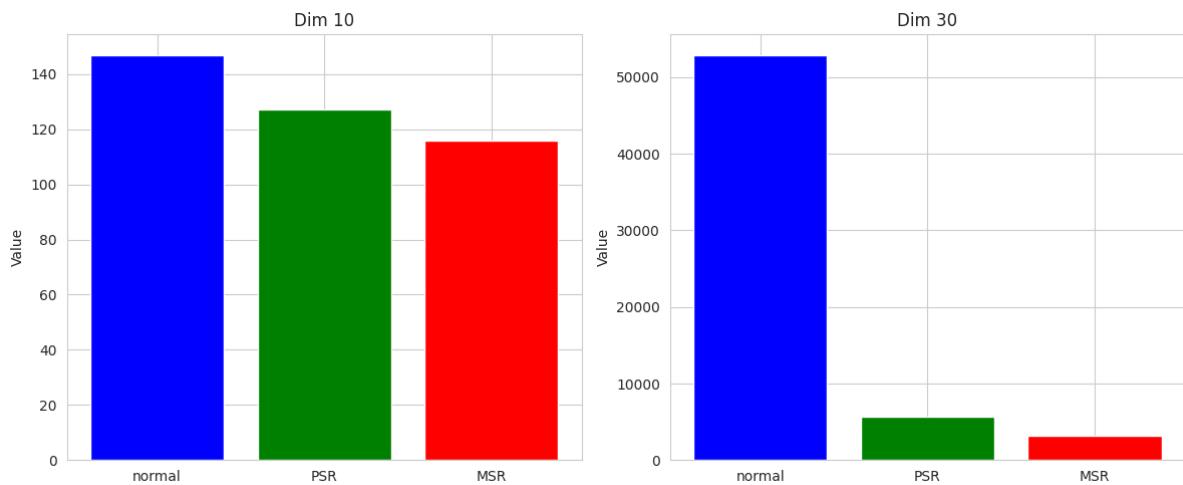
Dzięki temu otrzymano wskaźnik pozwalający stwierdzić o ile dany algorytm średnio myli się w wyliczaniu optimum danej funkcji dla całego benchmarku.

Otrzymano następujące wyniki:



Rysunek 18: Wskaźnik średniej pomyłki na funkcję w całym benchmarku

Po wnikliwym przeanalizowaniu wyników, zauważono, że bardzo duże wartości pomyłek metody zwykłej i MSR dla wymiaru 30 wynikają z pomyłek na funkcji nr 2. Postanowiliśmy pominąć tą funkcję (ze względu na informację o tym, że jest ona niestabilna i może być powodem niepoprawnych wyników) i uruchomić test jeszcze raz.



Rysunek 19: Wskaźnik średniej pomyłki na funkcję w całym benchmarku (poza f2)

Wyniki, które otrzymano, znacznie bardziej przypominają spodziewane rezultaty. Wiadać, że algorytm DE bez zmian dla wymiaru 10 radzi sobie porównywalnie - różnice nie są duże, natomiast dla wymiaru 30 radzi sobie zdecydowanie najgorzej.

Zmodyfikowane wersje algorytmu radzą sobie bardzo podobnie, różnice są minimalne (w przypadku 10 wymiarów na korzyść PSR, a 30 - MSR).

Ostatecznie algorytm DE z dodaniem dynamicznego zmieniania wartości F jest dużo bardziej stabilny. Zdolności optymalizacyjne takiego algorytmu w bardzo małym stopniu zależą od początkowej wartości F, co jest przeciwieństwem klasycznej wersji algorytmu, gdzie ta początkowa wartość często definiuje jak dobrze algorytm poradzi sobie z danym zadaniem optymalizacji. Można podejrzewać, że także tutaj wartość błędu dla klasycznej wersji może zmieniać się w zależności od ustalonego początkowego F, a dla zmodyfikowanych metod rezultaty będą podobne.