# Wykorzystanie ciągów Sobola do wyznaczania początkowej puli punktów algorytmów optymalizacji

Mariusz Kamola
Instytut Automatyki i Informatyki Stosowanej
Politechnika Warszawska
ul. Nowowiejska 15/19, 00-665 Warszawa
mkamola@ia.pw.edu.pl

W pracy zaproponowano ciągi Sobola jako alternatywny mechanizm generowania początkowej puli punktów dla algorytmów niedeterministycznych optymalizacji globalnej: CRS2 i ewolucyjnego. Przedstawione są wyniki testów praktycznych dla trzech rodzajów funkcji testowych. Zamieszczona jest propozycja procedury testującej, w której siatka punktów Sobola jest losowo przesuwana, aby wyeliminować wprowadzanie do populacji początkowej tych samych punktów. Testy wykazały dużą odporność obu algorytmów na sposób generowania puli początkowej.[*]

## 1. Motywacje

Uniwersalność, skuteczność i łatwość implementacji to cechy algorytmów niedeterministycznych optymalizacji globalnej, które przesądziły o ich silnej pozycji pomimo braku podstaw teoretycznych równie mocnych, jak dla algorytmów klasycznych. Być może właśnie ten brak kanonów skłania badaczy do modyfikowania pierwotnej procedury symulowanego wyżarzania czy algorytmu genetycznego i proponowania własnych usprawnień: inne kryterium zatrzymania, inne zasady selekcji, specyficzne operatory krzyżowania.

W całej różnorodności pomysłów zastanawia znikome zainteresowanie wpływem sposobu, w jaki powstaje początkowa pula punktów, na ogólną efektywność algorytmu. Pula początkowa powstaje w wyniku losowania za pomocą generatora liczb pseudolosowych, i jest to najczęściej generator najprostszy, zwykle dostarczany przez środowisko obliczeniowe, spełniający (albo nie) jedynie absolutne minimum. Takich generatorów należy się wystrzegać [1], zwłaszcza w zadaniach całkowania numerycznego. Można więc przypuszczać, że skoro sposób generacji zbioru punktów nie pozostaje obojętny dla zadań całkowania, to może mieć on wpływ również na efektywność algorytmów optymalizacji bazujących na puli punktów początkowych.

Proponowaną alternatywą dla prostego losowania kolejnych punktów początkowych są ciągi Sobola. Algorytm generowania kolejnych punktów, przedstawiony przez I. M. Sobola oraz R. B. Statnikowa w [2], a następnie usprawniony przez Antonowa i Saliejewa, działa bazując na $w$ liczbach naturalnych nazywanych liczbami kierunkowymi $D_i$, $i = 1,2...w$. Aby wygenerować kolejną liczbę $X_j$ ciągu, należy obliczyć iloraz różnicowy (XOR) liczb kierunkowych o tych indeksach $i$, dla których $j$ (zapisane w sposób binarny) ma na pozycji $i$-tej jedynkę. Ciąg utworzony w ten sposób pokrywa dziedzinę o wiele równomierniej niż punkty będące realizacją zmiennej losowej, a przy tym nie wymaga wcześniejszej znajomości liczby

[*] Praca została wykonana w ramach projektu badawczego CATID (Centrum Automatyki i Technik Informacyjno-Decyzyjnych) w latach 1999/2000.

punktów, które mają być wygenerowane (w odróżnieniu procedury od pokrywania $V$ siatką kartezjańską).

Wiele testów praktycznych wykazało, że choć ciągi Sobola nie są zawsze najlepszym sposobem generowania punktów [3], to wykazują zdecydowaną przewagę nad punktami, których współrzędne są realizacjami „zmiennej losowej" dostarczanej przez system.

Skoro doświadczenia z zastosowania ciągów Sobola w zadaniach całkowania numerycznego tak dobrze rokują ich użyciu w algorytmach optymalizacji, to warto sprawdzić ich działanie w praktyce.

## 2. Użyte algorytmy i funkcje testowe

Do testów wybrano dwa algorytmy optymalizacji i trzy testowe funkcje celu.

### *Algorytm CRS2*

Pełna nazwa algorytmu, przedstawionego w [4], to *Controlled Random Search*, wersja 2. Jest to procedura optymalizacji globalnej z poszukiwaniem bezpośrednim, nadająca się szerokiej klasy zadań. Poszukiwanie minimum funkcji $q(x),\ \dim(x)=n$ przebiega w sześciu krokach:

- **Krok 1.** Z dziedziny $V$ wylosuj $N$ punktów; niech stanowią one bieżący zbiór punktów. Oblicz wartość funkcji celu w każdym z punktów.
- **Krok 2.** W bieżącym zbiorze punktów znajdź punkty $x_{\mathbf{l}}$ i $x_{\mathbf{h}}$, dla których wartość funkcji celu jest odpowiednio najmniejsza i największa.
- **Krok 3.** Utwórz sympleks ($n+1$-elementowy podzbiór bieżącego zbioru punktów), w skład którego wejdzie $x_{\mathbf{l}}$, a pozostałe punkty zostaną wylosowane ze zbioru bieżącego. Wyznacz nowy punkt $x_{\mathbf{t}}$ odbijając $x_{\mathbf{h}}$ względem środka sympleksu.
- **Krok 4.** Jeśli $x_{\mathbf{t}} \in V$, to oblicz $q(x_{\mathbf{t}})$ i przejdź do kroku 5. Jeśli nie, przejdź do kroku 3.
- **Krok 5.** Jeśli $q(x_{\mathbf{t}}) < q(x_{\mathbf{h}})$, to w bieżącym zbiorze punktów zastąp $x_{\mathbf{h}}$ przez $x_{\mathbf{t}}$ i przejdź do kroku 6. Jeśli nie, przejdź do kroku 3.
- **Krok 6.** Zakończ, jeśli jest spełnione kryterium zatrzymania algorytmu. Jeśli nie, wróć do kroku 2.

  W testach przyjęto $N = 8(n+1)$, zgodnie z sugestiami w literaturze.

### *Algorytm ewolucyjny*

Spośród wielu wersji algorytmów ewolucyjnych [5] wybrano w sposób arbitralny jeden, który dobrze sprawdzał się we wcześniejszych zadaniach optymalizacji napotkanych przez autora. Oto zastosowany schemat:

- **Krok 1.** Z dziedziny $V$ wylosuj $N$ punktów stanowiących populację $P_0$. (Kolejne populacje będą oznaczane przez $P_1$, $P_2$ itd.)
- **Krok 2.** Oblicz wartość funkcji celu dla wszystkich osobników z bieżącej populacji $P_k$. Zakończ, jeśli kryterium zatrzymania jest spełnione. Jeśli nie, przejdź do kroku 3.
- **Krok 3.** Stwórz nową populację, $P_{k+1}$. Każdy jej osobnik jest wyłoniony albo wprost z turnieju[*] (z prawdopodobieństwem 0,3), albo jest krzyżówką 10 innych zwycięzców turnieju (zastosowano krzyżowanie uśredniające).
- **Krok 4.** Poddaj mutacji każdego osobnika z $P_{k+1}$, tj. zaburz każdą z jego współrzędnych zmienną losową z rozkładem Cauchy'ego.
- **Krok 5.** Uczyń $P_{k+1}$ populacją bieżącą i przejdź do kroku 2.

---

[*] Pojedynczy turniej przebiega w ten sposób, że spośród $P_k$ wybiera się losowo dwa osobniki, a zwycięża ten, dla którego wartość funkcji celu jest mniejsza.

Przyjęto, że populacja będzie liczyć 50 osobników, a rozkład Cauchy'ego będzie miał takie parametry, że średnio 5% osobników w wyniku mutacji znalazłoby się poza dziedziną (odpowiednie współrzędne takich osobników będą wówczas modyfikowane tak, by osobniki znalazły się w dziedzinie, na ograniczeniach).

### *Zmodyfikowana funkcja testowa Griewanka*

$$q(x) = 1 + \frac{1}{80}\sum_{i=1}^{n}\left(x_i^2\right) - \prod_{i=1}^{n}\cos\left(x_i(2-\cos i)\right)$$

Jest to paraboloida zsumowana z funkcją cosinus, co daje wielość minimów lokalnych i praktycznie uniemożliwia rozwiązanie zadania metodami klasycznymi. Modyfikacje w stosunku do postaci oryginalnej polegają na dodaniu czynnika $2-\cos i$, który urozmaica zagęszczenie minimów w zależności od wymiaru. Funkcja osiąga wartość minimalną $0$ dla $x = \mathbf{0}$ (dla dowolnego $n$). W badaniach przyjęto $n = 4$.

### *Funkcja testowa Shekela*

Jest to jedna z czterowymiarowych funkcji testowych opisanych w [6], nazwana SQRN7:

$$q(x) = -\sum_{j=1}^{7}\left(\sum_{i=1}^{4}\left(x_i - a_{ij}\right)^2 + c_j\right)^{-1},$$

$$c = \begin{bmatrix}0{,}1 & 0{,}2 & 0{,}2 & 0{,}4 & 0{,}6 & 0{,}6 & 0{,}3\end{bmatrix}^{\mathrm{T}}, \qquad a = \begin{bmatrix}4 & 1 & 8 & 6 & 3 & 2 & 5\\ 4 & 1 & 8 & 6 & 7 & 9 & 5\\ 4 & 1 & 8 & 6 & 3 & 2 & 3\\ 4 & 1 & 8 & 6 & 7 & 9 & 3\end{bmatrix}$$

Funkcja osiąga wartość minimalną $-10{,}4028$ dla $x = \begin{bmatrix}4 & 4 & 4 & 4\end{bmatrix}^{\mathrm{T}}$.

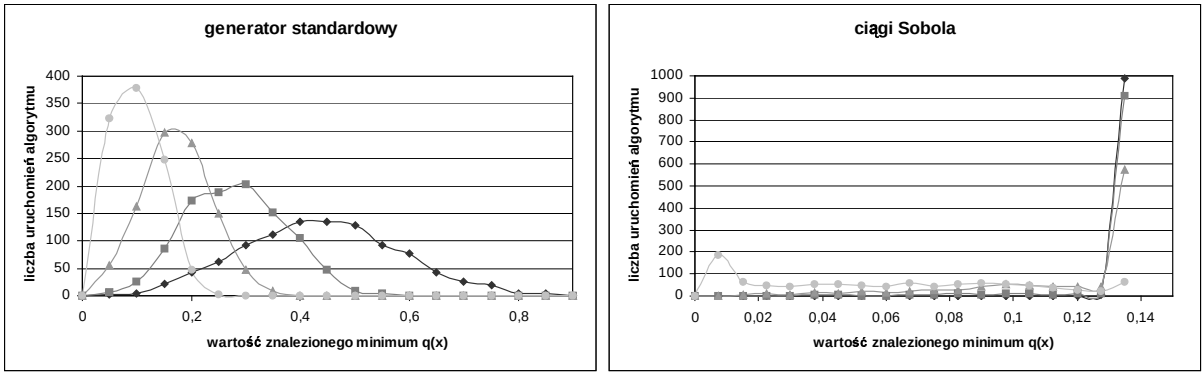
### *Funkcja o wartościach wyznaczanych w drodze symulacji*

Funkcja ta została zaczerpnięta z pewnego zadania poszukiwania optymalnego punktu pracy instalacji przemysłowej, której model został zaimplementowany w postaci programu symulacyjnego. Wyznaczenie $q(x)$ oznacza więc uruchomienie owego symulatora z $x$ jako danymi wejściowymi. Jej cechą charakterystyczną, przy stosunkowo prostym kształcie hiperpowierzchni, jest duża liczba ograniczeń implikowanych, które dodatkowo zawężają dziedzinę, i to w sposób nieprzewidywalny. Praca z funkcją o takim charakterze wymagała pewnych modyfikacji opisywanych algorytmów, lecz nie ma to wpływu na istotę opisywanych badań.

Wartość minimalna funkcji to $1{,}646$. Wymiar zadania: $n = 9$.

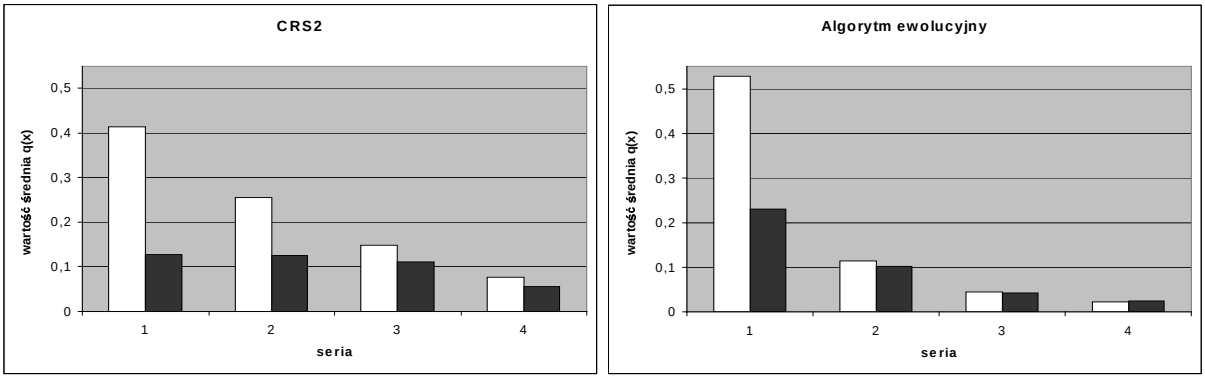
## 3. Wyniki wstępne

Jako kryterium zatrzymania, wspólne dla obu algorytmów i wszystkich funkcji celu, przyjęto liczbę wykonanych oszacowań funkcji celu. Nie pozwala to, co prawda, czynić porównania algorytmów pod względem wydajności — ale nie ono jest celem badań. Interesującym nas wskaźnikiem jest (domniemana) poprawa znajdowanego minimum $q(x)$ uzyskiwana przy zmianie procedury generowania początkowej puli punktów z klasycznej (z systemowego generatora liczb pseudolosowych) na alternatywną (wykorzystującą ciągi Sobola).



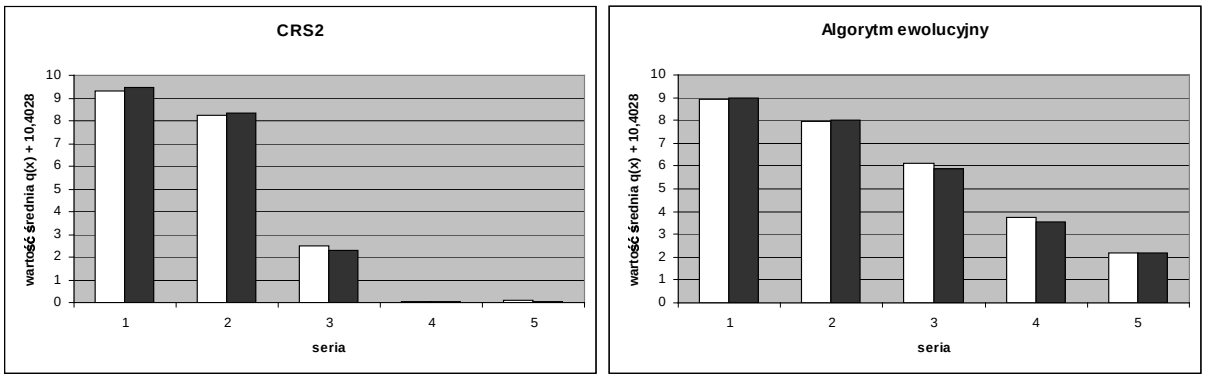

**Rys. 1.** Histogramy minimów funkcji Griewanka znalezionych przez algorytm CRS2 przy standardowej (z lewej) i alternatywnej (z prawej) metodzie generowania puli początkowej. Próba polegała na 1000-krotnym uruchomieniu algorytmu. Wykresom oznaczonym symbolami ◆, ■, ▲ i ● odpowiadają próby dla maksymalnie 100, 300, 1000 i 3000 oszacowań $q(x)$.

Dla każdej z funkcji celu wykonano serię uruchomień każdego z algorytmów. Warto rozpocząć analizę wyników od przyjrzenia się wartościom znalezionym w kilku wybranych seriach. Z rys. 1 wynika, że w puli utworzonej z ciągów Sobola może już na początku znaleźć się bardzo dobry punkt, który jest poprawiany dopiero przy znacznie większym limicie wywołań $q(x)$. Taki traf zawsze będzie deklasował optymalizacje uruchamiane z pulą uzyskaną w sposób standardowy — a nie jest to sytuacja rzadka, zwłaszcza, gdy rozpatruje się przykłady akademickie, w których minimum leży w samym środku dziedziny — czyli tam, gdzie ciąg Sobola umieszcza jeden z pierwszych swoich punktów.

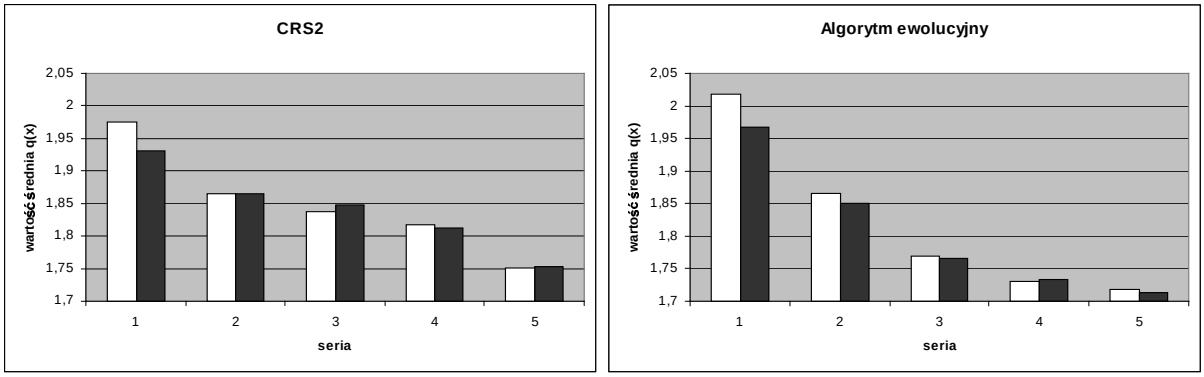



**Rys. 2.** Zestawienie wartości średnich wyników z minimalizacji funkcji Griewanka algorytmem CRS2 (z lewej) i ewolucyjnym (z prawej). Białe słupki odpowiadają uruchomieniom z pulą początkową wylosowaną w sposób standardowy; czarne — przy wykorzystaniu ciągów Sobola. Numery 1, 2, 3 i 4 odpowiadają seriom z limitem 100, 300, 1000 i 3000 oszacowań.

Zestawienie na rys. 2 uśrednionych wyników dla obu algorytmów potwierdza poprzednie wnioski. Start z pulą wygenerowaną przez ciąg Sobola daje przewagę, ale traci się ją w miarę wzrostu limitu oszacowań $q(x)$, przy czym algorytm ewolucyjny szybciej niż CRS2 niweluje dysproporcje. Oznacza to, że jest mniej podatny na metodę generacji puli początkowej.

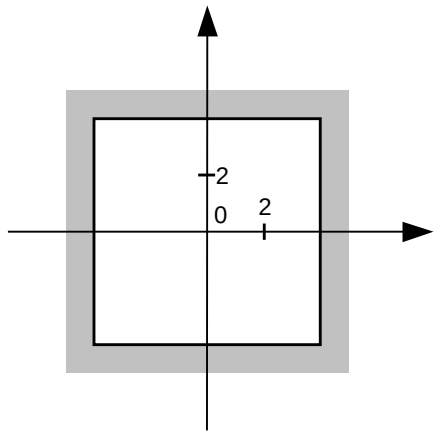



**Rys. 3.** Funkcja Shekela minimalizowana oboma algorytmami — wartości średnie. Kolejne serie uruchamiane dla limitów 100, 300, 1000, 3000 i 10000 oszacowań. Znaczenie kolorów słupków jak na rys. 2.




**Rys 4.** Zestawienie wartości średnich z minimalizacji funkcji trzeciego typu (połączonej z symulatorem instalacji przemysłowej). Kolejne serie uruchamiane dla limitów 300, 1000, 3000, 10000 i 30000 oszacowań. Znaczenie kolorów słupków jak na rys. 2. Ze względu na czasochłonność obliczeń w tym przypadku serie liczyły jedynie po 20 optymalizacji, więc wyniki mają charakter orientacyjny.

Korzyści ze stosowania ciągów Sobola przestają być oczywiste, kiedy spojrzeć eksperymenty z pozostałymi dwiema funkcjami testowymi (rys. 3 i 4). Uzyskane wartości średnie praktycznie się nie różnią, zatem optymistyczne wyniki dla funkcji Griewanka to jedynie sprzyjający zbieg okoliczności.

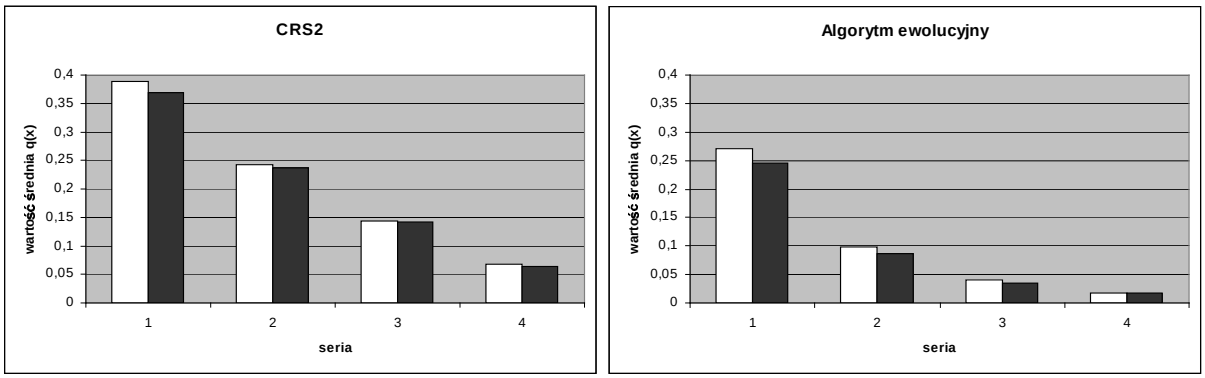



**Rys. 5.** Dziedzina $V$ (kwadrat z czarnym obrysem) na tle marginesów (szarych) swojego dopuszczalnego położenia.

## 4. Proponowane modyfikacje procedur testujących

Jak zatem obiektywnie stwierdzić, czy używanie ciągów Sobola prowadzi do jakichkolwiek korzyści w optymalizacji? Zaproponowane rozwiązanie polega na losowym zaburzaniu dziedziny $V$ tak, aby nie zmienić jej miary, ale położenie. Jeśli założyć, że dziedzina jest hiperkostką, to w każdym z wymiarów należy górny i dolny limit zmienić o losowo wybraną liczbę. Na przykład, rys. 5 przedstawia dziedzinę $V = \langle -4;4 \rangle \times \langle -4;4 \rangle$, która może „przemieścić się” w kierunku każdego z wersorów o wartość od $-1$ do 1.

Dla tak zdefiniowanej dziedziny powtórzono serie testów, a wyniki przedstawiono na rys. 6. Wprawdzie rozwiązania uzyskiwane przy starcie z użyciem ciągów Sobola wciąż mają przewagę, ale jest ona znikoma — około 10% (por. rys. 2). Pocieszające, że przewaga ta jest w przybliżeniu stała dla wszystkich serii.




**Rys. 6.** Zestawienie wartości średnich wyników z minimalizacji funkcji Griewanka przy losowym zaburzaniu $V$. Oznaczenia analogiczne do rys. 2.

## 5. Wnioski

Posłużenie się ciągami Sobola jako alternatywnym mechanizmem generowania początkowej puli punktów dla przebadanych algorytmów nie przyniosło znaczących korzyści. Procedura weryfikacji wyników, zastosowana dla najlepiej rokującego przykładu (tj. funkcji Griewanka), jedynie ujawniła, że walory ciągów Sobola są w tym zastosowaniu pozorne. Ewentualne korzyści pochodzą stąd, że jeden z pierwszych wyrazów ciągu Sobola wypada dokładnie w środku dziedziny (gdzie ma zwyczaj znajdować się minimum większości funkcji testowych) — ale nie ma to znaczenia w większości zastosowań praktycznych.

Niemniej pocieszające jest, że nawet po „rozmyciu” dziedziny (por. rys. 6) zastosowanie ciągów Sobola korzystnie, chociaż nieznacznie, wpływa na efektywność algorytmu. Tym samym pomysł I. M. Sobola zachowuje swoje walory także w zagadnieniach optymalizacji — ale tylko do tego stopnia, jaki stanowi nakład poniesiony na wyznaczenie populacji początkowej względem całkowitego nakładu na rozwiązanie zadania. Ponadto nie stwierdzono, by po zastosowaniu ciągów Sobola dla którejkolwiek z badanych funkcji nastąpiło pogorszenie średniego uzyskiwanego wyniku. Zatem metodę alternatywną generacji puli początkowej można stosować bez obaw — przeciwnie, z nadzieją na poprawę efektywności w przypadkach niektórych zadań optymalizacji.

## 6. Podziękowania

Dziękuję doktorowi Jarosławowi Arabasowi za motywowanie i cenne uwagi na wszystkich etapach prowadzonych przeze mnie badań.

## 7. Bibliografia

1. William H. Press, Saul A. Teukolsky, William T. Vetterling i Brian P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1992.
2. I. M. Sobol. Distribution of points in a cube and the approximate evaluation of integrals. *USSR Comput. Math. and Math. Phys.* 7(2):784-802, 1967.
3. C. Lemieux i P. L'Ecuyer, A comparison of Monte Carlo, lattice rules and other low-discrepancy point sets. *Monte Carlo and Quasi-Monte Carlo methods 1998*. Springer, 2000. http://www.iro.umontreal.ca/~lecuyer/myftp/papers/mcqmc98.ps

4. W. L. Price. Global optimization algorithms for a CAD workstation . *Journal of Optimization Theory and Applications*, 55(1):133-146, 1987.
5. Thomas Bäck, David B. Fogel i Zbigniew Michalewicz. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, 1997.
6. Zbigniew Michalewicz. *Algorytmy genetyczne + struktury danych = programy ewolucyjne*. Wydawnictwa Naukowo-Techniczne, 1996.