



POLITECHNIKA WARSZAWSKA

WYDZIAŁ MATEMATYKI
I NAUK INFORMACYJNYCH



PRACA DYPLOMOWA MAGISTERSKA

**ANALIZA MOŻLIWOŚCI WYKORZYSTANIA
W ALGORYTMIE CMA-ES WIEDZY
O OGRANICZENIACH KOSTKOWYCH**

AUTOR:

INŻ. ROBERT JAKUBOWSKI

PROMOTOR:

DR HAB. INŻ. JAROSŁAW ARABAS
PROF. NZW. PW

WARSZAWA MAJ 2016

.....

podpis promotora

.....

podpis autora

Spis treści

1	Streszczenie	5
2	Wstęp	6
3	Algorytm CMA-ES	8
3.1	Macierz kowariancji	9
3.2	Wartość oczekiwana rozkładu	10
3.3	Długość kroku	10
3.4	Generowanie punktów	10
3.5	Selekcja punktów	10
3.6	Ocena	11
4	Techniki uwzględniania ograniczeń	12
4.1	Transformacje rozwiązań	12
4.1.1	Metoda konserwatywna	13
4.1.2	Rzutowanie	13
4.1.3	Reinicjacja	13
4.1.4	Powtórna generacja	13
4.1.5	Zawijanie	13
4.1.6	Odbicie	14
4.2	Błądzenie przypadkowe	14
4.3	Metoda przeprowadzania testów	15
4.3.1	Wartość oczekiwana generowanego punktu	15
4.3.2	Rozkład prawdopodobieństwa generowanych punktów	16
4.4	Wyniki testów	16
4.5	Wnioski	27
5	Metodyka testowania metod optymalizacji globalnej	28
5.1	Funkcje benchmarkowe	28
5.2	Techniki porównywania wyników	28
6	Weryfikacja wpływu technik uwzględniania ograniczeń na efektywność CMA-ES	30
6.1	Cel testów	30
6.2	Metoda przeprowadzenia testów	30

6.3	Wyniki testów	31
6.4	Wnioski	33
6.5	Porównanie wariantów transformacji rozwiązań	33
7	Podsumowanie	37
7.1	Wnioski	37
7.2	Możliwości rozwoju	37

1. Streszczenie

Ta praca bada wpływ różnych metod uwzględniania ograniczeń w algorytmie CMA-ES. Wyniki badań przeanalizowane są w celu sprawdzenia, czy można wykorzystać informacje o ograniczeniach kostkowych w tymże algorytmie.

We wstępnie został przedstawiony przedmiot badań i cel pracy. Ten rozdział zawiera kontekst badań oraz wprowadzenie do nich. Cały rozdział można potraktować jako motywację do wykonania tej pracy.

Po wstępnie opisano algorytm CMA-ES. Teoretyczna analiza tego algorytmu jest niezbędna do zrozumienia jego mechanizmów, a w konsekwencji do przeprowadzenia badań. Z tego samego powodu opisano również techniki uwzględniania ograniczeń. W rozdziale poświęconym ograniczeniom zawarto nie tylko opisy poszczególnych metod, ale również pokazano w sposób empiryczny jak mogą wpływać na algorytmy. W tym celu użyto algorytmu błędzenia przypadkowego. Zaimplementowano ten algorytm z różnymi wariantami uwzględniania ograniczeń, a następnie przeprowadzono szereg testów badających między innymi rozkłady prawdopodobieństw oraz wartości oczekiwanych.

Następny rozdział opisuje sposoby, w jaki mogą być testowane algorytmy optymalizacji globalnej. Niezbędne do testowania są specjalnie przygotowane funkcje benchmarkowe, które badają skuteczność algorytmów. W tej części przedstawiono również test Wilcooxona, który wykorzystuje się do porównania wyników funkcji benchmarkowych.

Opisane metody testowania zostały użyte w rozdziale, który przedstawia testy wykonane na algorytmie CMA-ES oraz jego modyfikacjach. Modyfikacje polegały wyłącznie na zmianie metod uwzględniania ograniczeń. Wyniki zostały zebrane i ze sobą porównane.

Ostatnim rozdziałem jest podsumowanie, które syntetyzuje wyniki badań. Znajdują się tam wnioski wynikające z całej pracy. Rozdział ten zawiera również pomysły, w jaki sposób można poprawić algorytm CMA-ES.

2. Wstęp

Algorytmy ewolucyjne wykorzystuje się głównie w problemach z wieloma zmiennymi, np. w planowaniu procesów przemysłowych.[10] Klasyczne algorytmy ewolucyjne nie dostosowują się do charakterystyki optymalizowanej funkcji. W większości z nich rozkład prawdopodobieństwa generowanych punktów nie zmienia się w trakcie symulacji algorytmu. Z tego faktu wynika problem ustalenia parametrów początkowych, np. wariancji rozkładu prawdopodobieństwa. Ustalenie zbyt niskiej wartości wariancji może prowadzić do znalezienia jedynie pewnego optimum lokalnego, natomiast ustalenie zbyt dużej wartości wariancji sprawia, że trudno znaleźć algorytmowi optimum globalne.

Naprzeciw problemowi ustalenia wariancji wychodzi algorytm CMA-ES. Już samo rozwinięcie akronimu CMA-ES mówi o tym: Covariance Matrix Adaptation - Evolution Strategy (adaptacja macierzy kowariancji - strategia ewolucyjna). Jest to algorytm oparty na technikach ewolucyjnych, aczkolwiek punkty losowane są na podstawie macierzy kowariancji (wielowymiarowe uogólnienie wariancji), która jest w każdej iteracji dostosowywana do aktualnej sytuacji przeszukiwań. Takie rozwiązanie sprawia, że dobór parametrów początkowych nie jest tak istotny jak w innych algorytmach ewolucyjnych.

Mimo kilkunastu lat istnienia algorytmu CMA-ES, podejmowane są kolejne próby jego optymalizacji ([3][11]). Warto jednocześnie pamiętać, że w większości wariantów algorytmu CMA-ES trudno mówić o uniwersalnej optymalizacji. Zazwyczaj poprawa w pewnej grupie funkcji skutkuje pogorszeniem wyników w innym rodzaju funkcji.

Ta praca zawiera badania, które mogą być pomocą podczas optymalizacji algorytmu CMA-ES skoncentrowanej na ograniczeniach kostkowych. Algorytm CMA-ES został stworzony głównie do problemów bez ograniczeń. Można go jednak stosować również do problemów z ograniczeniami, a w literaturze trudno znaleźć prace poświęcone temu zagadnieniu. Warto to zbadać, ponieważ ograniczenia napotykamy w wielu problemach - materiały mają wytrzymałość, komputery skończoną pamięć, a czasem ogranicza prędkość światła.

W aktualnej wersji algorytmu CMA-ES punkty, które znajdują się poza ograniczeniem, są na to ograniczenie rzutowane (o rzutowaniu można przeczytać w rozdziale 4.1.2). Jest to tylko jedna z metod uwzględniania ograniczeń. Warto zauważyć, że za-

miast rzutować punkt na ograniczenie, można go reiniicjować, powtórnie losować, czy też przekształcić symetrycznie względem ograniczeń.

Cel pracy

Celem pracy jest weryfikacja wpływu uwzględniania ograniczeń na algorytm CMA-ES. Zbadanych będzie kilka metod uwzględniania ograniczeń. Cel będzie realizowany poprzez teoretyczne rozważania oraz przeprowadzenie testów.

3. Algorytm CMA-ES

Zgodnie z tym, co zostało ustalone we wstępnie, algorytm CMA-ES należy do rodziny algorytmów ewolucyjnych. Cechą wyróżniającą ten algorytm jest specyficzny sposób generowania punktów. Można to zauważać w poniższym pseudokodzie opisującym algorytm CMA-ES (równania są cytowaniami z [7]):

inicjuj zmienne

dopóki nie są spełnione warunki stopu:

generuj punkty

$$(1) \quad x_k^{t+1} \sim m^t + \sigma^t N(0, C^t)$$

oblicz wartości funkcji w wygenerowanych punktach

posortuj punkty

$$(2) \quad f(x_{1:\lambda}^{t+1}) \leq f(x_{2:\lambda}^{t+1}) \leq \dots \leq f(x_{\lambda:\lambda}^{t+1})$$

przeprowadź selekcję punktów

oblicz wartość oczekiwana rozkładu

$$(3) \quad m^{t+1} = m^t + c_m \sum_{i=1}^{\mu} w_i (x_{i:\lambda}^{t+1} - m^t)$$

oblicz długość kroku

$$(4) \quad p_{\sigma}^{t+1} = (1 - c_{\sigma}) p_{\sigma}^t + \sqrt{c_{\sigma}(2 - c_{\sigma}) \mu_{eff}} C^{t - \frac{1}{2} \frac{m^{t+1} - m^t}{\sigma^t}}$$

$$(5) \quad \sigma^{t+1} = \sigma^t \exp\left(\frac{c_{\sigma}}{d_{\sigma}} \left(\frac{\|p_{\sigma}^{t+1}\|}{E \|N(0, I)\|} - 1\right)\right)$$

oblicz macierz kowariancji

$$(6) \quad p_c^{t+1} = (1 - c_c) p_c^t + \sqrt{c_c(2 - c_c) \mu_{eff}} \frac{m^{t+1} - m^t}{\sigma^t}$$

$$(7) \quad C^{t+1} = (1 - c_1 - c_{\mu} \sum_{j=1}^{\lambda} w_j) C^t + c_1 p_c^{t+1} (p_c^{t+1})^T + c_{\mu} \sum_{i=1}^{\lambda} w_i y_{i:\lambda}^{t+1} (y_{i:\lambda}^{t+1})^T$$

gdzie:

- x_k^t - k-ty punkt iteracji numer t
- m^t - wartość oczekiwana rozkładu w iteracji t
- σ^t - długość kroku iteracji t
- $N(p, C)$ - losowanie punktu z rozkładem normalnym o wartości oczekiwanej w punkcie p oraz macierzy kowariancji C

- C^t - macierz kowariancji iteracji t
- $f(x)$ - wartość funkcji w punkcie x
- λ - wielkość populacji (ilość punktów x w jednej iteracji)
- c_m - współczynnik nauki wartości oczekiwanej
- μ - liczba wybieranych punktów spośród wszystkich λ punktów (patrz 3.5 Selekcja punktów)
- w_i - waga punktu x_i
- p_σ^t - sprzężona ścieżka ewolucji w iteracji t
- c_σ - współczynnik nauki długości kroku
- $\mu_{eff} = \frac{1}{\sum_{i=1}^{\mu} w_i^2}$
- $d_\sigma \approx 1$ - stała tłumienia
- p_c^t - ścieżka ewolucji w iteracji t
- c_c - współczynnik nauki macierzy kowariancji
- $c_1 \approx \frac{2}{n^2}$
- $c_\mu \approx \min(\frac{\mu_{eff}}{n^2}, 1 - c_1)$
- $y_{i:\lambda}^{t+1} = \frac{x_{i:\lambda}^{t+1} - m^t}{\sigma^t}$

Najważniejsze części algorytmu są opisane poniżej w następującej kolejności: macierz kowariancji, wartość oczekiwana rozkładu, długość kroku, generowanie punktów, selekcja punktów.

3.1. Macierz kowariancji

Punkty w algorytmie CMA-ES są generowane poprzez losowanie na podstawie macierzy kowariancji C^t . Oznacza to, że nie istnieje bezpośredni związek pomiędzy punktami kolejnych iteracji - w klasycznych algorytmach ewolucyjnych zazwyczaj można było mówić o rodzinach wygenerowanego punktu. Punkty wpływają jednak na macierz kowariancji, a pośrednio przez nią na kolejną generację punktów. Odbiera się to w następujący sposób: spośród wszystkich punktów wybiera się μ punktów (zazwyczaj połowę), która zwraca lepsze wartości funkcji celu. Następnie przypisuje się im wagę w - w zależności od tego, jak dobry jest punkt. Na podstawie tych danych wyznaczana jest empiryczna macierz kowariancji (równania 6-7).

3.2. Wartość oczekiwana rozkładu

Punkty generowane są na podstawie rozkładu określonego macierzą kowariancji. Sama macierz nie mówi jednak nic o wartości oczekiwanej. Wymaga to oddzielnego wyznaczenia tej wartości(3).

Podobnie jak w przypadku macierzy kowariancji, wybiera się μ punktów. Dla tych punktów wyznaczane są wektory przesunięć względem wartości oczekiwanej (każdy wektor jest skalowany przez wagę punktu). Wektory te sumujemy otrzymując pewien wektor wynikowy, który następnie jeszcze skalujemy przez współczynnik nauki c_m . O tak otrzymany wektor przesuwamy dotychczasową wartość oczekiwana.

3.3. Długość kroku

Generowanie punktów z macierzy kowariancji pozwala na skalowanie tego rozkładu poprzez użycie dodatkowego parametru σ^t . Ten parametr jest nazywany długością kroku. W algorytmie CMA-ES długość kroku obliczana jest na podstawie kierunku przemieszczania się wartości oczekiwanej rozkładu. Długość kroku zwiększa się, gdy wartość oczekiwana przemieszcza się w tym samym kierunku. Skracą natomiast, gdy kierunek jest zmienny.

3.4. Generowanie punktów

Punkty są generowane poprzez wylosowanie wektora liczb na postawie macierzy kowariancji. Wylosowany wektor jest skalowany przez długość kroku, a na koniec jest przesuwany o wektor równy wartości oczekiwanej rozkładu.

3.5. Selekcja punktów

Selekcja punktów, to wybór μ punktów spośród całej populacji. Wybierane są te punkty, które mają lepszą wartość funkcji (2). Wartość funkcji wpływa również na przypisywane punktom wagi, o których wspomniano wcześniej.

3.6. Ocena

Powyższe opisy pokazują, że algorytm CMA-ES posiada wiele zmiennych i parametrów. Zmienne, takie jak macierz kowariancji czy długość kroku, są wyznaczane na podstawie rozbudowanych równań. W tym miejscu powstaje pytanie, czy warto zajmować się takim algorytmem, gdy istnieją rozwiązania o przejrzystej strukturze. Odpowiedzią niech będą wyniki badań [12], które pokazują, że dla skomplikowanych funkcji algorytm CMA-ES zwraca bardzo dobre rezultaty.

4. Techniki uwzględniania ograniczeń

Niektóre problemy optymalizacyjne posiadają ograniczenia. Szukając rozwiązania należy zapewnić, że będzie ono dopuszczalne. Bazując na [1] techniki uwzględniania ograniczeń można podzielić ze względu na wykorzystanie:

- definicji przestrzeni przeszukiwań - zapewnienie, że podczas krzyżowań, mutacji i innych zmian punktów, żaden z punktów nie wypadnie poza przestrzeń przeszukiwań,
- modyfikacji funkcji celu - zmienienie funkcji celu tak, aby funkcja celu dla punktów spoza ograniczeń zwracały gorsze wyniki,
- transformacji rozwiązań - punkty, które są poza ograniczeniami zostają zamieniane na punkty, które znajdują się w ograniczeniach.

Każda z tych technik posiada wiele wariantów. Przeanalizowanie wszystkich mogłoby być bardzo czasochłonne. W związku z tym w tej pracy skupiono się na transformacji rozwiązań.

Ograniczenia w problemach mogą różnić się ze względu na swój charakter, np. mogą być nieciągłe lub punktowe. Niniejsza praca skupia się na ograniczeniach kostkowych.

4.1. Transformacje rozwiązań

Istnieje wiele technik transformacji rozwiązań. W kolejnych podrozdziałach znajdują się metody transformacji, które były badane na potrzeby tej pracy. Każda z technik jest opisana słownie oraz pseudokodem. Opis słowny zawiera wyjaśnienie, co się dzieje z punktem, który znalazł się poza ograniczeniem. W pseudokodzie zastosowano następujące oznaczenia:

- x - punkt, który poddajemy naprawie
- x' - rodzic punktu x ; z punktu x' z zadanym rozkładem został wygenerowany punkt x
- $x(i)$ - wartość i -tej współrzędnej punktu x
- lb - ograniczenie dolne
- ub - ograniczenie górne

4.1.1. Metoda konserwatywna

Nowy punkt zostaje odrzucony i wraca do poprzedniej pozycji.

$x = x'$

4.1.2. Rzutowanie

Punkt jest transformowany do najbliższego punktu, który spełnia ograniczenia kostkowe. Oznacza to, że dla każdej współrzędnej sprawdzany jest warunek zawierania się w ograniczeniach. Wartości współrzędnych, dla których nie jest on spełniony, zmieniane są na wartości ograniczeń (odpowiednio dolne lub górne), które są najbliższe.

```
dla każdej współrzędnej i
    jeżeli lb(i) > x(i)
        x(i) = lb(i)
    jeżeli ub(i) < x(i)
        x(i) = ub(i)
```

4.1.3. Reinicjacja

Punkt jest przenoszony do ustalonej pozycji - często jest to punkt początkowy.

$x = xr$

4.1.4. Powtórnna generacja

Punkt jest powtórnie generowany do momentu, aż spełni ograniczenie.

```
dopóki !w_ograniczeniach(x)
    x = generuj(x')
```

4.1.5. Zawijanie

Dla każdej współrzędnej sprawdzane są warunki ograniczeń. W przypadku współrzędnych, na których punkt jest poza ograniczeniem, różnica pomiędzy ograniczeniem a wartością współrzędnej punktu jest zapamiętywana. Tę różnicę odkładamy na przeciwnym ograniczeniu po stronie, która jest wewnętrz ograniczeń. W tym miejscu znajduje się nowa wartość współrzędnej punktu. Z perspektywy punktu nie ma ograniczeń, a przestrzeń przeszukiwań po każdym wymiarze jest jakby "zawinięta".

```

dla każdej współrzędnej i
    jeżeli  $lb(i) > x(i)$ 
         $x(i) = ub(i) - (lb(i) - x(i))$ 
    jeżeli  $ub(i) < x(i)$ 
         $x(i) = lb(i) + (x(i) - ub(i))$ 

```

4.1.6. Odbicie

Dla każdej współrzędnej sprawdzane są warunki ograniczeń. W przypadku współrzędnych, na których punkt jest poza ograniczeniem, wartość punktu tej współrzędnej jest symetrycznie odbita względem ograniczenia, którego warunek został złamany.

```

dla każdej współrzędnej i
    jeżeli  $lb(i) > x(i)$ 
         $x(i) = x(i) + 2 * (lb(i) - x(i))$ 
    jeżeli  $ub(i) < x(i)$ 
         $x(i) = x(i) - 2 * (x(i) - ub(i))$ 

```

4.2. Błądzenie przypadkowe

Na podstawie pseudokodu zawartego w rozdziale 3 (szczególnie równań 4-5) można się spodziewać, że algorytm CMA-ES dla funkcji stałej będzie zachowywał się analogicznie do błądzenia przypadkowego. Takie założenie skłoniło autora, żeby zbadać błądzenie przypadkowe z ograniczeniami. Błądzenie przypadkowe jest algorytmem prostszym niż CMA-ES, więc umożliwia szybsze testowanie i wyciąganie wniosków.

Niech X_1, X_2, \dots będą niezależnymi n-wymiarowymi zmiennymi losowymi o wartości oczekiwanej równej $[0]^n$. Błądzeniem przypadkowym nazywamy sekwencję punktów S_1, S_2, \dots , takich że:

$$S_1 = X_1, S_i = S_{i-1} + X_i \quad (1)$$

Zmienne losowe mogą być realizowane w różny sposób. Mogą to być wektory o rozkładzie normalnym, jednostajnym lub innym.

4.3. Metoda przeprowadzania testów

Celem testów jest zaobserwowanie, jaki wpływ na symulację ma metoda uwzględniania ograniczeń. Do tak postawionego celu można podejść na różne sposoby. Wybrano 2 cechy, które miały zrealizować postawiony cel:

1. Wartość oczekiwana generowanego punktu (razem z poprawą).
2. Rozkład prawdopodobieństwa generowanych punktów.

Sposoby testowania obu cech są opisane w kolejnych podrozdziałach.

4.3.1. Wartość oczekiwana generowanego punktu

Wartością oczekiwana liczbą X_i w błędzeniu przypadkowym jest 0. Implikuje to fakt, że wartością oczekiwana S_i jest S_{i-1} . Ustalenie ograniczeń kostkowych powoduje, że nadal losujemy liczbę X_i , której wartość oczekiwana jest równa 0. Dodajemy jednak metody naprawy. W związku z tym wartością oczekiwana S_i nie zawsze jest S_{i-1} .

Dla każdego punktu p przestrzeni przeszukiwań można przypisać wektor \vec{d} . Początek \vec{d} znajduje się w p . Założmy teraz, że p jest punktem S_i pewnego błędzienia przypadkowego z ograniczeniami. Koniec wektora \vec{d} znajduje się w wartości oczekiwanej punktu S_{i+1} .

Wykres, który będzie zawierał wektory \vec{d} , pozwoli na określenie charakterystyki przemieszczania się punktów w poszczególnych częściach przestrzeni przeszukiwań.

Implementacja

Do wizualizacji wybrano symulację, w której punkty pochodzą z przestrzeni dwuwymiarowej. Pozwala to na przejrzystą wizualizację wektorów \vec{d} oraz pokazuje zachowanie wektorów, które są blisko kilku ograniczeń.

Do symulacji wybrano punkty równomiernie rozłożone na prostokątnej siatce dwuwymiarowej. Dla każdego z tych punktów przeprowadzono następujące obliczenia. 1000 razy uruchomiono symulację jednego kroku algorytmu błędzienia przypadkowego (z rozkładem normalnym) z odpowiednią naprawą. Otrzymano tysiąc punktów, z których następnie obliczono średnią arytmetyczną. Tak uzyskany punkt potraktowano jako wartość oczekiwana i wyrysowano wektor \vec{d} .

4.3.2. Rozkład prawdopodobieństwa generowanych punktów

Punkty w błędzeniu przypadkowym poruszają się chaotycznie. Ta losowość jest częściami uporządkowana na ograniczeniach. W zależności od metody naprawy, punkty zachowują się inaczej, a rozkład prawdopodobieństwa jest zniekształcony. Autor postanowił przyjrzeć się jak wygląda rozkład prawdopodobieństwa wystąpienia punktu z perspektywy całej symulacji.

Dla każdej metody naprawy opisanej w 4.1 jednokrotnie uruchomiono algorytm błędnego przypadkowego jednego punktu. Oddziennie symulowano również błądzenie z rozkładem normalnym oraz jednostajnym na przedziale $[-0.5; 0.5]$ (przedział co najmniej kilkukrotnie krótszy od ograniczeń kostkowych). Po każdej iteracji nowo wygenerowany punkt zapisywano w tablicy. Z tak przygotowanej tablicy generowano histogram.

Implementacja

Dla każdej z metod naprawy przygotowano oddzielny skrypt. Pseudokod skryptu zamieszczono poniżej.

```
x - błędzący punkt
punkty - tablica wszystkich położen punktu x
iteracje - liczba iteracji podana jako parametr
i = 0
dopóki i < iteracje
    d = wektor wylosowany z zadanym rozkładem
    x = x + d
    jeżeli x jest poza ograniczeniem
        popraw x
    dodaj x do tablicy punkty
    i = i + 1
```

4.4. Wyniki testów

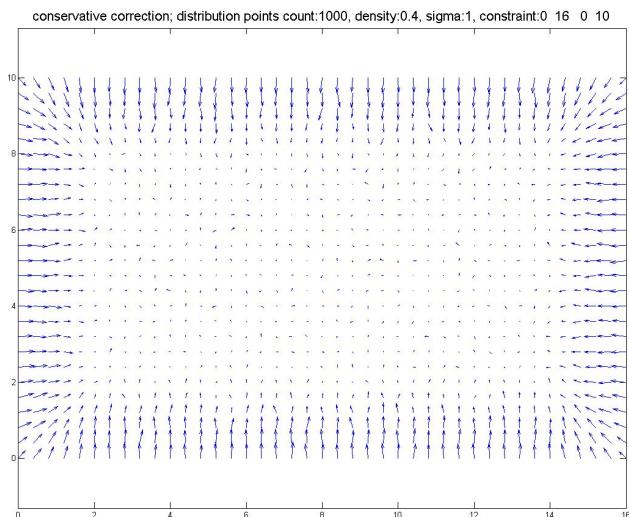
Wykresy zamieszczone w tym podrozdziale są 2 rodzajów. Są to wykresy wektorów \vec{d} przesunięć wartości oczekiwanych oraz histogramy wystąpień punktu.

Histogramy były tworzone poprzez symulację skryptów z liczbą iteracji wynoszącą 2-100 milionów. Szerokość przedziałów jest różna i była dobierana tak, aby jak najle-

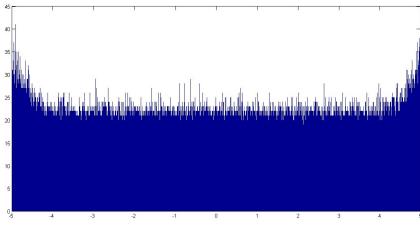
piej przedstawić interesujące fakty. Wszystkie wykresy, które pokazują wyniki błędzenia przypadkowego w dwóch wymiarach zostały przetworzone w następujący sposób: otrzymane wyniki symulacji powielono poprzez symetryczne odbicie ich względem osi X, osi Y oraz początku układu współrzędnych. Wykonano ten zabieg, aby zwiększyć dokładność wyników. Nie wpływają one na badania, ponieważ punkty 0 są środkami ograniczeń obu osi, a z perspektywy algorytmu nie ma znaczenia, w której ćwiartce znajduje się aktualnie punkt.

Metoda konserwatywna

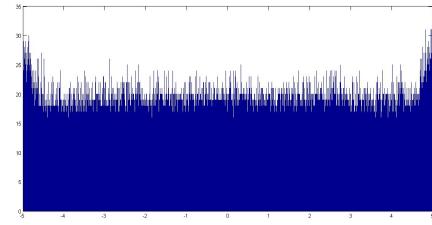
Wektory wartości oczekiwanych, które są blisko ograniczeń są skierowane od ograniczenia. Histogramy z kolei pokazują, że punkty z większym prawdopodobieństwem występują przy ograniczeniach. Takie zachowanie wynika z tego, że potomkowie punktów blisko granicy mogą "wypaść" poza ograniczenie. Po naprawie dziecko będzie tym samym punktem.



Rysunek 1: Wykres wektorów przesunięć wartości oczekiwanych

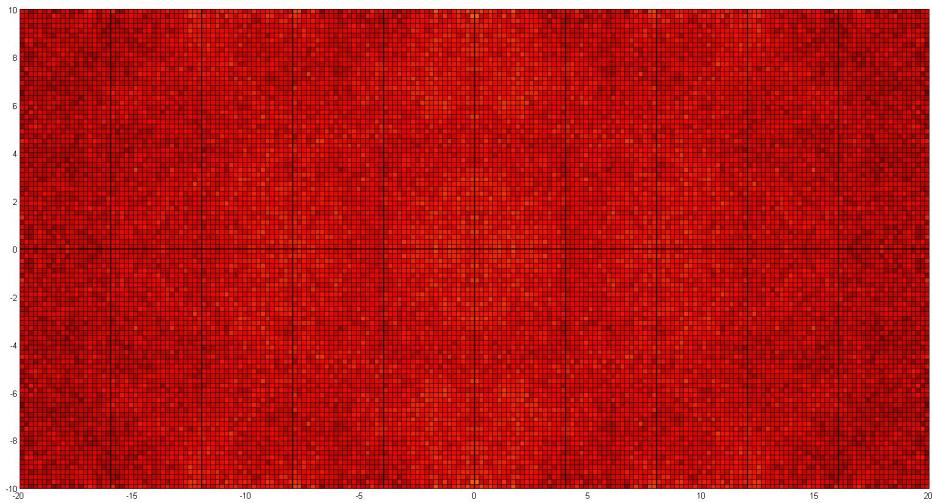


(a) Generowanie z rozkładem normalnym



(b) Generowanie z rozkładem jednostajnym

Rysunek 2: Histogram wystąpień punktów; 1 wymiar



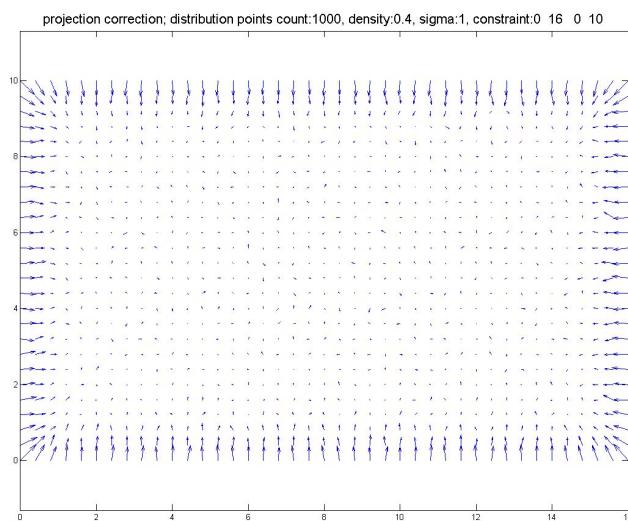
Rysunek 3: Histogram wystąpień punktów; 2 wymiary; generowanie z rozkładem normalnym

Rzutowanie

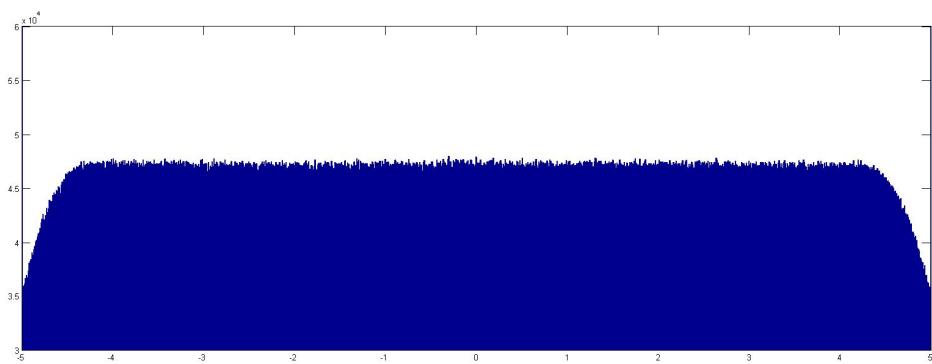
Zgodnie z przewidywaniami największe prawdopodobieństwo wystąpienia punktu jest na ograniczeniu, co bardzo dobrze obrazuje rysunek 7. Na rysunkach 5 i 6 zakres osi Y został zmniejszony do odpowiednio $[30000; 60000]$ oraz $[70000; 140000]$, aby uwypuklić interesujące efekty. W związku z tym, wartości przedziałów brzegowych nie są widoczne - były o kilka rzędów wielkości większe od pozostałych wartości.

Warto zwrócić uwagę na niespodziewane zjawisko - wartości przedziałów blisko ograniczeń są mniejsze, niż pozostałe. Oznacza to, że punkty w tych przedziałach występują z mniejszym prawdopodobieństwem, niż pozostałe. Wydawać się mogło, że powinno być inaczej, ponieważ podczas symulacji punkty często są rzutowane na ograniczenie.

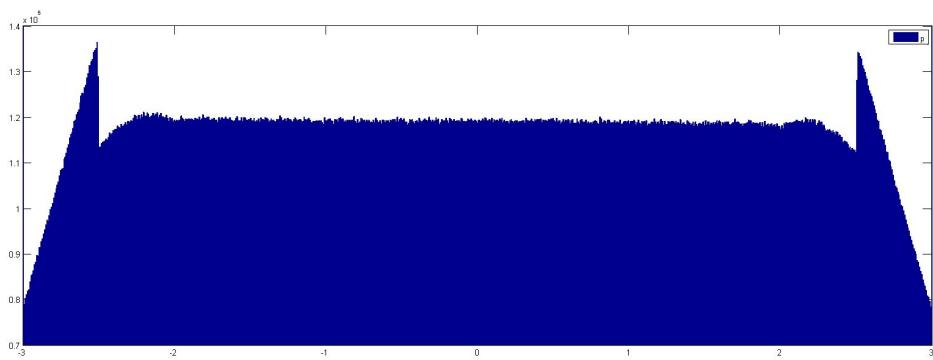
Dodatkowo, w rozkładzie jednostajnym widać, że istnieją przedziały, w których punkty występują z większym prawdopodobieństwem. Autorowi nie udało się formalnie uzasadnić tego zjawiska. Intuicja prowadzi do hipotezy, że punkty, które znalazły się poza ograniczeniem, bez naprawy po kilku iteracjach zapełniałyby "dołek". Naprawa natomiast sprawia, że cała dalsza symulacja zostaje niejako przesunięta. W ten sposób powstaje "górnka" w rozkładzie jednostajnym. Brak górnki w rozkładzie normalnym można tłumaczyć charakterystyką samego rozkładu - spadek prawdopodobieństwa wraz z oddalaniem się od wartości oczekiwanej.



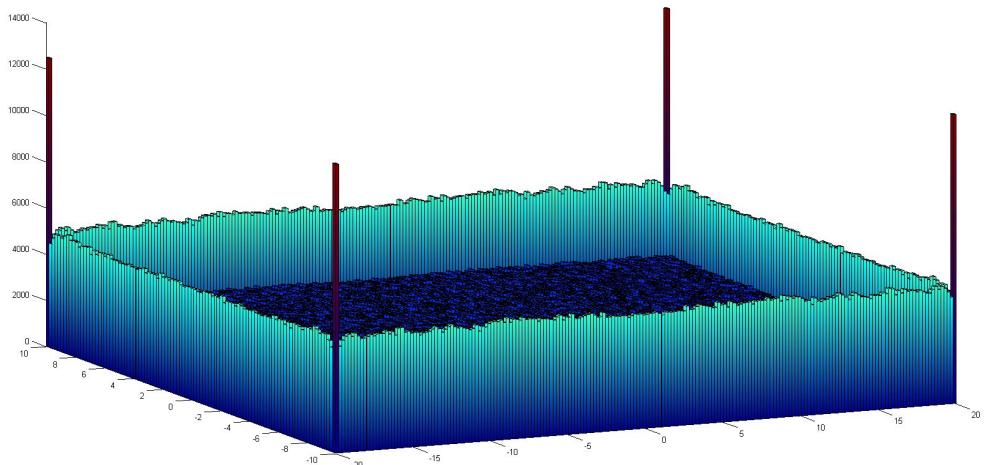
Rysunek 4: Wykres wektorów przesunięć wartości oczekiwanych



Rysunek 5: Histogram wystąpień punktów; 1 wymiar; generowanie z rozkładem normalnym



Rysunek 6: Histogram wystąpień punktów; 1 wymiar; generowanie z rozkładem jednostajnym



Rysunek 7: Histogram wystąpień punktów; 2 wymiary; generowanie z rozkładem normalnym

Reinicjacja

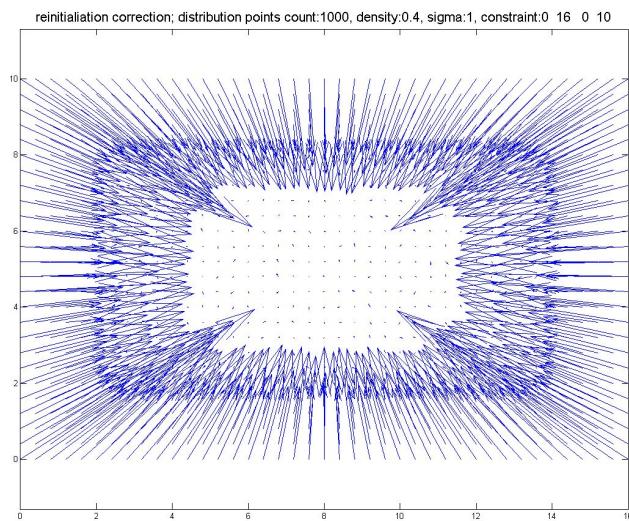
Wykres wartości oczekiwanych jest mało czytelny, ponieważ wartości przesunięć są duże. Ma w tym udział reinicjacja, która przesuwa część punktów na środek przestrzeni przeszukiwań.

Histogramy nie ukazują nic nadzwyczajnego, ponieważ łatwo zauważać pik związany z reinicjacją oraz wartości histogramu malejące wraz ze zbliżaniem się do ograniczeń. Warto jednak zwrócić uwagę na 2 fakty.

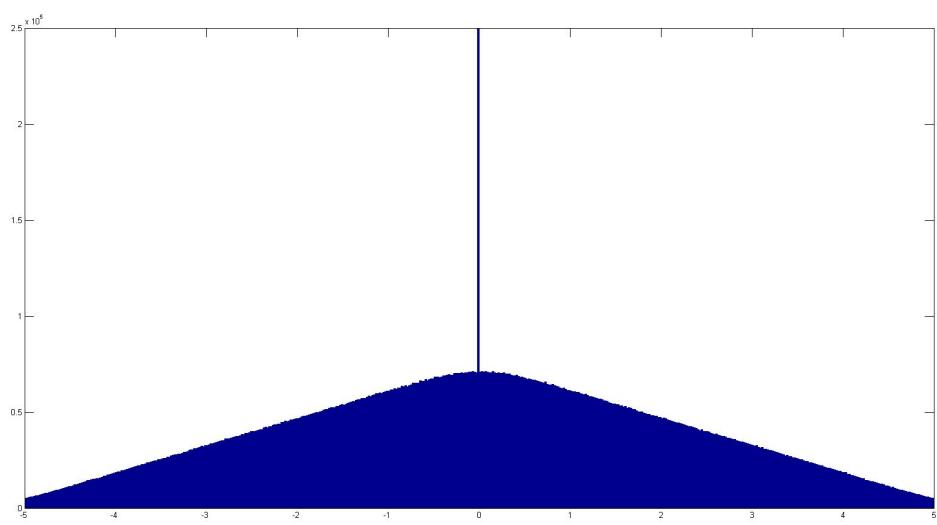
Pierwsze spostrzeżenie, to kształt histogramu. Zarówno dla rozkładu jednostajne-

go, jak i normalnego w jednym wymiarze jest on stożkowy. Łuk można zauważyc tylko blisko punktu środkowego, w pozostałej części spadek jest liniowy. Brakuje charakterystycznego, gaussowskiego przegięcia. Sytuacja jest ciekawsza, gdy występuje więcej wymiarów. Widać wówczas przegięcie (Rysunek 12). Dokładniejsze badania pokazały, że przegięcie nie występuje tylko na jednym wymiarze - tym, który jest relatywnie najkrótszy. Celowo jest użyte słowo relatywnie, ponieważ z perspektywy błędzenia przypadkowego i rozkładu normalnego trzeba brać pod uwagę parametr σ - odchylenie standardowe. Wymiary o małym σ będą relatywnie dłuższe od tych z dużym σ , ponieważ błędzenie będzie wykonywało mniejsze kroki.

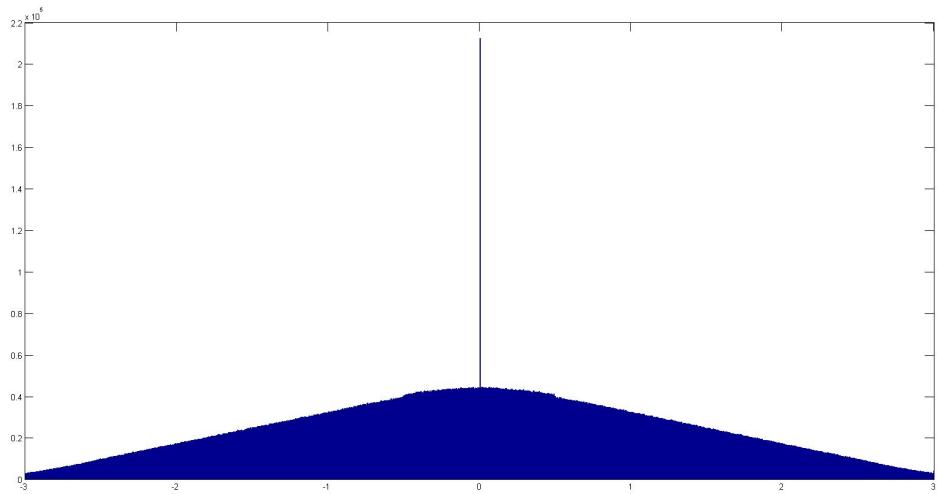
Na rysunku 10 można dostrzec też dwa uskoki, które związane są z pikiem w punkcie 0 oraz charakterystyką rozkładu jednostajnego.



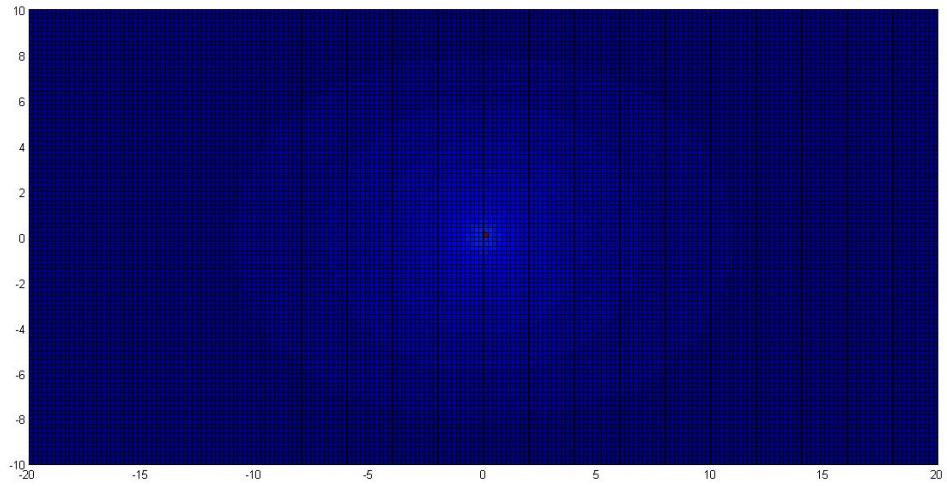
Rysunek 8: Wykres wektorów przesunięć wartości oczekiwanych



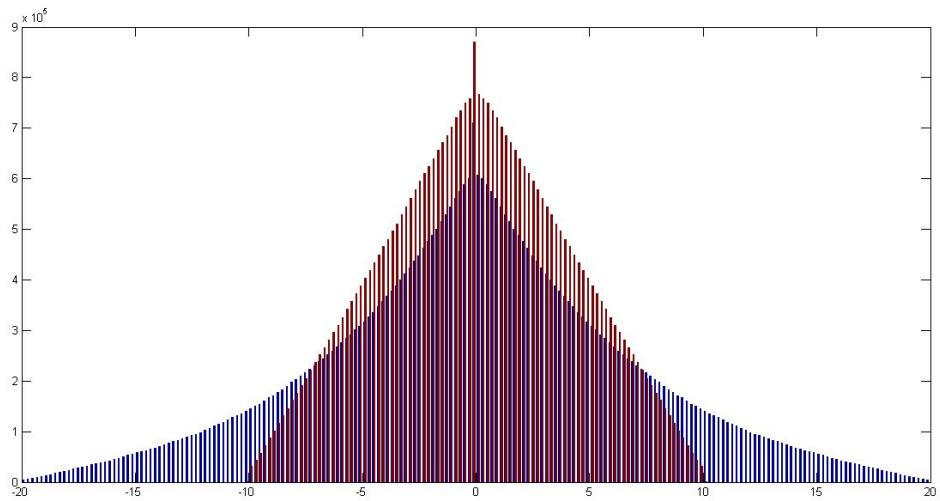
Rysunek 9: Histogram wystąpień punktów; 1 wymiar; generowanie z rozkładem normalnym



Rysunek 10: Histogram wystąpień punktów; 1 wymiar; generowanie z rozkładem jednostajnym



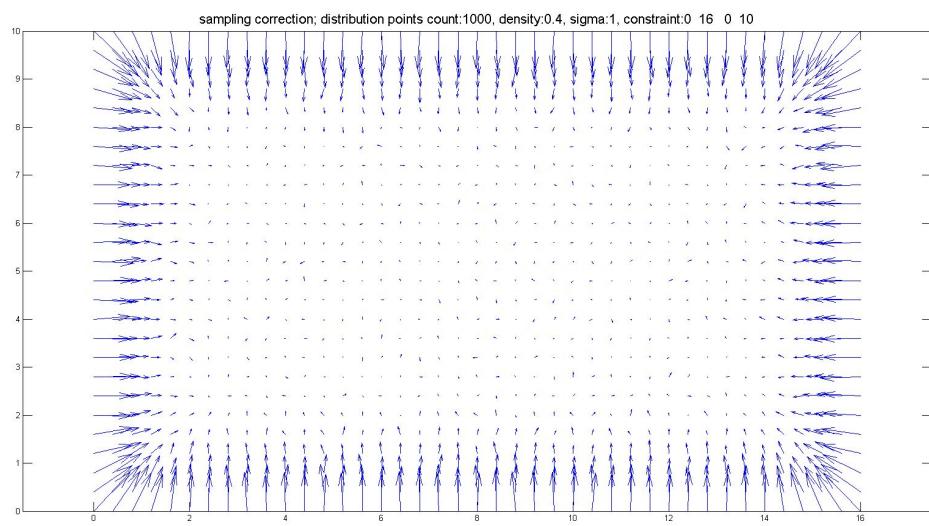
Rysunek 11: Histogram wystąpień punktów; 2 wymiary; generowanie z rozkładem normalnym



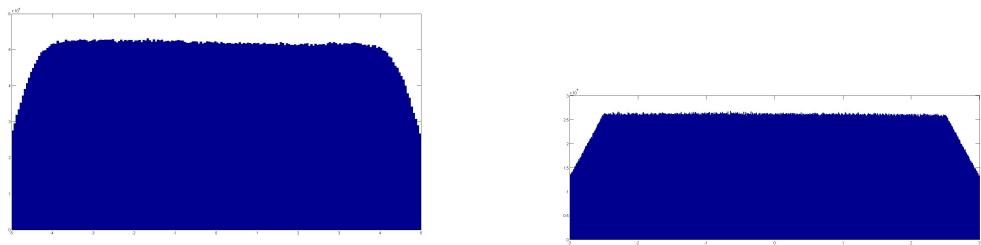
Rysunek 12: Histogram wystąpień punktów; 2 wymiary; generowanie z rozkładem normalnym; oddzielne histogramy dla obu wymiarów

Powtórna generacja

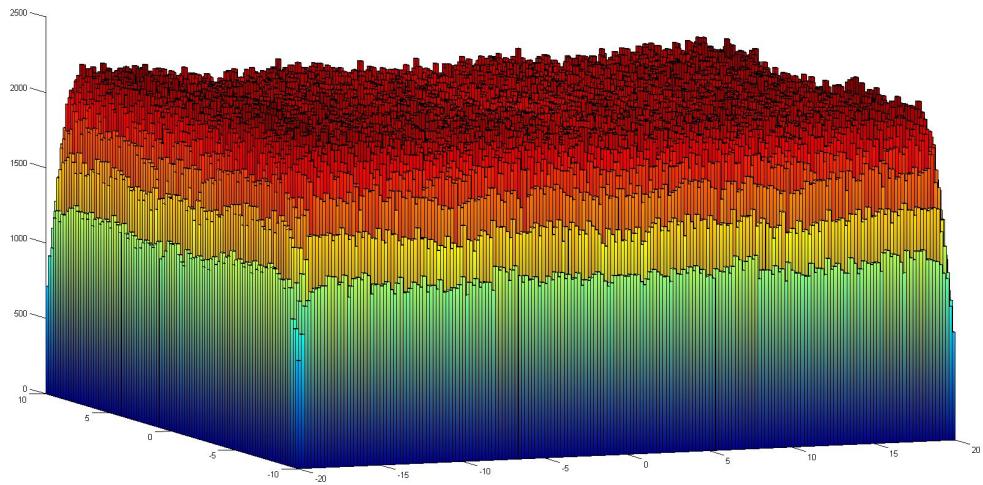
Ten rozkład charakteryzuje się spadkiem wartości prawdopodobieństwa wraz ze zbliżaniem się do ograniczenia. Punkty, które wypadłyby poza ograniczenia oraz ich potomkowie są przesuwane w kierunku środka przedziału.



Rysunek 13: Wykres wektorów przesunięć wartości oczekiwanych



Rysunek 14: Histogram wystąpień punktów; 1 wymiar

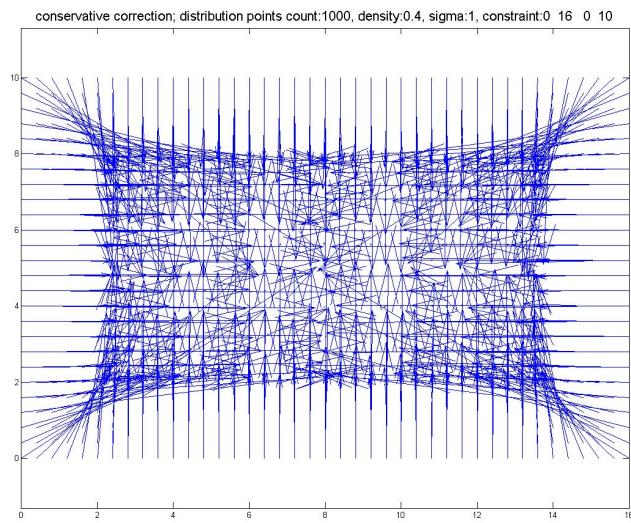


Rysunek 15: Histogram wystąpień punktów; 2 wymiary; generowanie z rozkładem normalnym

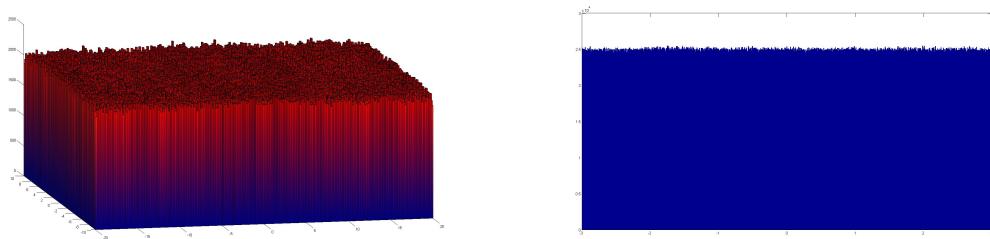
Zawijanie

Wykres przesunięć wartości oczekiwanych jest bardzo nieczytelny. Jest to implikacji dużych przesunięć punktów - te, które znajdują się poza ograniczeniem, wędrują na drugą stronę wykresu.

Uzasadniając wykresy histogramów warto wyobrazić sobie, że symulacja przeprowadzana jest bez ograniczeń, a następnie wykres z wynikami przecinany jest w równych odsłapech wzdłuż każdej osi. Na koniec tak pocięte części łączone są w jeden wykres. Takie zachowanie symuluje zawijanie. W związku z tym nie dziwi fakt, że histogramy przedstawiają rozkład jednostajny.



Rysunek 16: Wykres wektorów przesunięć wartości oczekiwanych



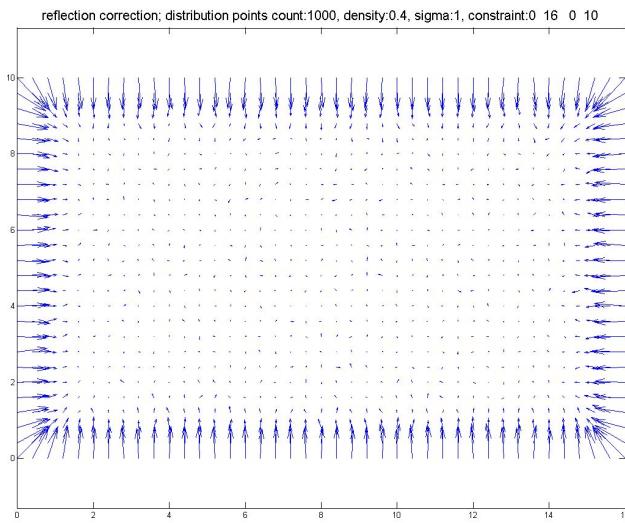
(a) 1 wymiar; generowanie z rozkładem normalnym (b) 2 wymiary; generowanie z rozkładem jednostajnym

Rysunek 17: Histogram wystąpień punktów;

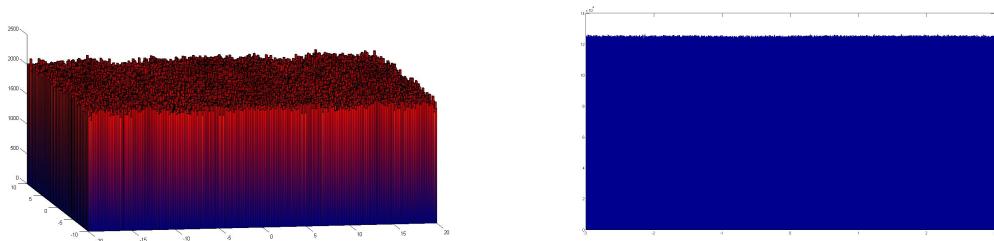
Odbicie

Podobnie jak zawijanie, odbicie zwraca histogram rozkładu jednostajny. W tej sytuacji nieco trudniej o analogię, lecz po chwili zastanowienia nie dziwi kształt poniższych wykresów.

Od zawijania różni się natomiast wykres wartości oczekiwanych, ponieważ w odbiciu nie ma znaczących przesunięć punktów.



Rysunek 18: Wykres wektorów przesunięć wartości oczekiwanych



(a) 1 wymiar; generowanie z rozkładem normalnym (b) 2 wymiary; generowanie z rozkładem jednostajnym

Rysunek 19: Histogram wystąpień punktów;

4.5. Wnioski

W większości metod zachowanie pojedynczego punktu przekłada się na globalne zachowanie całego algorytmu. Testy pokazały cechy, które nie były oczywiste podczas prostej analizy algorytmów.

Z perspektywy algorytmu CMA-ES warto zwrócić uwagę przede wszystkim na metody zawijania i reinicjacji. W tych metodach wartości oczekiwane punktów znajdują się daleko od rodzica. Oznacza to, że w algorytmie CMA-ES macierz kowariancji może znaczco się zwiększać w porównaniu do metod bez ograniczeń.

5. Metodyka testowania metod optymalizacji globalnej

Funkcje posiadają charakterystyczne cechy, m.in. monotoniczność, liczba ekstremów, ciągłość. Metody optymalizacji globalnej również posiadają swoje charakterystyczne własności. W zależności od przypadku, wymagania stawiane algorytmowi mogą być różne: szybkość działania, przeszukanie całej przestrzeni, czy też dokładne określenie ekstremum. Połączenie tego wszystkiego powoduje, że trudno jednoznacznie określić, która metoda optymalizacji jest najlepsza. Warto jednak posiadać wiedzę w których przypadkach metoda przynosi lepsze rezultaty.

Zapewne zgodzili by się z tym stwierdzeniem twórcy konkursów takich jak CEC, czy BBOB. Tworzą oni szereg funkcji benchmarkowych, które mają na celu empirycznie pokazać silne i słabe strony algorytmów w konkretnych sytuacjach.

5.1. Funkcje benchmarkowe

Funkcje benchmarkowe, to złożone funkcje. Skomplikowanie polega między innymi na wielu ekstremach funkcji lub specyficzny umieszczeniu minimum globalnego. Dzięki nim można porównać algorytmy.

5.2. Techniki porównywania wyników

Wybranie funkcji benchmarkowych nie wystarczy, aby stwierdzić, który z algorytmów ewolucyjnych jest lepszy. Są to algorytmy niedeterministyczne - poszukiwanie rozwiązania za każdym razem będzie wyglądało inaczej nawet przy takich samych parametrach. Oznacza to, że dla wiarygodności wyników należy testy uruchamiać wielokrotnie. Posiadając takie wyniki należy posłużyć się wybraną metodą, aby porównać skuteczność algorytmów. W tej pracy do porównywania został wybrany test Wilcooxona, ponieważ można go użyć do porównywania par obserwacji oraz ten test pokazuje różnice cech.

Test Wilcoxona

Test Wilcoxona jest testem nieparametrycznym - nie jest potrzebna wiedza na temat rozkładu populacji [2]. Ten test sprawdza, czy istnieje statycznie istotna różnica między dwoma zbiorami, które były losowane z pewnym rozkładem prawdopodobieństwa. Hipoteza dotyczy median wygenerowanych zbiorów. Hipoteza zerowa H_0 brzmi "Różnica median zbiorów wynosi 0". Na podstawie sprawdzenia hipotezy możemy wywnioskować, czy 2 rozkłady prawdopodobieństwa generują dane istotnie różne.

Do przedstawienia wyników wielu testów Wilcoxona wykorzystuje się tabele. Taka tabela zazwyczaj zawiera etykiety porównywanych zbiorów w wierszach i kolumnach, a na przecięciach znajduje się wynik testu.

6. Weryfikacja wpływu technik uwzględniania ograniczeń na efektywność CMA-ES

6.1. Cel testów

Zrealizowanie celu całej pracy wymaga sprawdzenia wpływu ograniczeń kostkowych na algorytm CMA-ES. Niemożliwe jest zrealizowanie tego celu bez przetestowania samego algorytmu.

Celem testów będzie sprawdzenie charakterystycznych zachowań macierzy kowariancji. Aby zrealizować ten cel, zostaną przeprowadzone testy, które sprawdzają rozkład prawdopodobieństwa punktów. Porównana zostanie także skuteczność algorytmu CMA-ES w zależności od sposobu uwzględniania ograniczeń.

6.2. Metoda przeprowadzenia testów

Przeprowadzone testy zostały oparte na bibliotece przygotowanej przez Nikolausa Hansenego. Tak jak w przypadku błędzenia przypadkowego, wykorzystano implementację w języku MATLAB [6]. Wspomniana biblioteka była modyfikowana w celu zrealizowania testów.

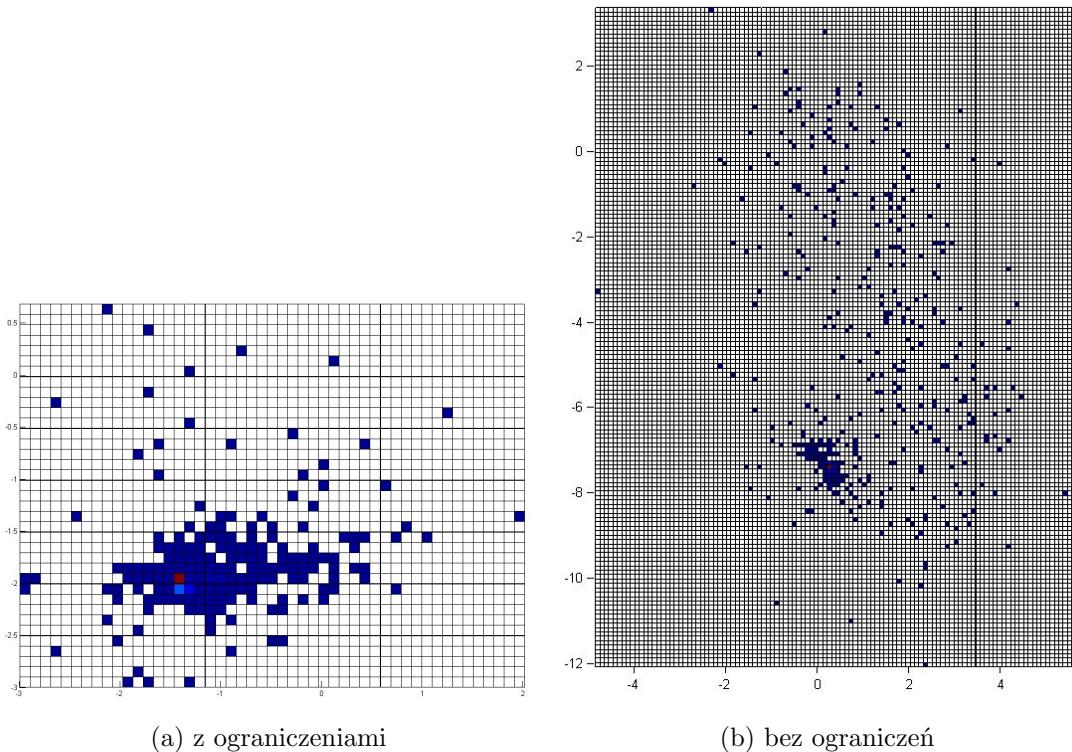
Testy przeprowadzano poprzez jednokrotne uruchomienie zmodyfikowanego algorytmu CMA-ES na funkcji stałej, losowej oraz kwadratowej. Modyfikacja polegała na usunięciu warunków stopu z pętli głównej algorytmu. Wynikała ona z tego, że po kilku iteracjach algorytm się zatrzymywał. Nie modyfikowano natomiast sposobu uwzględniania ograniczeń. Algorytm był przerywany po kilkuset iteracjach, gdy logi w konsoli wskazywały na stagnację algorytmu. Wszystkie wygenerowane przez algorytm punkty zostały przedstawione na histogramie (podobnie jak w przypadku błędzenia przypadkowego).

6.3. Wyniki testów

Funkcja stała

Wbrew oczekiwaniom algorytm CMA-ES uruchomiony na funkcji stałej ma tendencje do zmniejszania kroku algorytmu, więc po kilkuset iteracjach przemieszczanie się wartości oczekiwanej jest znikome. Obrazują to rysunki w tym podrozdziale. Widać na nich fragmenty, w których losowane punkty były znaczowo od siebie oddalone, obszar gdzie odległości pomiędzy losowanymi punktami były małe oraz miejsce, w którym algorytm prawie się nie poruszał (czerwony kwadrat)

Przeprowadzono testy, w których algorytm posiadał ograniczenie $[-3; 3]$ na obu wymiarach oraz test bez ograniczeń.

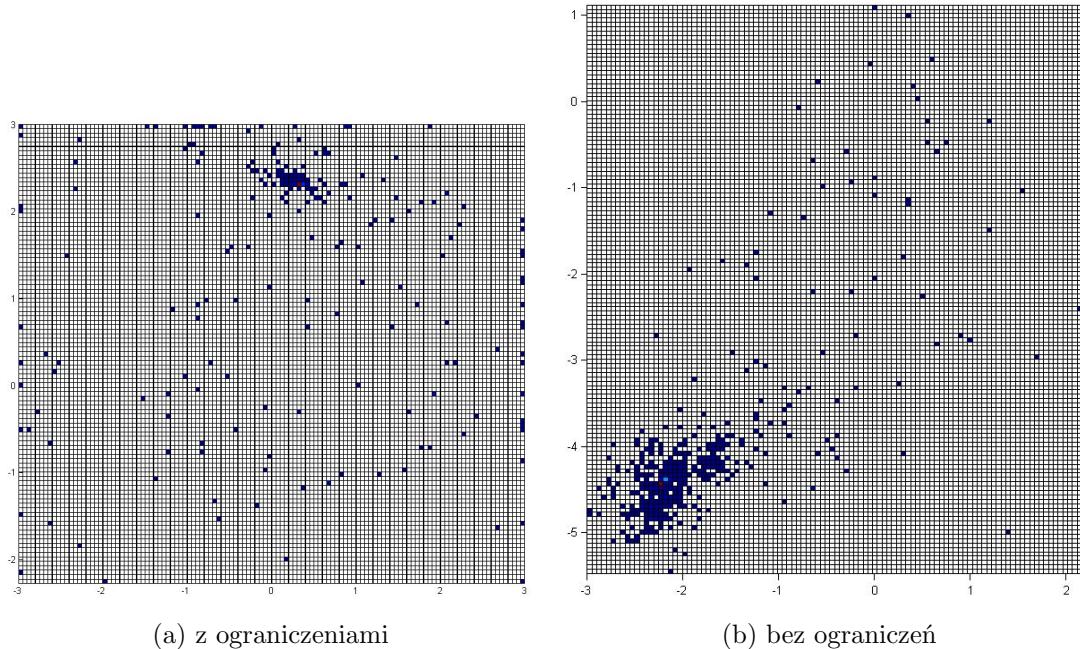


Rysunek 20: Histogram punktów; algorytm CMA-ES; funkcja stała; 2 wymiary

Funkcja losowa

Podobnie jak w przypadku obliczeń funkcji stałej, krok algorytmu CMA-ES na funkcji losowej zmniejszał się dość szybko. Poniższe wykresy, podobnie jak w przypadku testowania funkcji stałej, przedstawiają histogram wystąpień punktów w algorytmie

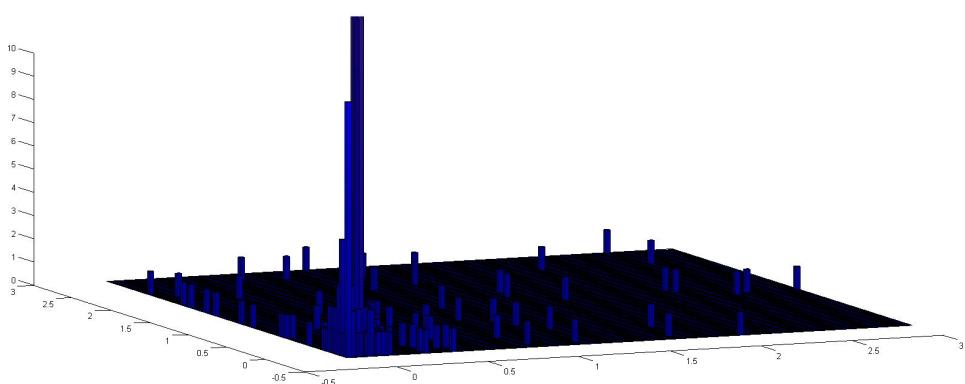
CMA-ES z ograniczeniami oraz bez ograniczenia.



Rysunek 21: Histogram punktów; algorytm CMA-ES; funkcja losowa; 2 wymiary

Funkcja kwadratowa dwuwymiarowa

Badano funkcję o wzorze $y = x_1^2 + x_2^2$. Dla tej funkcji zdecydowano się zmienić ograniczenie. Wynosiło ono $[-0,1;3]$ na obu wymiarach. Początkowa wartość oczekiwana wynosiła $[2;2]$. Wykres dla tak przygotowanej symulacji znajduje się na rysunku 22 (osi Z została obcięta do przedziału $[0;10]$).



Rysunek 22: Histogram punktów; algorytm CMA-ES; funkcja losowa; 2 wymiary

6.4. Wnioski

Na wykresach, które przedstawiają histogramy punktów podczas symulacji algorytmu CMA-ES na funkcji losowej i stałej, nie udało się zaobserwować żadnych efektów widocznych podczas testowania błądzenia przypadkowego. Wynika to z faktu zbiegania symulacji do pewnego punktu - również w przypadku funkcji stałej i losowej. Na podstawie tych wyników trudno cokolwiek powiedzieć o zachowaniu się macierzy kowariancji po dotarciu do ograniczeń kostkowych.

Autor nie zdecydował się na generowanie wykresów innych metod transformacji rozwiązań, ponieważ nie dałoby to widocznych różnic w wykresach.

6.5. Porównanie wariantów transformacji rozwiązań

Celem testów było sprawdzenie, czy zmiana technik uwzględniania ograniczeń w algorytmie CMA-ES wpływa na rozwiązanie. Do tego porównania wybrano 6 wariantów transformacji rozwiązań: rzutowanie na ograniczenie (wykorzystywane w implementacji Hansena), reinicjacja, odbijanie, zawijanie, powtórna generacja oraz wariant ewolucji darwinowskiej, który został opisane poniżej. Dla wszystkich wariantów uruchomiono symulacje algorytmu CMA-ES. W symulacjach wykorzystano funkcje benchmarkowe z konkursu CEC 2013, których dokładny opis znajduje się tutaj [8]. Ten zestaw został wybrany ze względu na duże zróżnicowanie funkcji oraz implementację w języku MATLAB, z której skorzystano [9]. Algorytm CMA-ES (dla każdego wariantu uwzględniania ograniczeń) uruchomiono po 25 razy dla każdej funkcji benchmarkowej (funkcje z 30 wymiarami). Algorytm CMA-ES dla każdej symulacji zwrócił najlepszy punkt, który znalazł. W ten sposób dla 28 funkcji uzyskano po 25 wartości. Wyniki zostały porównane testem Wilcooxona, aby sprawdzić, czy w istotny sposób się różnią.

Metoda konserwatywna

W tych testach nie znalazła się opisywana wcześniej metoda konserwatywna. Wynika to z faktu, że zakłada ona istnienie powiązań między punktami kolejnych iteracji. W algorytmie CMA-ES wpływ punktów na kolejną iterację jest pośredni i nie ma tutaj relacji rodzic-potomek.

Reinicjacja

W tej metodzie zmieniony został sposób reinitacji. Zgodnie z rozdziałem 4.1.3 cały punkt jest przenoszony do pozycji początkowej, gdy którakolwiek z granic jest przekroczena. Takie podejście w algorytmie CMA-ES sprawiało, że algorytm zatrzymywał się w pozycji początkowej, ponieważ losowanie bardzo często zwracało punkt poza ograniczeniami. Zdecydowano się na zmianę tej metody tak, że po przekroczeniu bariery reinitowane są tylko te wymiary, na których granica została przekroczena. Pozostałe wymiary zostały bez zmian. Obrazuje to poniższy pseudokod:

```
dla każdej współrzędnej i  
jeżeli ( $lb(i) > x(i)$ ) lub ( $ub(i) < x(i)$ )  
     $x(i) = xr(i)$ 
```

Ewolucja darwinowska

Do opisania pozostała wspomniana wcześniej ewolucja darwinowska. W tym wariantie punkt nie jest transformowany wewnątrz ograniczenia - jego współrzędne są niezmienne podczas transformacji. Transformacji podlega wartość funkcji. Jest ona taka, jak gdyby punkt został transformowany w ustalony sposób.

W poniższych testach użyto ewolucji darwinowskiej z rzutowaniem na ograniczenie.

Wyniki

Wyniki zostały zebrane w tabeli. Przedstawia ona porównanie wyników algorytmu CMA-ES z rzutowaniem (wariant podstawowy) do wyników algorytmu CMA-ES z innymi metodami uwzględniania ograniczeń. Wyniki zostały porównane za pomocą testu Wilcooxona. Kolumny są etykietowane numerami funkcji, natomiast wiersze wariantami uwzględniania ograniczeń. Oznaczenia symboli znajdujących się na przecięciach:

- + wariant podstawowy zwrócił statystycznie lepsze wyniki,
- - wariant podstawowy statystycznie gorsze wyniki,
- . różnice wyników wariantów były nieistotne statystycznie.

Numer funkcji	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Reinicjacja	+	-	-
Odbijanie	-
Zawijanie	+	.	.	.	-	-
Powtórna generacja	-	.	.	.	-	-
Darwinowska	+	+	+	+	+	+	+	.	+	+	+	+	+	+
Numer funkcji	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Reinicjacja	.	.	+	-	.	.	+	.	-	.
Odbijanie	-
Zawijanie	-	.	+	-	-	.	+	.	.	.
Powtórna generacja	-	.	+	-	-	.	.	.	-	.
Darwinowska	+	.	+	+	+	.	+	+	+	+	+	+	+	+

Powyższa tabela pokazuje, że wariant ewolucji darwinowskiej wypadł zdecydowanie gorzej. Nie należy się temu specjalnie dziwić, ponieważ w tej metodzie z perspektywy algorytmu dany wymiar poza ograniczeniem jest funkcją stałą.

W pozostałych przypadkach wersja bazowa wypadła słabiej. Ta różnica jest najmniej istotna przy reinitiacji (3 funkcje zwróciły lepsze wyniki dla bazowej, a 4 dla reinitiacji). Nieco tylko inaczej jest w przypadku zawijania (3 - bazowa, 5 - zawijanie). W kontekście tego wyniki odbijania, czyli tylko 2 funkcje zwróciły lepsze rezultaty, wydają się być dość marne. Należy jednak zauważyć, że nie otrzymano rezultatów, które byłyby na korzyść rzutowania.

Zdecydowanie najlepsze rezultaty zwróciła metoda powtórnej generacji - w 8 funkcjach zwróciła lepsze wyniki i tylko dla 1 gorsze.

Metody, które wypadły najlepiej (odbicie i powtórna generacja), sprawiają że średnie przesunięcie punktu nie jest znaczące - można to zauważyć na wykresach wektorów przesunięć w rozdziale 4. Można więc wywnioskować, że nie warto zbytnio ingerować w położenie punktu. Na tak postawiony wniosek można podać kontrprzykład z rzutowaniem, które wypadło gorzej. Warto jednak przypomnieć sobie rysunek 7, który pokazywał, że w rzutowaniu rozkład prawdopodobieństwa punktów jest bardzo za-

burzony, co prowadzi do mniejszej eksploracji przestrzeni. Z tego też względu można upatrywać słabszych wyników rzutowania.

Interesujący jest również fakt, że niektóre funkcje (9, 14, 17, 22) pokazują podobne zachowania. Takie wyniki pokazują, że nie istnieje jeden, najlepszy wariant dla każdej funkcji.

7. Podsumowanie

Celem pracy była weryfikacja wpływu uwzględniania ograniczeń kostkowych w algorytm CMA-ES. Cel ten został zrealizowany poprzez teoretyczne rozważania oraz przeprowadzenie testów.

7.1. Wnioski

Metoda uwzględnienia ograniczeń kostkowych wpływa na wyniki symulacji algorytmu CMA-ES. Ponadto połączenie pewnych funkcji i ograniczeń daje lepsze rezultaty. Z tych faktów wynika, że czasem warto zadać sobie trud nad wyborem metody uwzględniania ograniczeń, a nawet do tego samego uruchomić kilka symulacji z różnymi metodami.

Kolejnym wnioskiem jest to, że powtórna generacja daje lepsze rezultaty od rzu- towania w algorytmie CMA-ES. Należy jednak pamiętać, że powtórna generacja jest bardziej czasochłonne ze względu na konieczność ponownego losowania punktów. Może to być uciążliwe, gdy zależy na szybkości działania, a parametry początkowe sprawiają, że często losowane są punkty niedopuszczalne.

7.2. Możliwości rozwoju

Przedstawione rozważania mogą posłużyć jako baza do stworzenia nowych wersji algorytmu CMA-ES. Nowy algorytm mógłby w zależności od aktualnych parametrów (położenie wartości oczekiwanej rozkładu, długość kroku, itp.) dobierać odpowiednią metodę uwzględniania ograniczeń.

Literatura

- [1] *Wykłady z algorytmów ewolucyjnych*, Jarosław Arabas, 2004
- [2] *The Wilcoxon Signed-Rank Test*, Richard Lowry, <http://vassarstats.net/textbook/ch12a.html>
- [3] *Różnicowa implementacja algorytmu CMAES*, Michał Bobowski, 2015
- [4] *An Introduction to Probability Theory and its Applications, Volume II*, William Feller, 1970
- [5] *Completely Derandomized Self-Adaptation in Evolution Strategies w Evolutionary Computation*, 9(2), pp. 159-195, Nikolaus Hansen, Andreas Ostermeier, 2001
- [6] *CMA-ES: Evolution Strategy with Covariance Matrix Adaptation for nonlinear function minimization*, Nikolaus Hansen, https://www.lri.fr/~hansen/cmaes_inmatlab.html
- [7] *The CMA evolution strategy: A tutorial.* arXiv:1604.00772v1, Nikolaus Hansen, 2016
- [8] *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization*, J. J. Liang, B. Y. Qu, P. N. Suganthan, Alfredo G. Hernández-Díaz, 2013
- [9] *CEC13 Test Function Suite* , Jane Jing Liang, <http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC2013/cec13matlab.zip>
- [10] *Algorytmy ewolucyjne i ich zastosowania w Zeszyty naukowe*, 1/2006, pp. 81-92, Ewa Figielska, 2006
- [11] *Reducing the Space-Time Complexity of the CMA-ES*, James N. Knight, Monte Lunacek, 2007
- [12] *Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009*, N. Hansen, A. Auger, R. Ros, S. Finck, P. Posik, 2010

Warszawa, dnia

Oświadczenie

Oświadczam, że pracę magisterską pod tytułem „Analiza możliwości wykorzystania w algorytmie CMA-ES wiedzy o ograniczeniach kostkowych”, której promotorem jest dr hab. inż. Jarosław Arabas prof. nzw. PW, wykonałem samodzielnie, co poświadczam własnoręcznym podpisem.
