EχACT: Evolutionary Computation
and Global Optimization - Poland
https://exact.ipipan.waw.pl

Procedury oceny algorytmów optymalizacji ciągłej

Karol Opara[1] i Jarosław Arabas[2]

[1] Instytut Badań Systemowych Polskiej Akademii Nauk
[2] Instytut Systemów Elektronicznych Politechniki Warszawskiej

Streszczenie

Do oceny i porównywania efektywności algorytmów optymalizacyjnych wykorzystuje się standardowe zestawy problemów testowych zwane benchmarkami. Poniższa praca w przeglądowy sposób opisuje procedury benchmarkowania metod optymalizacji ciągłej. Przedyskutowano motywy, które ukształtowały ich formę, kryteria oceny algorytmów, typowe sposoby prezentacji i interpretacji wyników jak również odpowiednie procedury statystyczne. Rozważania prowadzone są na przykładzie benchmarków CEC i BBOB. Szczególną uwagę poświęcono zagadnieniom ich obiektywizmu i odporności na manipulacje. Podsumowaniem pracy jest przedstawienie perspektyw rozwoju procedur oceny algorytmów.

Słowa kluczowe: benchmarki, kryteria oceny, optymalizacja ciągła

1 Wstęp

Pytanie o to, która metoda optymalizacyjna może być uznana za „najlepszą" jest ważne zarówno z praktycznego jak i naukowego punktu widzenia. Pojęcie „najlepszego" algorytmu nie jest jednak jednoznacznie określone. Przesłanki teoretyczne okazują się bowiem niewystarczające do podjęcia decyzji, którą metodę wykorzystać dla danego zagadnienia. Z tych względów porównywanie algorytmów optymalizacyjnych dokonywane jest na podstawie benchmarków, czyli zestawów problemów testowych i odpowiednich kryteriów oceny. Dla uporządkowania dalszych rozważań warto krótko przypomnieć kilka kluczowych pojęć.

Zagadnienie optymalizacji ciągłej polega na poszukiwaniu argumentu x_{opt}, dla którego funkcja celu $f : D \to \mathbb{R}$ osiąga minimum, przy czym $D \in \mathbb{R}^n$.

$$x_{opt} = \arg\min_{x \in D} f(x) \tag{1}$$

Dziedzina funkcji celu D nazywana jest zbiorem dopuszczalnym. Często określona jest jako n-wymiarowy przedział (hiperkostka) $D = [l_1, u_1] \times [l_2, u_2] \times ... \times [l_n, u_n]$. Wartość funkcji celu w minimum globalnym oznaczana jest symbolem $f_{opt} = f(x_{opt})$. Algorytm optymalizacyjny można ogólnie opisać jako metodę służącą do wygenerowania serii punktów $x_1, x_2, ..., x_m \in D$. Najlepszą wartość funkcji celu znalezioną przez metodę optymalizacyjną oznacza się przez $f_{best} = \min_{i \in \{1,2,...,m\}} f(x_i)$, natomiast różnica $f_{best} - f_{opt}$ nazywana jest błędem optymalizacji.

Dobry algorytm optymalizacyjny rozwiązuje zadany problem dokładnie oraz szybko. Te dwie cechy nazywane są łącznie efektywnością (ang. *performance*). Pojęcie to ma jednak dosyć intuicyjny charakter i bywa formalizowane na różne sposoby. Ponadto, wszystkie metody optymalizacji ciągłej dają niedeterministyczne rezultaty. Jest to spowodowane dwiema przyczynami: losową inicjalizacją (np. doborem punktu startowego) oraz stochastyczną naturą nieklasycznych metod optymalizacji, w szczególności algorytmów ewolucyjnych. Eksperymentalne wyznaczanie efektywności wymaga zatem przeprowadzenia wielu niezależnych uruchomień algorytmu.

Metody optymalizacyjne są zazwyczaj porównywane za pomocą symulacji przeprowadzanych dla specjalnie przygotowanych zbiorów problemów testowych. Proces ten jest niebanalnym zadaniem wymagającym szczególnych umiejętności i wiedzy. Poniższa praca w przeglądowy sposób omawia kwestię benchmarkowania opisując: teoretyczne podstawy porównywania algorytmów, dostępne zestawy problemów testowych, kryteria oceny efektywności, interpretację uzyskiwanych wyników oraz procedury testowania ich statystycznej istotności. Celem tej pracy jest wskazanie zalet wynikających z wykorzystania systematycznych procedur porównawczych oraz zwrócenie uwagi na najważniejsze aspekty tego typu analiz. Ponadto, przedstawiono trudności występujące przy tworzeniu rankingu algorytmów oraz ocenę potencjalnych kierunków rozwoju benchmarków. Niniejszy artykuł stanowi rozwiniętą, polskojęzyczną wersję pracy (Opara i Arabas, 2011).

1.1 Obiektywne porównywanie algorytmów

Benchmarki stanowią odpowiedź na potrzebę obiektywnego porównywania algorytmów. Zagadnienie to można sformułować w postaci pytania: która metoda osiągnie „najlepsze" wyniki dla nowego problemu optymalizacyjnego o nieznanych właściwościach? Odpowiedzi można szukać poprzez badanie efektywności dla zestawu problemów testowych. Jako problem testowy rozumiana jest funkcja celu wraz z dodatkowymi czynnikami takimi jak zbiór dopuszczalny, obszar inicjalizacji, kryteria zatrzymania obliczeń, itp.

W praktyce, porównywanie algorytmów może być dokonane w dość łatwy sposób przy wykorzystaniu standardowych benchmarków. Takie podejście ma kilka zalet:

- Nie trzeba opracowywać własnych problemów i procedur testowych, co pozwala oszczędzić czas pracy oraz chroni przed ryzykiem popełnienia błędów metodologicznych.
- Korzystanie z benchmarków zwykle jest mało pracochłonne. Do przeprowadzenia oceny algorytmu wystarczy uruchomić odpowiednią procedurę[1] dla swojego algorytmu i przypisać czas obliczeniowy.
- Dzięki standaryzacji problemów testowych możliwe jest porównywanie rezultatów badanego algorytmu z wynikami uzyskanymi przez innych badaczy bez potrzeby powtarzania eksperymentów.
- Wyniki mogą być przetworzone i zaprezentowane w standardowy sposób.

1.2 Benchmarkowanie a darmowe lunche

Naukowcy i praktycy zasadniczo aprobują ocenianie algorytmów optymalizacyjnych w oparciu o benchmarki. Działanie takie bywa jednak krytykowane na podstawie twierdzenia *no free lunch theorem* udowodnionego i uzupełnionego w pracach (Wolpert and Macready, 1997) i (Köppen *et al.*, 2001). Twierdzenie to mówi, że dla dowolnego algorytmu oraz dowolnej miary efektywności lepsze od przeciętnych wyniki dla jednej klasy problemów muszą być skompensowane przez gorsze wyniki dla innej klasy problemów. Przeciętne wyniki w obrębie wszystkich możliwych problemów są zaś takie same dla każdego algorytmu. Rezultat ten oznacza, że nie istnieje jeden „najlepszy" algorytm ogólnego zastosowania. Jednakże, dla poszczególnych klas problemów pewne algorytmy dają lepsze wyniki niż inne.

Algorytm optymalizacyjny może działać efektywnie wówczas, gdy wykorzystuje regularności rozwiązywanego problemu (Weise *et al.*, 2009) takie jak symetria lub wypukłość funkcji celu. „Typowy" problem optymalizacyjny cechuje się natomiast losowym charakterem i brakiem struktury (English, 2000). Z drugiej strony, najczęściej rozwiązywane problemy optymalizacyjne zazwyczaj posiadają pewne regularności i w związku z tym stanowią jedynie niewielką część wszystkich możliwych problemów. Przykładowo, moc zbioru wszystkich funkcji $f : \mathbb{R} \to \mathbb{R}$ wynosi $2^{\mathfrak{c}}$ podczas gdy moc zbioru funkcji ciągłych $f : \mathbb{R} \to \mathbb{R}$ jest mniejsza i równa kontinuum $\mathfrak{c}$[2]. Benchmarkowanie można zatem wykorzystać do wyboru algorytmów, które dają najlepsze wyniki na niewielkiej, ale ważnej klasie funkcji ciągłych. Obniżona efektywność dla problemów nieciągłych (a zwłaszcza nieciągłych w każdym punkcie) wydaje się mieć niewielkie znaczenie praktyczne.

1.3 Co mierzą benchmarki?

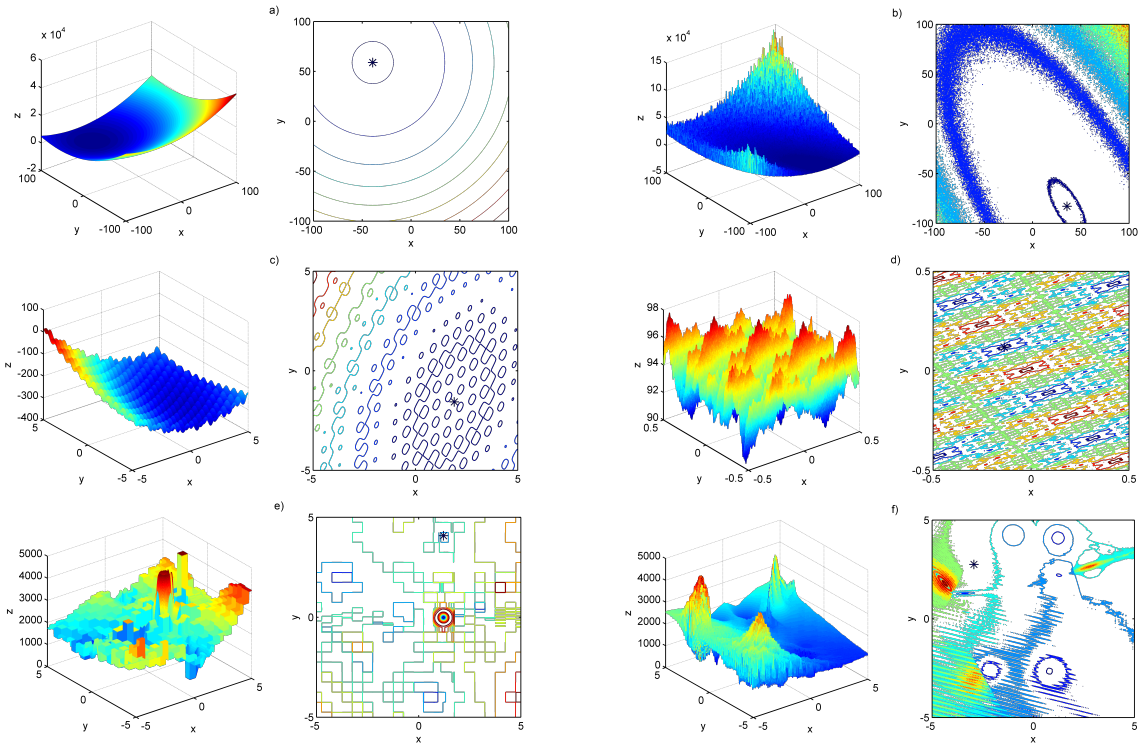
Benchmarki mierzą efektywność rozwiązywania wybranych problemów testowych. Z tego względu, ważne jest aby pamiętać, jakie cechy i regularności zostały w nich zawarte (Suganthan, 2005):

- Wysoki koszt obliczenia funkcji celu.
- Wysoki wymiar przestrzeni poszukiwań.
- Obecność liniowych i nieliniowych ograniczeń.
- Złe uwarunkowanie funkcji celu.
- Zaszumienie funkcji celu (niedokładność pomiaru jej wartości).
- Duża liczba optimów lokalnych (tysiące).
- Funkcje nieseparowalne liniowo.
- Położenie optimum globalnego na ograniczeniu.

[1] Benchmarki są zwykle dostępne w kilku językach programowania, np. zestaw BBOB'10 posiada interfejsy do języków C, Matlab i Java.

[2] Ostatnia właściwość wynika z obserwacji, że funkcja ciągła jest jednoznacznie określona przez swoje obcięcie do przeliczalnego podzbioru gęstego, np. zbioru liczb wymiernych $f : \mathbb{Q} \to \mathbb{R}$ (Kraszewski, 2007).

Niektóre z tych własności są widoczne na rysunku 1 przedstawiającym dwuwymiarowe warianty problemów testowych benchmarku CEC'05 (Suganthan *et al.*, 2005). Każdą funkcję wykreślono w obrębie swojego zbioru dopuszczalnego, zaś gwiazdką oznaczono położenie minimum globalnego. Większość problemów testowych występujących w benchmarkach CEC i BBOB jest od lat opisywana w literaturze przedmiotu. Podlegają one jednak przekształceniom takim jak translacje $f'(x) = f(x + c)$, dodawanie stałej $f'(x) = f(x) + c$ czy obroty. Zabieg ten ma promować algorytmy, które są niezmiennicze względem tego typu przekształceń. Dzięki temu rezultaty uzyskane dla pojedynczych problemów testowych można uogólniać na całe klasy problemów.



Rysunek 1: Dwuwymiarowe warianty wybranych problemów z benchmarku CEC'05: a) sfera, b) zaszumiona elipsoida, c) funkcja Rastrigina, d) funkcja Weierstrassa e), f) funkcje hybrydowe

Ocena algorytmu jest procedurą dwukrokową. Najpierw jego efektywność mierzona jest osobno dla każdego problemu testowego. Następnie, wyniki uzyskane dla wszystkich problemów są agregowane aby odzwierciedlić efektywność dla całego zestawu testowego. Kroki te zostały przedyskutowane w podrozdziałach 2 i 3.

Impulsem do rozwoju procedur benchmarkowania są konkursy algorytmów optymalizacyjnych organizowane podczas konferencji CEC (Congress on Evolutionary Computation) i GECCO (Genetic and Evolutionary Computation Conference). Benchmarki z rodziny CEC, przedstawione w tabeli 1, obejmują szerokie spektrum specjalizowanych problemów. Benchmarki wykorzystywane podczas konferencji GECCO nazywają się BBOB (Benchmarking Black-Box Optimizers) i obejmują problemy jednokryterialne oraz zaszumione. Benchmark BBOB zaimplementowany jest jako rozwinięta platforma zapewniająca wygodne przeprowadzanie eksperymentów oraz wizualizację wyników (Hansen *et al.*, 2009).

2 Efektywność dla pojedynczego problemu

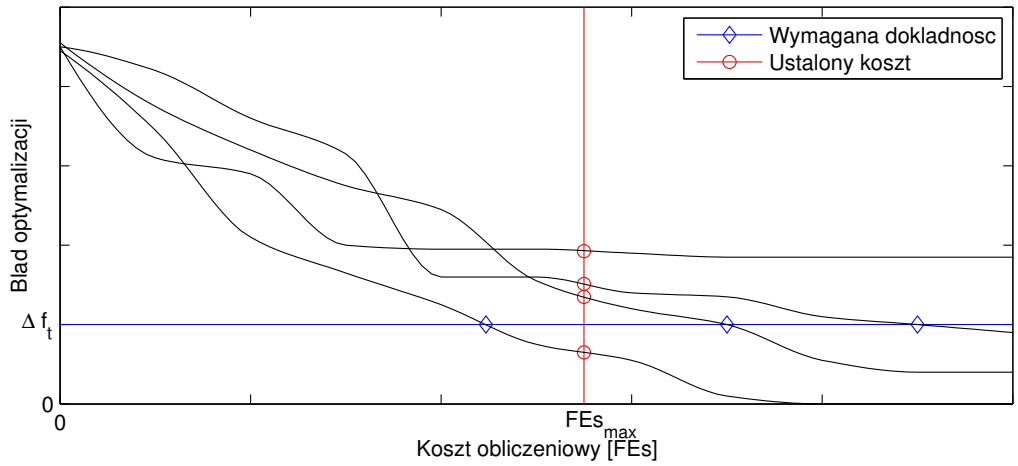
Oceny efektywności algorytmów optymalizacyjnych dla pojedynczego problemu testowego oparte są na dwóch metodach: ustalonego kosztu (ang. *fixed cost*) oraz wymaganej dokładności (ang. *fixed target*). W obu podejściach czas działania mierzony jest jako liczba obliczeń funkcji celu (FEs — ang *function evaluations*). Zabieg ten jest standardowym podejściem stosowanym w zakresie algorytmów optymalizacyjnych. Ma on na celu koncentrację na porównywaniu algorytmów a nie jakości ich implementacji, czy wykorzystanego sprzętu. Ponadto, błędy optymalizacji są niedeterministyczne, czyli różnią się dla poszczególnych uruchomień. Z tego powodu uzasadnione jest jedynie porównywanie wyników zagregowanych poprzez obliczanie takich statystyk jak średnia lub mediana.

Tabela 1: Benchmarki dla poszczególnych rodzajów problemów optymalizacyjnych

Rodzaj problemu	Benchmark
Jednokryterialne	CEC'05, BBOB'09, BBOB'10
Z ograniczeniami	CEC'06, CEC'10
Wielokryterialne	CEC'07, CEC'09
Wysokowymiaorowe	CEC'08, CEC'10
Dynamiczne	CEC'09
Zaszumione	BBOB'09, BBOB'10
Problemy praktyczne	CEC'11

2.1 Ustalony koszt

Metoda ustalonego kosztu polega na badaniu błędu optymalizacji $f_{best} - f_{opt}$ po ustalonej (maksymalnej) liczbie obliczeń funkcji celu. Na rysunku 2 przedstawiono krzywe zbieżności, czyli wykresy błędu optymalizacji w zależności od czasu obliczeń, dla czterech niezależnych uruchomień pewnej metody optymalizacyjnej. Pionowa linia obrazuje ustalony koszt obliczeń (Hansen *et al.*, 2009). W podejściu tym do porównań służą wielkości błędu powstające dla tej wartości kosztu: algorytm A oceniany jest jako lepszy niż algorytm B gdy daje niższe błędy. Porównania przeprowadzane są na skali porządkowej, która dostarcza jakościowej informacji o tym, który algorytm jest lepszy. Nie ma jednak możliwości ilościowego porównania ich efektywności (odpowiedzi na pytanie „jak duża jest różnica w efektywności?"), gdyż nie jest jasne o ile trudniej jest uzyskać mniejsze wartości błędów (Hansen *et al.*, 2009).



Rysunek 2: Kryteria ustalonego kosztu i wymaganej dokładności dla czterech przykładowych krzywych zbieżności (Hansen *et al.*, 2009)

2.2 Wymagana dokładność

Zamiast ustalania kosztu obliczeń i porównywania końcowych błędów optymalizacji można ustalić wymagany poziom dokładności Δf_t i porównywać koszty obliczeniowe konieczne do jego osiągnięcia. Dokładniej mówiąc, podejście wymaganej dokładności polega na szacowaniu oczekiwanego kosztu obliczeniowego jaki algorytm potrzebuje aby uzyskać wystarczająco dokładne rozwiązanie, to znaczy spełnić warunek $f_{best} - f_{opt} \leqslant \Delta f_t$. Kryterium to zostało zilustrowane na rys. 2 za pomocą poziomej linii. Oczekiwane czasy działania równych algorytmów można porównywać na skali interwałowej: możliwe jest stwierdzenie ile razy szybszy oraz o ile szybszy jest jeden algorytm od drugiego. Fakt ten ułatwia interpretację wyników porównań i przeważył o wyborze kryterium wymaganej dokładności w benchmarkach BBOB (Hansen *et al.*, 2009).

Podczas szacowania oczekiwanego czasu działania występują dwa problemy. Po pierwsze, wybór wymaganego poziomu dokładności Δf_t dokonywany jest w sposób uznaniowy. Problem ten można złagodzić poprzez ustalenie wielu (np. kilkudziesięciu) progów dokładności. Po drugie, występują trudności przy estymacji oczekiwanego czasu działania. Niektóre uruchomienia algorytmu mogą nie doprowadzić do znalezienia dostatecznie dokładnego

rozwiązania niezależnie od ilości poświęconego czasu. Dzieje się tak na przykład w przypadku przedwczesnej zbieżności, czyli sytuacji gdy algorytm zbiega do optimum lokalnego (Arabas, 2004). Kryterium zatrzymania obliczeń nie może być zatem oparte jednie na żądanej dokładności, ale musi również zawierać pewne „hamulce bezpieczeństwa", jak na przykład maksymalny koszt symulacji. Pojedyncze uruchomienie algorytmu nazywa się udanym wówczas, gdy wymagana dokładność zostaje osiągnięta w zadanym czasie. Bez takiego ograniczenia liczba udanych uruchomień przedstawionych na rysunku 2 wynosiłaby trzy. Obecność dodatkowego kryterium zatrzymania zmniejsza tę liczbę do jedynki, gdyż trzy uruchomienia kończą się na skutek przekroczenia czasu obliczeń, a nie znalezienia dostatecznie dokładnego rozwiązania.

Oczekiwany czas działania $\hat{E}(RT(\Delta f_t))$ dla zadanej dokładności Δf_t estymuje się jako sumę liczby obliczeń funkcji celu przypadających na jedno udane uruchomienie $\hat{E}(N^s_{eval})$ oraz iloczynu maksymalnego kosztu $\hat{E}(N^u_{eval})$ i szansy uzyskania nieudanego uruchomienia $(1-\hat{p})/\hat{p}$. Zmienna $\hat{p}$ oznacza oszacowanie prawdopodobieństwa rozwiązania problemu dla pojedynczego uruchomienia, czyli procentowy udział liczby udanych uruchomień (Hansen *et al.*, 2009).

$$\hat{E}\left(RT(\Delta f_t)\right) = \hat{E}(N^s_{eval}) + \frac{1-\hat{p}_s}{\hat{p}_s}\hat{E}(N^u_{eval}) \tag{2}$$

Oznaczając liczbę udanych uruchomień przez $\#s$ zaś nieudanych przez $\#u$ otrzymuje się naturalne oszacowanie $\hat{p} = \#s/(\#s + \#u)$. Równanie (2) może zostać przekształcone do postaci:

$$\hat{E}\left(RT(\Delta f_t)\right) = \frac{\#s \cdot \hat{E}(N^s_{eval}) + \#u \cdot \hat{E}(N^u_{eval})}{\#s} = \frac{N^t}{\#s} \tag{3}$$

przy czym N^t oznacza całkowitą liczbę obliczeń funkcji celu (FEs) dla wszystkich uruchomień algorytmu. Ponadto, przy wyznaczaniu wartości $\hat{E}(N^s_{eval})$ brane pod uwagę są tylko te obliczenia funkcji celu, które nastąpiły przed osiągnięciem wymaganego progu dokładności. Estymator (3) jest silnie zależny od doboru maksymalnego kosztu obliczeń (Hansen *et al.*, 2009). Gdyby pionową linię kosztu z rysunku 2 przesuwać na prawo, liczba udanych uruchomień $\#s$ wzrosłaby do dwóch a następnie do trzech skokowo zmieniając wielkość (3).

3 Efektywność dla benchmarku

3.1 Rankingi algorytmów

Porównywanie algorytmów optymalizacyjnych stanowi zagadnienie wielokryterialne, gdyż jest ono oparte na pewnej mierze efektywności P_{F_i} dla każdego z (kilkudziesięciu) problemów testowych F_1, F_2, ..., F_k. Algorytmy A i B mogą zostać naturalnie porównane za pomocą porządku Pareto

$$A \preceq B \equiv (\forall i \in \{1, 2, ..., k\})\ \ P^A_{F_i} \geqslant P^B_{F_i} \tag{4}$$

Algorytm A uznawany jest za lepszy niż algorytm B pod warunkiem, że daje lepsze wyniki dla każdego problemu testowego. Podejście to określa jednak jedynie porządek częściowy, gdyż niektóre pary algorytmów są nieporównywalne, np. wówczas gdy pierwszy uzyskuje lepsze wyniki dla niektórych problemów, ale gorsze dla innych. Z tego względu często trudno jest znaleźć odpowiedź na naturalne pytanie „która metoda optymalizacyjna jest najefektywniejsza?". Pojawia się więc potrzeba agregacji wyników dla poszczególnych problemów w jedną wartość, która pozwoli na utworzenie rankingu porównywanych algorytmów, czyli wprowadzenie porządku liniowego. Aby zapewnić obiektywny charakter porównań, metoda agregacji powinna zostać starannie dobrana.

Opracowywanie reguł decyzyjnych pozwalających na wybór najlepszego spośród dostępnych rozwiązań od lat stanowi obszar badań matematyków i ekonomistów zajmujących się teorią podejmowania decyzji i teorią sektora publicznego. Niektóre rezultaty z tych dziedzin można zastosować do analizy procedur tworzenia rankingów algorytmów optymalizacyjnych. Ważną rolę odgrywa tutaj słaby aksjomat ujawnionych preferencji (WARP — Weak Axiom of Revealed Preferences) wprowadzony w klasycznej pracy (Sen, 1971). Aksjomat ten można zdefiniować jako równoczesne spełnienie następujących warunków zwanych jako własności α i β:

- α — najefektywniejszy algorytm z pewnego zbioru algorytmów jest też najefektywniejszy z każdego jego podzbioru (jeżeli występuje w tym podzbiorze),
- β — jeżeli dwa algorytmy są jednocześnie najefektywniejsze w pewnym zbiorze, wówczas w każdym nadzbiorze albo oba będą najefektywniejsze albo żaden z nich.

W przypadku niespełnienia warunków WARP na wynik porównania można wpłynąć poprzez tworzenie krótkich list, porównywania parami albo wprowadzanie dodatkowych algorytmów zwiększających względne przewagi.

Tabela 2: Wpływ niezwiązanej alternatywy C na preferencje pomiędzy algorytmami A i B

$\{A, B\}$		Problem testowy				Średnia ranga
		F_1	F_2	F_3	F_4	
Ranga	1	A	A	B	B	$A = 1.5$
	2	B	B	A	A	$B = 1.5$

$\{A, B, C\}$		Problem testowy				Średnia ranga
		F_1	F_2	F_3	F_4	
Ranga	1	A	A	B	B	$A = 1.5$
	2	C	C	A	A	$B = 2.0$
	3	B	B	C	C	$C = 2.5$

Tabela 3: Oczekiwane czasy działania dla trzech problemów testowych i dwóch poziomów dokładności

		Problem testowy		
		F_1	F_2	F_3
Błąd	Δf_1	$3 \cdot 10^2$	$1 \cdot 10^3$	$4 \cdot 10^4$
	Δf_2	$2 \cdot 10^3$	$5 \cdot 10^3$	$7 \cdot 10^4$

Obliczanie efektywności dla benchmarku oparte jest na agregacji wyników uzyskanych przez algorytm dla poszczególnych problemów testowych. Niektóre z często używanych metod agregacji nie spełniają jednak własności β, co czyni je podatnymi na manipulację. Zagadnienie to można przedstawić za pomocą następującego przykładu. Przypuśćmy, że dwa algorytmy A i B zostały przetestowane dla czterech problemów F_1, F_2, F_3 i F_4, zaś ich wyniki zagregowano poprzez wyznaczenie średniej rangi (kolejności w rankingu) dla wszystkich problemów. Rangi błędów optymalizacji przedstawiono w górnej części tabeli 2. Średnia ranga obu algorytmów wynosi 1.5, zatem algorytmy A i B okazały się być równie efektywne. Przypuśćmy następnie, że do porównania dołączono dodatkowy algorytm C nieco słabszy niż A. Dla każdego problemu testowego algorytm C zajmuje więc pozycję o jedno miejsce słabszą niż A, co przedstawiono w dolnej części tabeli 2. Średnia ranga algorytmu A pozostała na poziomie 1.5, podczas gdy algorytm B wydaje się być teraz słabszy, gdyż jego średnia ranga wynosi 2. Zatem wprowadzenie niezwiązanej alternatywy C zmieniło preferencje pomiędzy algorytmami A i B na korzyść A. Przykład ten pokazuje, że porównania, w których średnia ranga wykorzystywana jest jako kryterium agregacji, są podatne na manipulacje. Problem ten występuje dla wielu naturalnych miar efektywności opartych na rangach. Z drugiej strony, warunki WARP są zawsze spełnione, gdy miara efektywności algorytmu nie zależy od innych, współzawodniczących algorytmów.

3.2 Dystrybuanta czasu działania

Efektywność algorytmu dla danego benchmarku można szacować poprzez mierzenie czasu działania dla wszystkich problemów testowych. Uzyskane w ten sposób rezultaty można wizualizować za pomocą wykresu empirycznej dystrybuanty oczekiwanego czasu działania, czyli procentowego udziału rozwiązanych problemów w zależności od oczekiwanego czasu działania. Ogólny sposób tworzenia takiego wykresu najwygodniej jest przedstawić za pomocą przykładu. Rozważmy analizę algorytmu na podstawie trzech funkcji testowych. W tabeli 3 przedstawiono oczekiwane czasy działania wyznaczone dla wielu niezależnych uruchomień oraz wymaganych dokładności Δf_1 i Δf_2. Kropkowana linia na rysunku 3 przedstawia rozkład czasu działania do uzyskania dokładności Δf_1. Linia ta pozwala odpowiedzieć na pytanie, dla jakiej części problemów można spodziewać się rozwiązania dysponując ustalonym budżetem FEs dla każdego problemu? Poniżej 300 FEs nie można rozwiązać żadnego z nich, pomiędzy 300 a 1000 tylko jeden (co stanowi 33% problemów), od 1000 do 40000 dwa (66%) zaś powyżej 40000 wszystkie trzy (100%). Kreskowana linia przedstawia analogiczne dane dla bardziej wymagającego kryterium dokładności Δf_2. Obie linie mogą zostać zagregowane poprzez potraktowanie każdej z sześciu par $(F, \Delta f) \in \{F_1, F_2, F_3\} \times \{\Delta f_1, \Delta f_2\}$ jako niezależnych problemów. Zastosowanie omówionej powyżej metody tworzenia wykresu prowadzi do wykreślenia nowej linii zaznaczonej na rysunku 3 czarnym kolorem. Zaletą tego typu agregacji jest częściowe rozwiązanie problemu uznaniowego doboru poziomu dokładności Δf.

W praktyce, czasy działania dla różnych progów dokładności bywają mierzone na podstawie tych samych uruchomień algorytmu (Hansen *et al.*, 2009). W związku z tym, czasy działania dla progów Δf_1 i Δf_2 są zależne, przez co przedstawiony sposób agregacji budzi wątpliwości natury statystycznej.

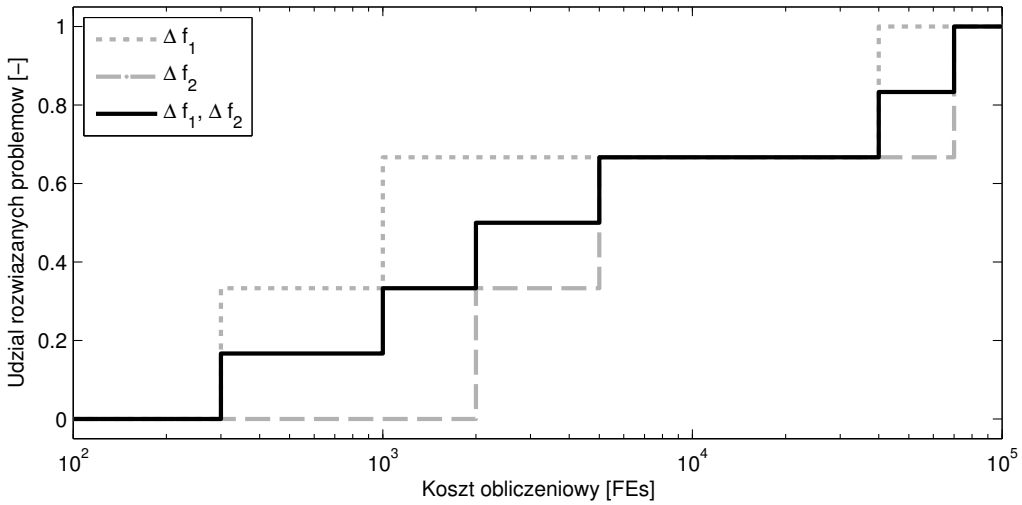
4 Benchmarki CEC i BBOB

W praktyce wykorzystywane są różnorodne wersje agregacji opartej na rangach dla poszczególnych problemów. Podczas konkursu CEC'06 ranking oparto na trzech statystykach efektywności: udziale rozwiązań dopuszczalnych, udziale uruchomień zakończonych sukcesem i oczekiwanym czasie działania (Liang and Suganthan, 2006). Podczas CEC'07 rangi nadawano na podstawie dwóch miar jakości dla problemów wielokryterialnych: wskaźnika R i różnicy (hiper)objętości ze zbiorem referencyjnym, co dało dwa podobne do siebie rankingi (Suganthan, 2007). Podczas CEC'08 i '09 każdy zespół oceniał szeregował algorytmy wszystkich pozostałych zespołów, a rangi zostały uśrednione (Tang, 2008; Zhang and Suganthan, 2009). Konkurs CEC'10 wygrał algorytm, który zdobył najwięcej punktów przyznawanych za każdy z 300 podproblemów optymalizacyjnych zgodnie ze schematem znanym z wyścigów Formuły 1 (Tang *et al.*, 2010). Podczas konferencji CEC'11 zwycięzcę wyłoniono poprzez obliczenie średniej rangi dla najmniejszej i średniej wielkości błędu optymalizacji (Suganthan, 2011).

W rodzinie benchmarków BBOB (Hansen *et al.*, 2009) nie tworzy się rankingów, gdyż agregacja wyników oparta jest o szacowanie dystrybuanty czasu działania. Na rysunku 4 przedstawiono wyniki konkursu BBOB'09 (Hansen *et al.*, 2010). Na poziomej osi odłożono czas działania mierzony jako iloraz liczby obliczeń funkcji celu i wymiaru przestrzeni poszukiwań n, który w tym przypadku równy jest 10. Oś pozioma przedstawia procentowy udział rozwiązanych problemów dla poziomów dokładności $\Delta f_t \in \{10^{1.8}, 10^{1.6}, 10^{1.4}, ..., 10^{-8}\}$. Rysunek 4 był stworzony w bardziej skomplikowany sposób niż rys. 3. Zamiast estymacji oczekiwanego czasu działania dla każdej pary $(F, \Delta f)$ losowano ze zwracaniem któreś z 15 uruchomień algorytmu aż do wylosowania uruchomienia zakończonego sukcesem. Łączną liczbę obliczeń funkcji celu do czasu osiągnięcia założonej dokładności przyjmowano jako czas działania algorytmu (Hansen *et al.*, 2010). Procedurę tę powtarzano 100 razy, dzięki czemu uzyskane wykresy dystrybuanty czasu działania (rys. 3) mają mniej schodkowy, a bardziej gładki kształt.

Podczas konkursu BBOB'09 każdy algorytm był wielokrotnie restartowany. Z jednej strony wspomaga to globalny charakter poszukiwań, z drugiej zaś łagodzi problem doboru „hamulców bezpieczeństwa" przy określaniu kryteriów zatrzymania, gdyż każdy zespół mógł sam zdecydować, kiedy zrestartować swój algorytm. Na rysunku 4 krzyżyki obrazują największą liczbę obliczeń funkcji celu dokonanych przez poszczególne algorytmy w jednym uruchomieniu. W opracowaniu (Hansen *et al.*, 2010) zasugerowano, że wypłaszczenie linii zaraz po krzyżyku (np. dla IPOP-SEP-CMA-ES) może wskazywać, że maksymalna liczba obliczeń funkcji celu powinna zostać zwiększona, podczas gdy stromy wzrost linii (np. dla simple GA) może oznaczać, że algorytm powinien być zrestartowany wcześniej.

Wykresy dystrybuanty czasu działania algorytmów dają duże możliwości interpretacyjne. Zilustrowano je na rysunku 5, na którym porównano dwa algorytmy: VNS i ALPS-GA. Dysponując budżetem $10^5 \cdot n$ FEs, gdzie n oznacza wymiar przestrzeni poszukiwań, algorytm ALPS-GA rozwiązuje około 60% problemów, podczas gdy VNS ponad 70%, co oznaczono pionowymi strzałkami. Różnica, a także iloraz tych wartości pokazują o ile oraz ile razy VNS jest skuteczniejszy niż ALPS-GA. Poziome strzałki określają koszt obliczeniowy jaki należy ponieść aby rozwiązać zadaną część problemów (tutaj 40%). Dla algorytmu VNS wynosi on około $400 \cdot n$ zaś dla ALPS-GA $10000 \cdot n$. Można zatem stwierdzić, że VNS jest w stanie 25 razy szybciej rozwiązać 40% problemów niż ALPS-GA.



Rysunek 3: Tworzenie zagregowanego wykresu dystrybuanty czasu działania

Autorzy benchmarku (Hansen *et al.*, 2010) sugerują, że pole pod wykresem (uśrednione dla skali logarytmicznej) jest jedną z najbardziej użytecznych zagregowanych miar efektywności. Wydaje się zatem, że taka wielkość może stanowić dobre kryterium doboru parametrów metody, na przykład za pomocą metaoptymalizacji.

5 Opracowanie wyników

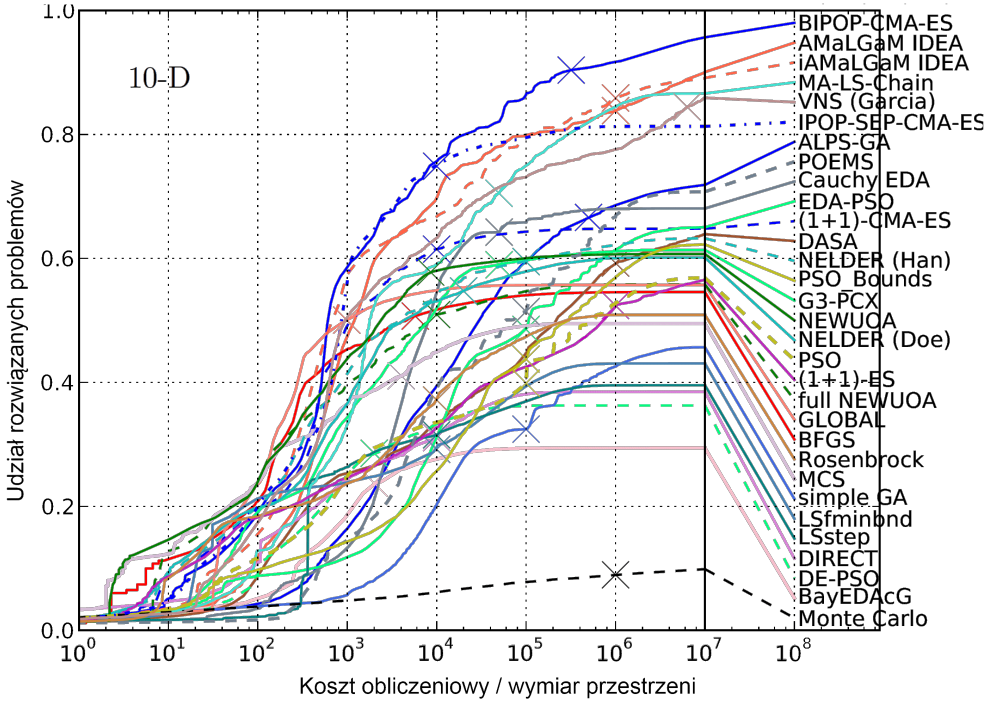
5.1 Analiza statystyczna

Statystyczna analiza wyników ma na celu sprawdzenie, czy obserwowane różnice w efektywności są istotne, czy też wynikają jedynie ze zbiegu okoliczności. W artykułach porównujących algorytmy optymalizacyjne często wykorzystuje się w tym celu test t-Studenta. Jest to test parametryczny, oparty na założeniu, że próbka (w tym wypadku tworzą ją końcowe wielkości błędów z wielu uruchomień) ma rozkład normalny. Alternatywnym rozwiązaniem jest oparcie się na testach nieparametrycznych, które można wykorzystywać dla próbek o dowolnych rozkładach. Analiza wyników uzyskanych w konkursie CEC'05 (García *et al.*, 2009) pokazała, że założenie o normalności rozkładu błędów zazwyczaj nie jest spełnione. Mimo tego, rezultaty analizy parametrycznej i nieparametrycznej okazały się być zgodne.

Oprócz porównywania wyników dla poszczególnych problemów testowych interesująca jest również ich agregacja w obrębie całych benchmarków. W tym przypadku zaleca się analizę nieparametryczną. W pracy (García *et al.*, 2009) zasugerowano wykorzystanie następujących testów: Friedmana, Imana-Davenporta, Bonferroniego-Dunna, Holma, Hochberga i Wilcoxona. Należy pamiętać, że w przypadku przeprowadzania serii porównań parami wzrasta prawdopodobieństwo popełnienia błędu I rodzaju, który oznacza uznanie przypadkowych różnic w efektywności algorytmów za statystycznie istotne. Problem ten rozwiązuje się stosując warianty testów przystosowane do porównań wielokrotnych. Można je skonstruować poprzez odpowiednie obniżenie poziomu istotności *alpha* poniżej typowej wartości 0.05.

Na rezultaty porównań algorytmów warto patrzeć nie tylko przez pryzmat testów statystycznych, ale również z szerszej perspektywy. Przykładowo, końcowe błędy optymalizacji rzędu $10^{-40} \pm 10^{-41}$ są istotnie mniejsze niż błędy rzędu $10^{-15} \pm 10^{-16}$. Rezultat ten nie ma jednak żadnego praktycznego znaczenia jeżeli został uzyskany np. dla funkcji kwadratowej i liczb w podwójnej precyzji[3]. Bezrefleksyjne stosowanie testów statystycznych

[3]Sytuację tę dobrze podsumowuje angielskie przysłowie: *difference is a difference only when it makes a difference*



Rysunek 4: Empiryczne dystrybuanty czasu działania dla 31 algorytmów zmierzonych na dziesięciowymiarowej wersji benchmarku BBOB'09 (Hansen *et al.*, 2010)

Tabela 4: Pełna tabela wyników

		Funkcja testowa				
		F1	F2	F3	F4	F5
$\mathrm{FEs} = 10^5$	$q_{0.00}$ (minimum)	5.7e−14	1.1e−02	3.0e+05	1.5e+00	...
	$q_{0.25}$	5.7e−14	1.8e−02	5.8e+05	5.1e+00	...
	$q_{0.50}$ (mediana)	5.7e−14	3.7e−02	7.8e+05	7.9e+00	...
	$q_{0.75}$	5.7e−14	6.5e−02	9.8e+05	1.2e+01	...
	$q_{1.00}$ (maksimum)	5.7e−14	1.1e−01	1.2e+06	2.2e+01	...
	średnia	5.7e−14	1.5e−01	9.8e+05	2.0e+01	...
	odch. std.	0.0e+00	2.3e−01	4.3e+05	3.1e+01	...

Tabela 5: Porównanie kilku algorytmów

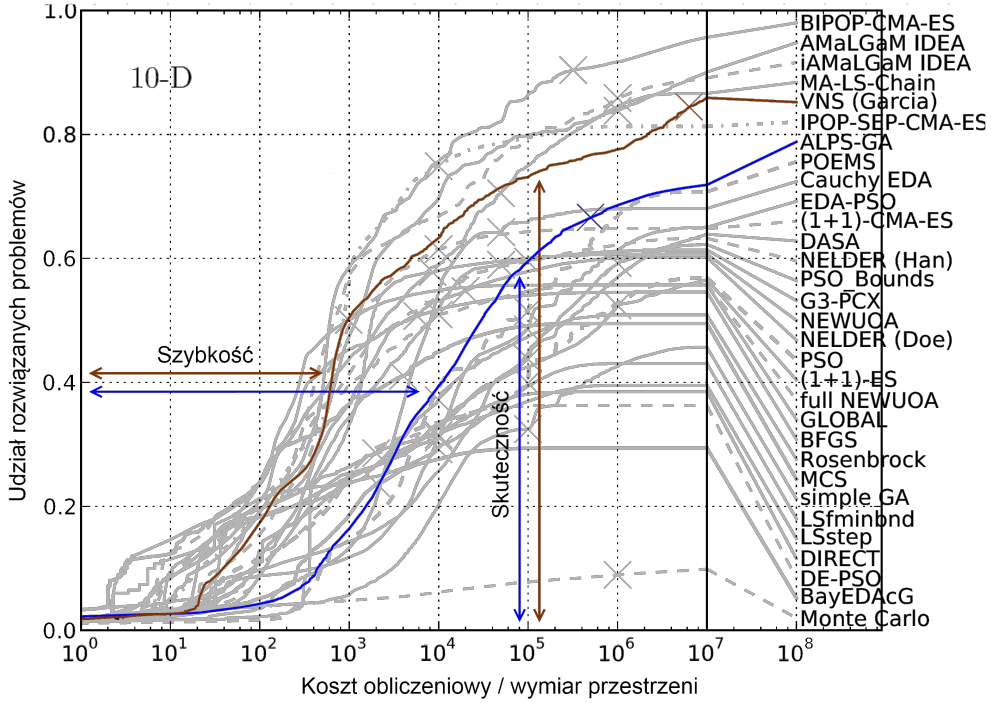
	`Alg. 1`	`Alg. 2`	`Alg. 3`
F1 (sphere)	**3.7e−15**	2.1e−08	5.0e+01
F2 (ellipsoid)	**5.0e−14**	4.6e−08	5.8e+01
F3 (Rastrigin)	**4.4e−06**	**4.7e−06**	8.7e−03
F4 (Rosenbrock)	5.3e+02	**6.5e−02**	9.8e+00
...	...	...	...

czasami wręcz zaciemnia prawdziwy obraz sytuacji zamiast go rozjaśniać.

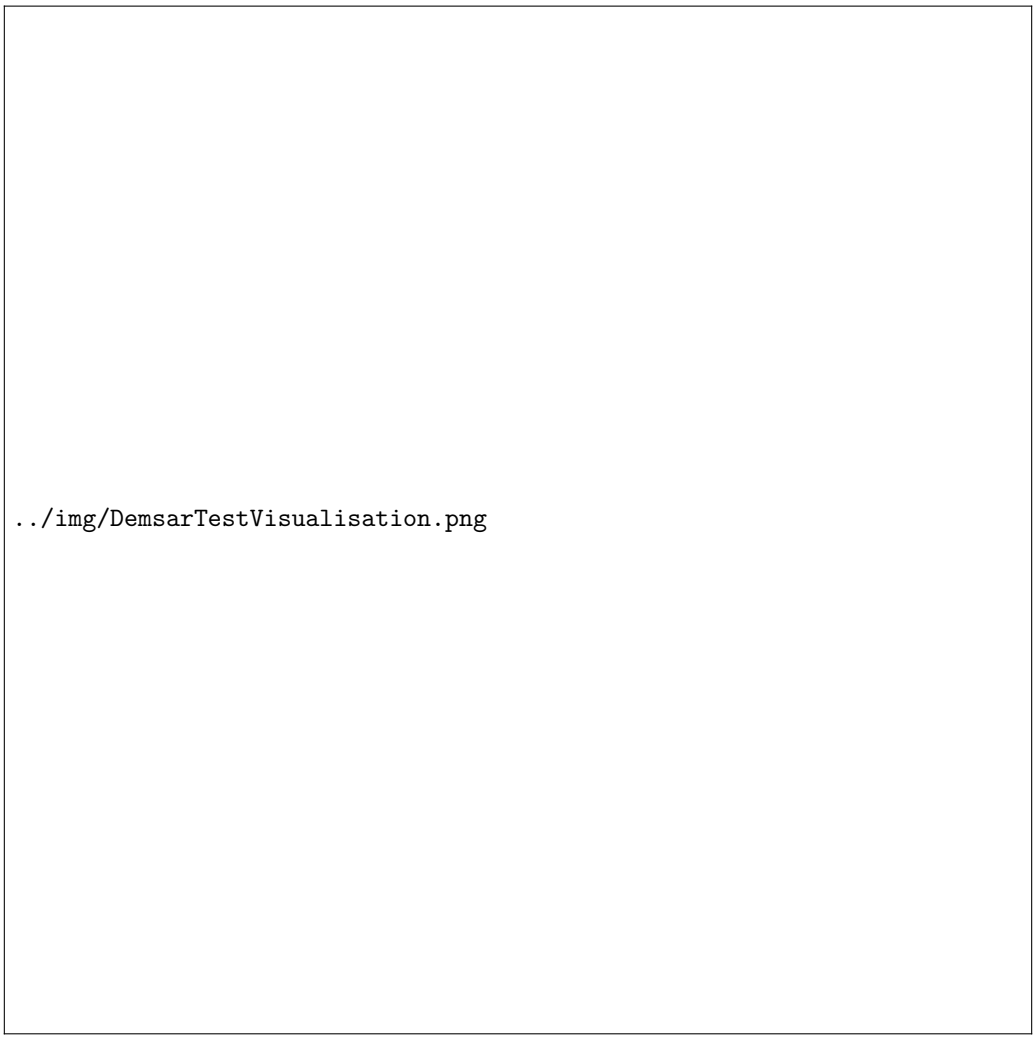
Demšar **?** ML tests!!!!!!!!!!!!

5.2 Raportowanie wyników

NAPISAĆ O TABELACH



Rysunek 5: Porównanie algorytmów VNS i ALPS-GA w zakresie skuteczności (pionowa odległość między krzywymi) i szybkości (pozioma odległość)



Rysunek 6: Graficzne przedstawienie statystycznej istotności różnic pomiędzy rangami dla czterech algorytmów, na przykładzie drzew decyzyjnych (**?**)

6 Zagadnienia badawcze

Procedury analiz porównawczych algorytmów optymalizacyjnych stanowią stosunkowo młodą dziedzinę w obrębie badań nad optymalizacją globalną. Poniżej wymieniono i krótko opisano subiektywnie wybraną listę potencjalnych kierunków rozwoju oraz wyzwań stojących przed procedurami benchmarkowania.

Dobór problemów testowych. Obecnie duża część problemów testowych cechuje się regularnościami takimi jak obecność globalnej struktury przypominającej wielowymiarową funkcję kwadratową, czy położeniem minimów lokalnych na (losowo obróconej i rozciągniętej) siatce. Wybór postaci i liczby funkcji testowych stanowi jednak nadal otwartą kwestię. Idealny problem testowy powinien być dobrze umotywowany (posiadać charakterystyczne własności, które ma on testować), odmienny od innych problemów występujących w benchmarku oraz selektywny (nie „za łatwy" ani „zbyt trudny"). Alternatywne podejście opiera się na tworzeniu benchmarku w oparciu o zestawy problemów zaczerpniętych z praktyki — zabiegu takiego dokonano podczas konkursu CEC'11 (Suganthan, 2011).

Rozwój procedur statystycznych. Wydaje się, że wciąż istnieje pole do rozwoju procedur analiz statystycznych wyników benchmarków zarówno od strony doboru odpowiednich metod, jak i ich automatyzacji.

Strojenie parametrów. Wynik algorytmu uzyskany dla benchmarku zależy także od doboru odpowiednich parametrów. Zagadnienie to ma raczej techniczny charakter i w związku z tym niektóre zespoły mogą nie poświęcać mu należytej uwagi. Tego typu sytuacja może skutkować zaniżeniem oceny wartościowych metod optymalizacyjnych. Problem ten można by złagodzić poprzez rozwinięcie benchmarku o moduł automatycznie strojący parametry odpowiednio do przyjętych kryteriów oceny, na przykład poprzez metaoptymalizację.

Benchmarki dla obliczeń równoległych. Obliczenia równoległe stanowią wyzwanie dla rozwoju wyspecjalizowanych metod optymalizacyjnych. Procedury analiz porównawczych powinny zostać dostosowane do specyfiki tego typu obliczeń, a zwłaszcza wykorzystania masowej równoległości zapewnianej przez karty graficzne stosowane w celach obliczeniowych. W pracy (Opara i Arabas, 2011) pokazano, że zmiana ta pociąga za sobą nie tylko skutki ilościowe ale również jakościowe. Ponadto, zaproponowano odpowiednie kryteria oceny.

7 Podsumowanie

W poniższej pracy starano się uwypuklić potrzebę i zalety systematycznego oraz starannego podejścia do kwestii porównywania algorytmów optymalizacyjnych. Procedury oceny ich efektywności przeanalizowano zarówno od strony teoretycznej jak i rozwiązań występujących w benchmarkach CEC i BBOB. Uwypuklono motywacje i intuicyjne przesłanki, które wpłynęły na ich strukturę. Przedstawiono również kryteria oceny i porównywania algorytmów oraz sposoby interpretacji uzyskiwanych wyników. Szczególną uwagę poświęcono tym elementom, które skutkują podatnością benchmarków na manipulacje. Artykuł zakończony jest listą potencjalnych kierunków rozwoju procedur porównawczych.

8 Podziękowania

Praca współfinansowana przez Unię Europejską w ramach Europejskiego Funduszu Społecznego.

Literatura

J. Arabas (2004), *Wykłady z algorytmów ewolucyjnych (Lecture notes in Evolutionary Algorithms)*, WNT, Warszawa, (in Polish).

T. English (2000), Optimization Is Easy and Learning Is Hard In the Typical Function, in *Proceedings of the 2000 Congress on Evolutionary Computation: CEC 2000*, pp. 924–931.

S. García, D. Molina, M. Lozano, and F. Herrera (2009), A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC 2005 Special Session on Real Parameter Optimization, *Journal of Heuristics*, 15(6):617–644.

N. Hansen, A. Auger, S. Finck, and R. Ros (2009), Real-Parameter Black-Box Optimization Benchmarking 2009: Experimental setup, Technical report, INRIA, URL `http://coco.gforge.inria.fr`, (access: July 2010).

N. Hansen, A. Auger, R. Ros, S. Finck, and P. Posik (2010), Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009, in *Proceedings of GECCO'10*, ACM, (access: July 2010).

Mario Köppen, David H. Wolpert, and William G. Macready (2001), Remarks on a recent paper on the "No Free Lunch" theorems, *IEEE Transactions on Evolutionary Computation*, 5(3):295–296, ISSN 1089-778X.

Jan Kraszewski (2007), *Wstęp do matematyki (Introduction to Mathematics)*, WNT, (in Polish).

J. J. Liang and P. N. Suganthan (2006), Comparison of Results on the 2006 CEC Benchmark Function Set, URL `http://www.ntu.edu.sg/home/epnsugan/`, (access: September 2011).

K. Opara j J. Arabas (2011), Benchmarking procedures for continuous optimization algorithms, *Journal of Telecommunications and Information Technology*, (to appear).

Amartya K. Sen (1971), Choice functions and revealed preference, *The Review of Economic Studies*, 38(3):307–317.

P. Suganthan (2005), Special Session on Real-Parameter Optimization at CEC 2005. Summary of feedbacks received from potential participants, URL `http://www.ntu.edu.sg/home/epnsugan/`, (access: May 2010).

P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari (2005), Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Technical report, Nanyang Technological University, Singapore and KanGAL Report Number 2005005, (access: June 2010).

P. N. Suganthan (2007), Performance Assessment on Multi-objective Optimization Algorithms, URL `http://www.ntu.edu.sg/home/epnsugan/`, (access: October 2011).

P. N. Suganthan (2011), Testing Evolutionary Algorithms on Real-World Numerical Optimization Problems, URL `http://www.ntu.edu.sg/home/epnsugan/`, (access: November 2011).

Ke Tang (2008), Summary of Results on CEC'08 Competition on Large Scale Global Optimization, URL `http://www.ntu.edu.sg/home/epnsugan/`, (access: September 2011).

Ke Tang, Thomas Weise, Zhenyu Yang, Xiaodong Li, and P. N. Suganthan (2010), Results of the Competition on High-dimensional Global Optimization at WCCI 2010, URL `http://www.ntu.edu.sg/home/epnsugan/`, (access: September 2011).

T. Weise, M. Zapf, R. Chiongand, and A. Nebro (2009), Why Is Optimization Difficult?, in R. Chiong, editor, *Nature-Inspired Algorithms for Optimisation*, volume 193 of *Studies in Computational Intelligence*, chapter 1, pp. 1–50, Springer, ISBN 978-3-642-00266-3, URL `http://www.springer.com/engineering/book/978-3-642-00266-3`.

D. Wolpert and W. Macready (1997), No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, URL `http://dx.doi.org/10.1109/4235.585893`.

Qingfu Zhang and Ponnuthurai N. Suganthan (2009), Final Report on CEC'09 MOEA Competition, URL `http://www.ntu.edu.sg/home/epnsugan/`, (access: September 2011).