Modyfikacja reguł adaptacji macierzy kowariancji oraz zasięgu mutacji w strategiach ewolucyjnych

Eryk Warchulski

Instytut Informatyki, Politechnika Warszawska
eryk.warchulski.stud@pw.edu.pl

Streszczenie. W niniejszym artykule podejmuję problem modyfikacji dotychczasowych reguł adaptacji macierzy kowariancji oraz zasięgu mutacji w metodach strategii ewolucyjnych. Proponuję trzy metody kontroli zasięgu mutacji. Każda z nich bazuje na wartości funkcji celu punktu środkowego oraz na uproszczonej regule adaptacji macierzy kowariancji. Na podstawie przeprowadzonych przeze mnie eksperymentów wynika, że osiągane przez nie wyniki na standaryzowanych problemach testowych zdefiniowanych w ramach konkursu `CEC'2017` sugerują konkurencyjność wobec obecnie stosowanych modyfikacji.

Słowa kluczowe: Strategie ewolucyjne, optymalizacja, CMA-ES, CSA

1 Wprowadzenie

Strategia ewolucyjna z adaptacją macierzy kowariancji (ang. *Covariance Matrix Adaptation Evolution Strategy*, `CMA-ES`) [1] znajduje się w czołówce metod optymalizacyjnych w ramach konkursów `IEEE CEC` [2] oraz `BBOB` [3].

W każdej iteracji algorytm tworzy populację złożoną z punktów wylosowanych przy użyciu wielowymiarowego rozkładu normalnego, którego parametry, tj. wektor wartości oczekiwanej oraz macierz kowariancji, są modyfikowane w każdej iteracji. Modyfikacja tych parametrów odbywa się na podstawie reguły CMA (ang. *Covariance Matrix Adaptation*). Wskutek działania powyższej reguły funkcja gęstości prawdopodobieństwa lokalnie aproksymuje funkcję celu i tym samym zwiększa szansę na wylosowanie punktów blisko optimum lokalnego. Dynamika adaptacji rozkładu normalnego sterowana jest również przez zmianę zasięgu mutacji, która bazuje na mechanizmie ścieżki ewolucyjnej (ang. *evolution path*) [4]. Zwiększona efektywność działania algorytmu wskutek stosowania powyższych mechanizmów jest okupiona znaczącym narzutem obliczeniowym, który często ogranicza zakres praktycznych zastosowań metody.

W artykule tym proponuję modyfikacje działania algorytmu `CMA-ES`, które pozwalają zredukować złożoność obliczeniową metody przy niewielkiej stracie efektywności działania. Proponowane przeze mnie modyfikacje dotyczą reguły adaptacji macierzy kowariancji oraz zasięgu mutacji.
Zmiana sposobu sterowania zasięgiem mutacji bazuje na kontroli punktu środkowego populacji [5], a zmiany dotyczące adaptacji macierzy kowariancji na pomyśle rozważanym przez Beyer'a [6].

Artykuł ten posiada następującą strukturę. W sekcji (2) krótko przedstawiam zasadę działania algorytmu `CMA-ES` oraz nakreślam problemy związane ze złożonością obliczeniową w jego kanonicznej wersji. Omawiam również dotychczasowe próby rozwiązania tego problemu. Dokładny opis proponowanych przeze mnie metod znajduje się w sekcji (3). Sekcja (4) zawiera komentarz do rezultatów osiągniętych w ramach eksperymentów numerycznych. Sekcja ostatnia, (5), stanowi podsumowanie artykułu oraz omawiam w niej plan dalszych badań.

2 Mechanizmy adaptacyjne

Pseudokod zawarty na poniższym wydruku przedstawia kanoniczną wersję algorytmu `CMA-ES`. Metoda utrzymuje trzy parametry: wektor oczekiwany $\mathbf{m}^t$, macierz kowariancji $\mathbf{C}^t$ oraz zasięg mutacji σ^t. Parametry te specyfikują wielowymiarowy rozkład normalny, który służy do tworzenia nowych punktów w kolejnych generacjach. Indeks górny t oznacza numer iteracji algorytmu. W każdej iteracji algorytm przy pomocy rozkładu normalnego z zerową wartością oczekiwaną oraz macierzą kowariancji $\mathbf{C}^t$ generuje zbiór wektorów $\{\mathbf{d}_1^t, ..., \mathbf{d}_\lambda^t\}$. Wektory te służą do zaburzenia obecnego w danej iteracji wektora wartości oczekiwanej $\mathbf{m}^t$ wskutek czego powstają wektory pochodne

$$\mathbf{x}_i^t = \mathbf{m}^t + \sigma^t \mathbf{d}_i^t. \tag{1}$$

Zbiór wektorów pochodnych jest sortowany malejąco (10). Ze zbioru wektorów bazowych o liczności λ wydzielany jest podzbiór o liczności μ wektorów o najmniejszej wartości funkcji celu. Podzbiór ten służy do aktualizacji parametrów algorytmu.
Wartość oczekiwana $\mathbf{m}^t$ rozkładu aktualizowana jest wskutek zsumowania jej ze średnią ważoną wyselekcjonowanych wektorów $\mathbf{d}_i^t$. Wagi w_i powinny być liczbami dodatnimi, których suma wynosi 1 [7].

Sposób aktualizacji macierzy kowariancji oraz parametru zasięgu mutacji ze względu na jego znaczenie opisałem kolejno w sekcji (2.1) oraz (2.2).

2.1 Adaptacja macierzy kowariancji

Macierz kowariancji $\mathbf{C}^{t+1}$ wyliczana jest jako suma złożona z dwóch składników. Pierwszy z nich, $\mathbf{C}_\mu^t$, jest macierzą rangi μ, która powstaje jako ważona suma iloczynu zewnętrznego μ wektorów $\mathbf{d}_i^t$. Drugi z nich, $\mathbf{C}_1^t$, jest macierzą o rzędzie równym 1 powstałą jako iloczyn zewnętrzny wektora $\mathbf{p}_c^t$. Wektor ten jest jedną ze ścieżek ewolucyjnych, które utrzymuje algorytm w ramach swojego działania.

Oba składniki mają na celu zwiększenie prawdopodobieństwa wylosowania wektorów w kierunku poprawy, tj. mniejszej wartości funkcji celu. Wkład macierzy $\mathbf{C}_\mu^t$ odpowiada za zysk, który przynosić ma selekcja osobników, a z kolei macierz $\mathbf{C}_1^t$ – odpowiada za zysk wnoszony przez ścieżkę ewolucyjną $\mathbf{p}_c^t$. Ścieżkę tę należy rozumieć jako trajektorię populacji zgodnie z którą kierunki tworzenia nowych punktów były najlepsze [4].

Listing Algorytm `CMA-ES`

$t \leftarrow 1$
$\mathbf{p}_c^1 \leftarrow \mathbf{0}$, $\mathbf{p}_\sigma^1 \leftarrow \mathbf{0}$
while !stop **do**
  for $i = 1 : \lambda$ **do**
    $\mathbf{d}_i^t \sim N(\mathbf{0}, \mathbf{C}^t)$
    $\mathbf{x}_i^t = \mathbf{m}^t + \sigma^t \mathbf{d}_i^t$
    oceń $(\mathbf{x}_i^t)$
  end for
  sort $\left(\{\mathbf{x}_i^t\}\right)$
  $\mathbf{\Delta}^t \leftarrow \sum_{i=1}^{\mu} w_i \mathbf{d}_i^t$
  $\mathbf{m}^{t+1} \leftarrow \mathbf{m}^{t+1} + \sigma^t \mathbf{\Delta}^t$
  $\mathbf{p}_c^{t+1} \leftarrow (1 - c_p)\mathbf{p}_c^t + \sqrt{\mu_{\text{eff}} c_p (2 - c_p)} \cdot \mathbf{\Delta}^t$ gdzie $\mu_{\text{eff}} = 1/\left(\sum_{i=1}^{\mu} (w_i)^2\right)$
  $\mathbf{C}^{t+1} \leftarrow (1 - c_1 - c_\mu)\mathbf{C}^t + c_1 \mathbf{C}_1^t + c_\mu \mathbf{C}_\mu^t$ gdzie $\mathbf{C}_\mu^t = \frac{1}{\mu_{\text{eff}}} \sum_{i=1}^{\mu} w_i (\mathbf{d}_i^t)(\mathbf{d}_i^t)^{\mathsf{T}}$, $\mathbf{C}_1^t = (\mathbf{p}_c^t)(\mathbf{p}_c^t)^{\mathsf{T}}$
  $\sigma^{t+1} \leftarrow \text{CSA}\ (\sigma^t, \mathbf{C}^t, \mathbf{\Delta}^t)$
  $t \leftarrow t + 1$
end while

2.2 Kumulatywna adaptacja zasięgu mutacji

Podobnie jak w przypadku adaptacji macierzy kowariancji strojenie zasięgu mutacji odbywa się przy pomocy ścieżki ewolucyjnej $\mathbf{p}_\sigma^t$.

Ścieżka ta kumuluje ona kierunki przesunięcia punktu środkowego, ale znormalizowanego przez odwrotny pierwiastek kwadratowy macierzy kowariancji w taki sposób, aby między kolejnymi wektorami nie było korelacji. Norma wektora $\mathbf{p}_\sigma^t$ porównywana jest z oczekiwaną długością N-wymiarowego wektora losowego z rozkładu $\mathcal{N}(0, \mathbf{I})$. Wynik porównania służy do dostosowania wartości zasięgu mutacji σ^{t+1} w następnej iteracji algorytmu. Jeśli stosunek norm przedstawiony w równaniu (2):

$$\frac{\|\mathbf{p}_\sigma^t\|}{E\|N(\mathbf{0}, \mathbf{I})\|} \tag{2}$$

jest większy od jedności, to zasięg mutacji powinien zostać zwiększony, a o algorytmie możemy powiedzieć, że jest w fazie *eksploracji* przestrzeni przeszukiwań. W przeciwnym razie zasięg powinien zostać zmniejszony – algorytm jest wówczas w fazie *eksploatacji*.

Listing 2 Reguła `CSA`

$\mathbf{p}_\sigma^{t+1} \leftarrow (1 - c_s)\mathbf{p}_\sigma^t + \sqrt{\mu_{\text{eff}} c_s (2 - c_s)} \cdot (\mathbf{C}^t)^{-\frac{1}{2}} \mathbf{\Delta}^t$
$\sigma^{t+1} \leftarrow \sigma^t \exp\left(\frac{c_s}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma^{t+1}\|}{E\|N(\mathbf{0}, \mathbf{I})\|} - 1\right)\right)$

2.3 Złożoność obliczeniowa

Obie opisane wyżej reguły przyczyniają się do znacznej poprawy działania algorytmu w kontekście ekstrapolacji i eksploatacji przestrzeni przeszukiwań. Niestety, ceną tego jest fakt znacznie zwiększonej złożoność obliczeniowej algorytmu. Wynika ona z faktu stosowania operacji wyliczania pierwiastka kwadratowego macierzy oraz odwrotności tego pierwiastka.

W kanonicznej wersji algorytmu operacja ta jest realizowana przez rozkład spektralny macierzy symetrycznej, tj. rozkład macierzy kowariancji $\mathbf{C}$ według poniższej formuły:

$$\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^T. \tag{3}$$

Umożliwia ona zarówno wyliczenie pierwiastka kwadratowego jak i odwrotność samej macierzy. Jednakże złożoność tych operacji jest rzędu $\mathcal{O}(n^3)$, co ogranicza stosowalność metody `CMA-ES` w przypadku zadań optymalizacyjnych o większej wymiarowości, tj. $N > 100$ [8].

Poczyniono szereg działań w kierunku zmniejszenia złożoności obliczeniowej algorytmu. Część badaczy skłania się do przedstawienia macierzy kowariancji jako sumy macierzy jednostkowej oraz iloczynu zewnętrznego (ang. *outer product*) gradientu naturalnego funkcji celu [9]. Podejmowano również próby eliminacji stosowania jakichkolwiek operacji macierzowych na rzecz wektorowego modelowania różnicy między macierzą kowariancji, a macierzą jednostkową [10]. Ograniczenia operacji macierzowych podjął się również autor algorytmu `CMA-ES`, zastępując regułę `CSA` regułą `MSR` (ang. Median Success Rule) [11].

Zauważam jednak pewne problemy z proponowanymi modyfikacjami. Redukcji złożoności obliczeniowej lub pamięciowej najczęściej towarzyszył znaczny spadek jakości działania względem oryginalnej wersji algorytmu. Tracona również była mniej wymierna własność algorytmu `CMA-ES`, tj. jego prostota koncepcyjna.

3 Modyfikacje reguł adaptacyjnych

Dokonane przeze mnie modyfikacje algorytmu dotyczyły postaci reguły adaptacji macierzy kowariancji oraz zasięgu mutacji.

W celu zmniejszenia narzutu obliczeniowego zastosowałem zmiany sugerowane w [6]. Autorzy, bazując na obserwacji, że klasycznie stosowana reguła adaptacji macierzy kowariancji daje się sprowadzić do ogólnej postaci:

$$\mathbf{A}\mathbf{A}^T = \mathbf{M}\left[\mathbf{I} + \mathbf{B}\right]\mathbf{M}^T \tag{4}$$

zastosowali rozwinięcie macierzy $\mathbf{A}$, która odpowiada macierzy $\mathbf{C}^{t+1}$ w kontekście logiki algorytmu, w szereg potęgowy:

$$\mathbf{A} = \mathbf{M}\sum_{i=0}^{\infty}\gamma_i\mathbf{B}^i. \tag{5}$$

Następnie, ograniczając szereg do wyrazu liniowego, uzyskali formułę poprawnie przybliżającą oryginalną regułę adaptacji. W ten sposób została wyeliminowana konieczność wyliczania pierwiastka kwadratowego macierzy kowariancji.

Jednakże sposób modyfikacji zasięgu mutacji σ^t nadal wymagał złożonych operacji macierzowych.

W ramach proponowanych zmian zasugerowałem użycie uogólnionej postaci reguły 1/5 [12], która nie stosuje jakichkolwiek operacji macierzowych. W pierwotnej wersji dotyczy ona wyłącznie strategii ewolucyjnych, w których populacja bazowa składa się z jednego punktu. Poniżej znajdują się moje propozycje dostosowania tej reguły do strategii ewolucyjnych, w których populacja bazowa wynosi $\lambda \geq 2$. W każdej z nich punkt środkowy $\tilde{\mathbf{x}}$ należy rozumieć jako średnią arytmetyczną z całej populacji, tj.

$$\tilde{\mathbf{x}} = \frac{1}{\lambda} \sum_{i=1}^{\lambda} \mathbf{x}_i. \tag{6}$$

Postać pierwszej, tj. reguły `PPMF` (ang. *Previous Population Midpoint Fitness*), z nich znajduje się na wydruku poniżej. Bazuje ona na wartości funkcji celu punktu środkowego z poprzedniej iteracji – $\tilde{\mathbf{x}}^{t-1}$. Prawdopodobieństwo sukcesu p_s w jej ramach definiowane jest jako liczba punktów w populacji, których wartość funkcji celu jest mniejsza od wartości funkcji celu punktu środkowego poprzedniej iteracji (linia 3).

Listing 3 Reguła `PPMF`

1: $\overline{\mathbf{x}}^{t-1} = \frac{1}{\lambda} \sum_{i=1}^{\lambda} \mathbf{x}_i^{t-1}$
2: oceń $(\overline{\mathbf{x}}^{t-1})$
3: $p_s = \left|\{i : q(\mathbf{x}_i^t) < q(\overline{\mathbf{x}}^{t-1})\}\right| / \lambda$
4: $\sigma^{t+1} \leftarrow \sigma^t \exp\left(\frac{1}{d} \cdot \frac{p_s - p_{\text{target}}}{1 - p_{\text{target}}}\right)$

Druga metoda, reguła `CPEF` (ang. Current Population Expectation Fitness), jest zaprezentowana na poniższym wydruku. Mechanizm działania reguły jest ten sam co dla reguły `PPMF` z tą różnicą, że zamiast punktu środkowego jako średniej arytmetycznej do wyliczenia parametru p_s służy wartość oczekiwana $\mathbf{m}^t$.

Listing 4 Reguła `CPEF`

1: oceń $(\mathbf{m}^t)$
2: $p_s = \left|\{i : q(\mathbf{x}_i^t) < q(\mathbf{m}^t)\}\right| / \lambda$
3: $\sigma^{t+1} \leftarrow \sigma^t \exp\left(\frac{1}{d} \cdot \frac{p_s - p_{\text{target}}}{1 - p_{\text{target}}}\right)$

Trzecia metoda, reguła `CPMF` (ang. *Current Population Midpoint Fitness*), oparta jest na obserwacji poczynionej w [5], zgodnie z którą punkt środkowy $\tilde{\mathbf{x}}^t$ najlepiej estymuje optimum lokalne spośród wszystkich punktów populacji. Jeśli

wartość funkcji celu $\tilde{\mathbf{x}}^t$ jest lepsza od wartości funkcji celu najlepszego punktu w populacji $\mathbf{x}^t_{\text{best}}$, to najprawdopodobniej znajduje się ona w bliskim sąsiedztwie optimum lokalnego. Wówczas zasięg mutacji powinien zostać zmniejszony, aby algorytm mógł z większą precyzją eksploatować dany obszar funkcji celu. W metodzie tej prawdopodobieństwo sukcesu jest porównywane z k-tym percentylem wartości funkcji celu punktów populacji.

Listing 5 Reguła `CPMF`

1: $\overline{\mathbf{x}}^t = \frac{1}{\lambda}\sum_{i=1}^{\lambda} \mathbf{x}_i^t$
2: oceń $(\overline{\mathbf{x}}^t)$
3: $p_s = \left|\{i : q(\mathbf{x}_i^t) < q(\overline{\mathbf{x}}^t)\}\right| / \lambda$
4: **if** $p_s < k$ **then**
5: $\quad \sigma^{t+1} \leftarrow \alpha \cdot \sigma^t$
6: **else**
7: $\quad \sigma^{t+1} \leftarrow \alpha^{-1} \cdot \sigma^t$
8: **end if**

Ze względu na wczesny etap prac nie ustaliłem jeszcze optymalnych parametrów metody. W ramach eksperymentów opisanych w sekcji (4) opracowane przeze mnie metody posiadają następujące wartości parametrów sterujących: $k = 0.09$, $p_{\text{target}} = 0.25$, $d = 8$. Wybór akurat tych wartości ma charakter arbitralny.

4 Eksperymenty numeryczne

W celu weryfikacji proponowanych zmian skorzystałem ze standaryzowanych zadań optymalizacyjnych stosowanych w ramach konkursu `IEEE CEC'17` [2]. Ze względu na fakt, że jest to jeden z najbardziej popularnych zbiorów problemów testowych, uzyskałem możliwość względnie obiektywnego porównania moich pomysłów z klasyczną wersją algorytmu `CMA-ES` oraz jego modyfikacjami.

Problemy w ramach zestawu zadań podzielone są na funkcje: unimodalne, multimodalne oraz kompozytowe. Problemy kompozytowe są złożeniami problemów multimodalnych, które dodatkowo podlegają pewnym przekształceniom jak obrót czy translacja. Sposób w jaki funkcje testowe są skonstruowane ma na celu sprawdzić różne własności algorytmu, a w szczególności: odporność na zaszumienie funkcji celu, niewrażliwość na przekształcenia afiniczne lub wymiar zadania.

Porównanie testowanych metod sprowadza się do porównania, która z nich uzyskała mniejszą wartość funkcji celu przy ustalonym budżecie wywołań tej funkcji. Organizatorzy konkursu sugerują, aby wyniki testów przedstawiane były w postaci tabelarycznej, ale ze względu na większą czytelność zdecydowałem się przedstawiać je za pomocą krzywych ECDF (ang. *Empirical Cumulative Distribution Function*). Krzywa ECDF konstruowana jest w taki sposób, że na osi odciętych odkładany jest odsetek posiadanego budżetu wywołań funkcji, a na

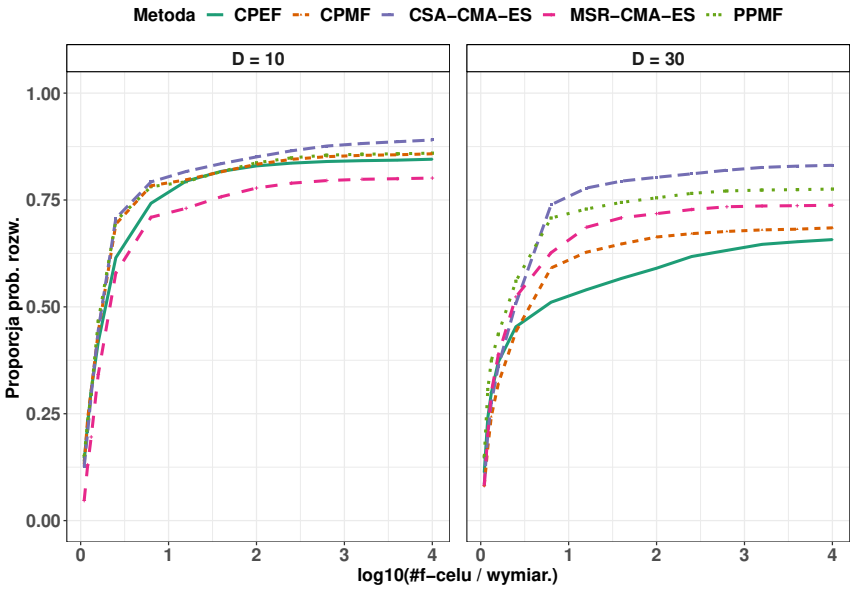
osi rzędnych – procent problemów rozwiązanych przy zadanym budżecie przez algorytm. Im większe jest pole pod krzywą danego algorytmu, tym lepszy wynik uzyskał on względem pozostałych metod.

4.1 Parametry eksperymentów

W ramach eksperymentu porównałem proponowane przeze mnie metody z algorytmem `CMA-ES` w klasycznej postaci (t.zw. `CSA-CMA-ES`) oraz z algorytmem `MSR-CMA-ES`. Wszystkie wspólne parametry metod jak rozmiar populacji λ są zgodne z zalecaniami przedstawionymi przez autora algorytmu w [13]. Specyficzne parametry algorytmu `MSR-CMA-ES` ustalono zgodnie z sugerowanymi wartościami w [11]. Algorytmy testowano dla zadań o wymiarowości $D = 10, 30$ przy budżecie wywołań funkcji celu równym $10^4 D$.

4.2 Wyniki na standaryzowanych problemach testowych

Na Rys. 1 przedstawione są uzyskane wyniki. Standardowa wersja algorytmu `CMA-ES` przewyższa wszystkie inne rozważane w artykule modyfikacje tej metody. Reguła `CPEF` charakteryzuje się najgorszą wydajnością. Ogólna wydajność `PPMF` jest porównywalna z `CPMF`, z niewielką przewagą `PPMF` nad `CPMF`. Obie metody dają lepsze wyniki niż `MSR-CMA-ES`.



Rys. 1: Krzywe ECDF uzyskane dla wszystkich problemów zdefiniowanych w konkursie `CEC'17`

5 Podsumowanie

Uzyskane wyniki na zbiorze testowym `CEC'17` są zadowalające. Na ich podstawie udało mi się wykazać, że możliwe jest uogólnienie reguły 1/5 dla algorytmu `CMA-ES` bez dużej straty w efektywności działania algorytmu. Ponadto fakt, że proponowane przeze mnie reguły działają lepiej niż dotychczas stosowane metody jak [11], jest zachęcający do dalszych badań.

W kontekście przeprowadzonych eksperymentów istotna jest również obserwacja, że algorytm z jednocześnie uproszczoną adaptacją macierzy kowariancji i zasięgu mutacji działa w sposób porównywalny do swojej kanonicznej wersji. Następnym krokiem badania jest optymalizacja parametrów strojenia opracowanych metod.

Literatura

1. Trautmann, H., Mersmann, O., Arnu, D.: cmaes: Covariance Matrix Adapting Evolutionary Strategy. (2011) R package version 1.0-11.
2. Awad, N.H., Ali, M., Liang, J., Qu, B., Suganthan, P.N.: Problem definitions and evaluation criteria for the CEC 2017 special session and competition on real-parameter optimization. Technical report, Nanyang Technol. Univ., Singapore and Jordan Univ. Sci. Technol. and Zhengzhou Univ., China (2016)
3. Hansen, N., et al.: Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation. GECCO '10, New York, USA, ACM (2010) 1689–1696
4. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2) (2001) 159–195
5. Arabas, J., Biedrzycki, R.: Improving evolutionary algorithms in a continuous domain by monitoring the population midpoint. IEEE Trans. Evol. Comput. **21**(5) (2017) 807–812
6. Beyer, H.G., Sendhoff, B.: Simplify your Covariance Matrix Adaptation Evolution Strategy. IEEE Trans. Evol. Comput. **21**(5) (2017) 746–759
7. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2) (June 2001) 159–195
8. Hansen, N.: The CMA evolution strategy: a comparing review. In Lozano, J.A., ed.: Towards a new evolutionary computation: advances on estimation of distribution algorithms. Springer (2006) 75–102
9. Poland, J., Zell, A.: Main vector adaptation: A cma variant with linear time and space complexity. In: Proc. Genet. Evol. Comput. Conf., Morgan Kaufmann (2001) 1050–1055
10. Loshchilov, I.: LM-CMA: An alternative to L-BFGS for large-scale black box optimization. Evol. Comput. **25**(1) (2017) 143–171
11. ElHara, O.A., Auger, A., Hansen, N.: A median success rule for non-elitist evolution strategies: study of feasibility. In Blum, C., Alba, E., eds.: Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013, ACM (2013) 415–422
12. Schwefel, H.: Evolution and optimum seeking. Sixth-generation computer technology series. Wiley (1995)
13. Hansen, N.: The CMA Evolution Strategy: A Tutorial. (2016)