AUTOMATYKA • 2009 • Tom 13 • Zeszyt 3

Patryk Orzechowski*

# Metoda deterioracji funkcji celu dla algorytmów poszukiwań ewolucyjnych z miękką selekcją

## 1. Wprowadzenie

Metodami optymalizacji określa się sposoby analizy problemu minimalizacji (lub maksymalizacji) rzeczywistej funkcji poprzez systematyczny dobór coraz to lepszych wartości ze zbioru punktów dopuszczalnych w celu odnalezienia rozwiązania optymalnego. W klasycznych algorytmach optymalizacyjnych [1, 9], do dalszej analizy dopuszczane są wyłącznie najlepsze spośród dotychczas znalezionych punktów. Dlatego też algorytmy te nazywa się także metodami *ścisłej* lub *twardej selekcji* [13] W praktycznych zastosowaniach takich, jak choćby procesy biologicznej ewolucji czy konkurencja rynkowa, oczekuje się możliwości wyboru innego typu – nie tylko najlepszego spośród dotychczasowego rozwiązania, ale także dopuszczenia wyboru rozwiązania suboptymalnego, oczywiście z pewnym mniejszym prawdopodobieństwem.

Koncepcja wielu algorytmów ma swoje źródła niejednokrotnie w obserwacji procesów zachodzących w środowisku naturalnym. W ten właśnie sposób powstały różnorodne techniki optymalizacyjne, jak choćby algorytmy mrówkowe [7, 8], algorytmy symulujące stadne zachowanie zwierząt (czyli tzw. algorytmy rojowe) [3, 16] czy też metody bazujące na sztucznych systemach immunologicznych [4, 5].

Jednym z przykładów zastosowań procesów biologicznych w matematyce są metody *selekcji miękkiej*, które poza najlepszymi rozwiązaniami biorą również pod uwagę rozwiązania nieco gorsze [13]. Algorytmy te mają swoją genezę w ewolucji gatunków, w której poza najlepiej dostosowanymi osobnikami do aktualnie panujących warunków w środowisku, wielokrotnie przeżywały osobniki słabiej przystosowane. Ich obecność dawała możliwość na wykształcenie wśród potomków niespotykanych cech osobniczych w historii całej populacji, które częstokroć pozwalały na skuteczniejsze dostosowanie się do zmiennych warunków środowiskowych. Dowodzi to fundamentalnego znaczenia różnorodności cech

* Katedra Automatyki, Akademia Górniczo-Hutnicza w Krakowie

wśród osobników dla przebiegu procesu ewolucji. Uwzględnienie w procesie selekcji nie tylko najlepszych rozwiązań jest więc biologicznie uzasadnione.

Skuteczność wykorzystania metod twardej selekcji w klasycznych zagadnieniach optymalizacji globalnej została postawiona pod znakiem zapytania w procesie poszukiwania rozwiązania optymalnego funkcji multimodalnej [19]. Klasyczne metody optymalizacji średnio radziły sobie z przekraczaniem siodeł zlokalizowanych pomiędzy wzgórzami funkcji. Wada ta nie ujawniła się w przypadku algorytmów selekcji miękkiej.

Wśród grupy metod *miękkiej selekcji* wyróżnić można następujące klasy algorytmów [19]:

- programowanie ewolucyjne (*Evolutionary Programming* EP) [10, 11],
- strategie ewolucyjne (Evolutionary Strategies ES) [22, 23],
- algorytmy genetyczne (*Genetic Algorithms* GA) [14, 15, 18],
- symulowane wyżarzanie (*Simulated Annealing* SA) [6, 17],
- algorytmy poszukiwań ewolucyjnych z miękką selekcją (*Evolutionary Search with Soft Selection*, ESSS) [13], będący wersją (μ,μ)-ES strategii ewolucyjnej [2].

W niniejszym artykule zaprezentowano pewien wariant algorytmu poszukiwań ewolucyjnych opartego o miękką selekcję, polegający na stopniowym obniżaniu funkcji celu. W następnym rozdziale przybliżono ogólne założenia oraz schemat algorytmów wykorzystujących miękką selekcję, w kolejnym opis koncepcyjny oraz implementacyjny metody degeneracji funkcji *fitness*. Dalsze rozdziały zawierają opis poszczególnych składowych algorytmu oraz wyniki wstępnych eksperymentów polegających na celowej deterioracji wartości funkcji celu, mających w zamierzeniu „wypędzenie" populacji z ekstremów lokalnych.

## 2. Właściwości algorytmów z miękką selekcją

W algorytmie poszukiwań ewolucyjnych z miękką selekcją [12, 20] zakłada się istnienie środowiska o ograniczonej pojemności oraz istnienie w nim reprodukującej się populacji o cechach zakodowanych w fenotypie. Każdemu osobnikowi przypisany jest wskaźnik jakości, czyli tzw. *fitness*. Zdolność populacji do reprodukcji jest wystarczająca, by w każdej iteracji zapełnić całą pojemność środowiska (tzn. wszystkie dostępne miejsca). Potomstwo nieznacznie różni się od przodków, ze względu na występowanie modyfikacji otoczenia – co także przekłada się na wartość ich *fitness*. Ewolucja odbywa się losowo – na podstawie stochastycznej selekcji i modyfikacji. Prawo selekcji odbywa się zgodnie z zasadą miękkiej selekcji i zakłada, że każdy osobnik ma szansę zajęcia dowolnego miejsca w środowisku (z tym większym prawdopodobieństwem, im większą posiada wartość funkcji *fitness*); prawo modyfikacji – przyjmuje, że szansa na zaistnienie modyfikacji zależy od wziętych pod uwagę osobników i wartości losowych.

Schemat grupy algorytmów ewolucyjnych z miękką selekcją zaproponowany przez [19], a szczegółowo opisany w [21] przedstawiono w formie schematycznej na rysunku 1. Algorytm w każdej iteracji, jeśli spełniony jest wariant testu pułapki, wykonuje algorytm

degeneracji krajobrazu. Następnie, przeprowadzana jest selekcja (zgodnie z założeniami, metodą ruletki), zaś przy zadanej parametrycznie szansie na wystąpienie mutacji – poddawana jest modyfikacji.


**Rys. 1.** Schemat algorytmu ESSS-DOF. Kolorem szarym zaznaczono wprowadzone modyfikacje w porównaniu z podstawową wersją algorytmu z miękką selekcją. Schemat na podstawie opisu umieszczonego w [21]

Klasyczną wersję algorytmu przedstawiono na białym tle (*Evolutionary Search with Soft Selection*) [21], ciemniejszym zaś oznaczono jedno z rozszerzeń algorytmu, które polega na degeneracji funkcji celu (*Deterioration of the Objective Function*; ESSS-DOF). Prezentowany w rozdziale 3 lgorytm jest wariantem metody degeneracji funkcji fitness zaproponowanym w pracy [19], w której deterioracja polega na wyznaczeniu aktualnej wartości *fitness* poprzez jej złożenie z funkcją Gaussa.

## 3. Metoda deterioracji funkcji celu

Koncepcja algorytmu degeneracji funkcji celu bazuje na spostrzeżeniu, że ekstremum ciągłej funkcji można przybliżać za pomocą kombinacji liniowej wielowymiarowych funkcji Gaussa. Odpowiednia aproksymacja centrów funkcji składowych miałaby szanse lepiej odzwierciedlać rozkład populacji w przestrzeni niż deterioracja krajobrazu za pomocą funkcji Gaussa o centrum w wartości oczekiwanej populacji zastosowana w [19].

Drugim spostrzeżeniem, leżącym u podstaw algorytmu, jest fakt, iż biorąc pod uwagę kilka kryteriów klasyfikacji osobników, najłatwiej podzielić osobniki na grupy względem tej cechy, która najbardziej populację rozróżnia. Wyznaczając średnią wartość poszczególnych cech i stwierdzając względem której populacja wykazuje największe rozproszenie (tzn. w którym wymiarze odchylenie standardowe w populacji jest największe), można dokonywać podziału na grupy osobników o wyższej i niższej wartości danej cechy. Podział może być dokonywany rekurencyjnie, w zależności od pożądanej dokładności obliczeń (ustalanej jako najmniejsza akceptowalna wartość odchylenia standardowego).

Na podstawie obydwu spostrzeżeń, opracowano następujący algorytm, złożony z dwóch części: nadrzędnej i podrzędnej.

**Algorytm nadrzędny** (degenerujący funkcji celu) składa się z następujących kroków:

1. Wykonaj algorytm podrzędny *crunch* (dla całej populacji), w celu wyznaczenia centrów wielowymiarowej funkcji Gaussa. Dla każdego centrum zdefiniuj funkcję Gaussa wg wzoru (6):

$$f(c) = \frac{1}{(2\pi)^{n/2}\left|M\right|^{1/2}} e^{-\frac{1}{2}(x-c)M^{-1}(x-c)} \tag{6}$$

gdzie:

$c$ – oznaczono współrzędne środka funkcji Gaussa,

$M$ – oznaczono półdodatnio określoną macierz kowariancji zmiennych losowych $X_1, \ldots X_n$, określoną wzorem (7):

$$M = \begin{bmatrix} \sigma_1^2 & \ldots & \operatorname{cov}(x_1, x_n) \\ \ldots & & \ldots \\ \operatorname{cov}(x_1, x_n) & \ldots & \sigma_n^2 \end{bmatrix} \tag{7}$$

2. Na podstawie wyznaczonych centrów oraz współrzędnych i wartości *fitness* osobników populacji, rozwiąż układ liniowy (8) metodą Simplex [24] w celu wyznaczenia współczynników kombinacji liniowej poszczególnych funkcji Gaussa. W układzie (8) przyjęto założenie, że suma wartości funkcji *fitness* dla kombinacji liniowej „wygryzających" funkcji w każdym analizowanym punkcie nie może być większa niż aktualna wartość funkcji *fitness* w tym punkcie. Równocześnie zakłada się, że funkcja *fitness* nie może być ujemna.
   Dla poszczególnych chromosomów $j = 1...m$ rozwiąż układ równań (8) w celu wyznaczenia $\alpha_i$:

$$\begin{cases} \sum_{i=1}^{m} \alpha_i f(x_j, c_i) \le F(x_j) \\ f(x_j, c_i) \ge 0 \end{cases} \tag{8}$$

   gdzie:

   $c_i$ – oznaczono środek funkcji Gaussa,
   $\alpha_i$ – współczynnik kombinacji liniowej dla funkcji Gaussa o środku w $c_i$,
   $f(x_i, c_i)$ – wartość funkcji Gaussa o centrum w $c_i$ dla punktu $x_i$,
   $F(x_i)$ – aktualna wartość funkcji fitness dla osobnika $x_i$.

   W rozwiązaniu przyjęto niezależność poszczególnych wymiarów; w macierzy kowariancji odchylenia standardowe wyznaczono na podstawie wariancji listy osobników aproksymowanych (lub uznano za jednostkową, jeśli w funkcji analizowany był pojedynczy osobnik)
3. Wyznacz wartość nowej funkcji *fitness* dla wszystkich osobników: dla każdego osobnika od aktualnej funkcji *fitness* odejmij poszczególne funkcje Gaussa pomnożone przez współczynniki $\alpha_i$.
4. Zastąp bieżącą funkcję *fitness* nową funkcją.

**Algorytm podrzędny** (wyznaczający centra funkcji Gaussa) ma następującą postać:

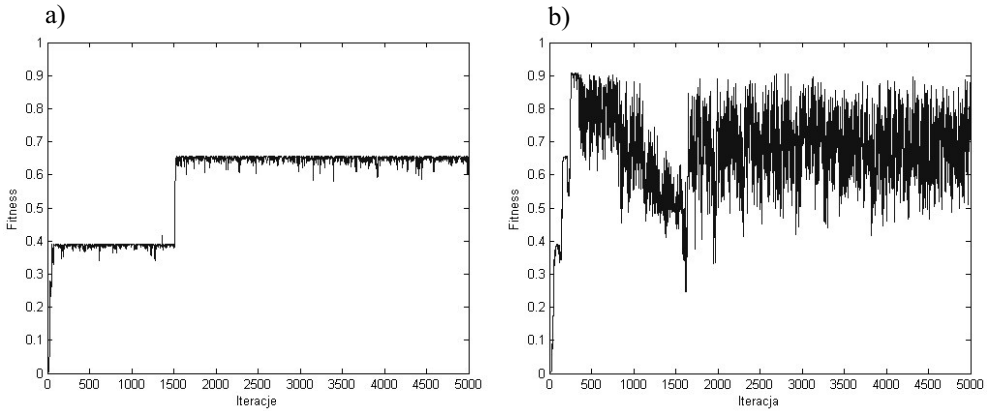
Funkcja *crunch* (lista osobników)

1. Wyznacz wartość oczekiwaną oraz wariancję dla populacji.
2. Jeśli lista osobników zawiera tylko pojedynczy punkt (lub wiele kopii tego samego punktu), zwróć ten element jako centrum.
3. Wyznacz kierunek o największej wariancji. Jeśli największa wariancja jest mniejsza niż określony parametr minimalnego poziomu, aproksymuj osobniki za pomocą fantomu osobnika umiejscowionego w wartości oczekiwanej próbki.
4. Dokonaj podziału osobników na te lepsze (A) i gorsze (B) od wartości oczekiwanej dla kierunku największej wariancji.
5. Wywołaj funkcję *crunch* (*A*) oraz *crunch* (*B*), połącz poszczególne wyniki w jedną listę C.
6. Zwróć listę C.

## 4. Wyniki

Do analizy działania algorytmu wybrano funkcję składającą się z sumy trzech funkcji Gaussa o różnej wysokości (por. rys. 3a). Populację zainicjowano w narożniku, w którym nie znajdował się żaden wierzchołek. Przyjęto, że podczas mutacji wylosowany osobnik zmieni swoje położenie zgodnie z (6), jednak nie więcej niż o 2% wykresu. Ustalono selekcję zgodną z metodą ruletki. Przyjęto następujący test pułapki: sprawdzano czy w 10 ostatnich iteracjach wartość funkcji fitness dla najlepszego osobnika nie zmieniła się więcej niż o 5%. Jeśli zmiana była mniejsza, stosowano degenerację krajobrazu. W trakcie symulacji obserwowano najlepszego osobnika w danej iteracji.

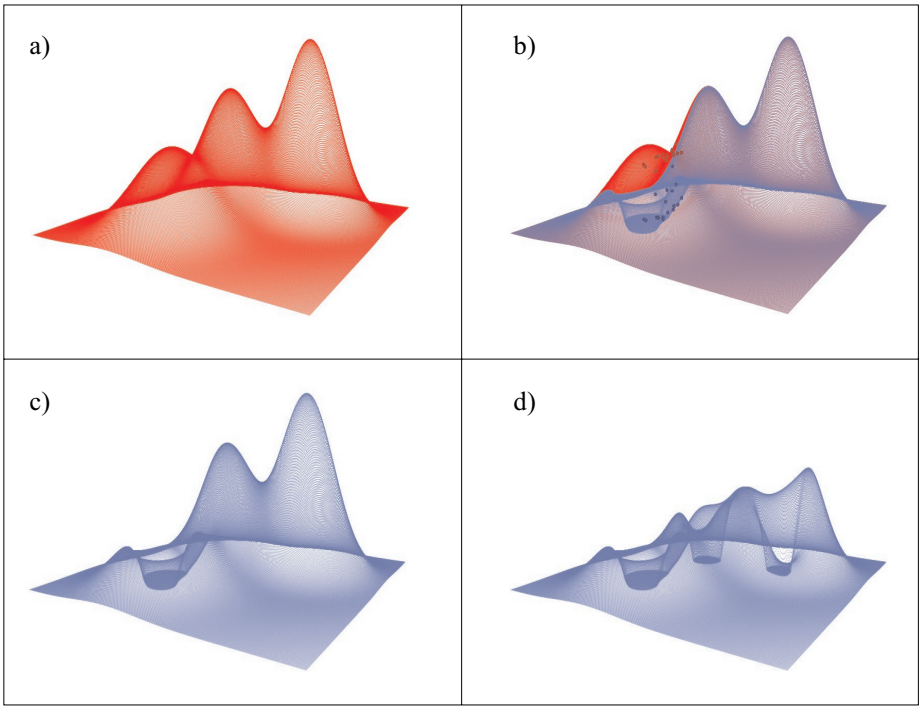
Wyniki i przebieg symulacji dla populacji złożonej z 20 osobników zaprezentowano na rysunku 2 – gdzie porównano zachowanie programu przy braku zastosowania algorytmu degeneracji (a) oraz z zastosowaniem (b). Zwrócić uwagę należy na to, iż bez deterioracji funkcji celu, populacja nie osiągnęła najwyższego wierzchołka w przeciągu 5000 iteracji.




**Rys. 2.** Wartość funkcji fitness najlepszego osobnika w danej iteracji algorytmu: a) bez deterioracji funkcji *fitness*; b) z deterioracją funkcji *fitness*

Na rysunku 3 przedstawiono wyniki degeneracji funkcji *fitness* dla sumy trzech funkcji Gaussa (a): zestawienie wykresów (b) oraz degenerację poszczególnych wierzchołków: (c) i (d).

Przykład przedstawiony na rysunku 3d tłumaczy dość dziwne chaotyczne wyznaczanie najlepszego osobnika w populacji. Niedokładność wyznaczenia optimum oraz silne drgania wynikają z deterioracji najwyższego wzgórza, co sprawia, że algorytm zaczyna nieco „błądzić" po ścianach wzniesienia. Dlatego też, pomimo bardzo szybkiego osiągnięcia najwyższego wzgórza przez metodę „wygryzającą wzgórze", po degeneracji wzgórza algorytm gubi się – odnalezienie optymalnego rozwiązania z dużą dokładnością staje się niemożliwe.




**Rys. 3.** Wyniki działania algorytmu: a) początkowa wartość funkcji *fitness*; b) kolor niebieski – funkcja *fitness* obniżona w wyniku zastosowaniu algorytmu na tle funkcji oryginalnej. Punkty w odpowiednich kolorach odpowiadają osobnikom przykładowej populacji przed i po deterioracji; c) i d) wyniki deterioracji najniższego i najwyższego wzgórza

## 5. Podsumowanie

W pracy zaprezentowano autorski wariant algorytmu ESS-DOF, dokonujący deterioracji funkcji celu za pomocą kombinacji liniowej funkcji Gaussa. Wyniki eksperymentalne potwierdzają, że algorytm może być z powodzeniem stosowany w celu wspomagania pokonywania przez populację siodeł funkcji multimodalnych. Dalszych prac wymaga analiza skuteczności algorytmu podziału (*crunch*), jak również warunków testu pułapki.

Wyższa skuteczność algorytmu miękkiej selekcji w rozwiązywaniu problemów siodłowych opisana w [13] została eksperymentalnie potwierdzona. Przy przyjętych parametrach, algorytm ewolucyjny bez funkcji deterioracji krajobrazu w 5000 iteracji nie poradził sobie z odnalezieniem ekstremum globalnego. Dopiero zastosowanie metody degeneracji funkcji krajobrazu pozwoliło algorytmowi na opuszczenie pułapki lokalnego optimum i dalszą optymalizację funkcji celu. Należy zwrócić uwagę na szybkość, z jaką algorytm doszedł do poziomu najwyższego wzniesienia – zaprezentowana metoda potrafiła skutecznie uporać się z obniżeniem poszczególnych wzgórz. Deterioracja funkcji jest niewątpliwą zaletą

przedstawionego algorytmu, lecz również jego przekleństwem, gdyż obniżaniu podlega zostaje optimum globalne. Algorytm stara się wówczas znaleźć najlepsze z dostępnych rozwiązań, co przekłada się na niższą jakość wyznaczanego rozwiązania.

Najistotniejszym czynnikiem determinującym działanie algorytmu jest niewątpliwie topologia funkcji celu. Celem dalszych badań jest przetestowanie działania algorytmu dla różnorodnych funkcji wielowymiarowych.

Skuteczność algorytmu miękkiej selekcji zależy dodatkowo od liczebności populacji punktów przeszukujących. Mniejsza liczebność populacji zmniejsza czas potrzebny na obliczenia algorytmu oraz charakteryzuje się wysoką zdolnością do przekraczania siodeł, z drugiej strony otrzymane rozwiązania będą mniej dokładne oraz mniej stabilne.

Na szybkość dochodzenia do optimum mają wpływ również odchylenia standardowe punktów w danej iteracji, co potwierdza kolejną tezę postawioną w [19]. Małe odchylenia powodują większe skupienie populacji, co spowalnia poszukiwania optimum (za to pozwala na uzyskanie dokładniejszego przybliżenia); zaś duże – generują rozproszenie populacji, co z kolei skutkować może w chaotycznym i niezbyt dokładnym przeszukiwaniu funkcji (z drugiej strony zwiększoną łatwością pokonywania siodeł).

## Literatura


[1] Ashlock D., *Evolutionary Computation for Modeling and Optimization*. Springer Verlag, 2006.

[2] Bäck T., Schwefel H.-P., *An Overview of Ewolutionary Algorithms for Parameter Optimisation*. Evolutionary Computation, vol. 1, 1993, 1–23.

[3] Bonabeau E., Dorigo M., Theraulaz G., *Swarm Intelligence: From Natural to Artificial Systems*. Oxford Univ. Press, NY, 1999.

[4] Byrski A., Kisiel-Dorohinicki M., *Agent-Based Evolutionary and Immunological Optimization*. [w:] Proceedings of the 7th international Conference on Computational Science – ICCS, 2007.

[5] Cutello V., Nicosia G., *An Immunological Approach to Combinatorial Optimization Problems.* Lecture Notes in Computer Science, Springer, vol. 2527, 2002, 361–370.

[6] Davis L. (Ed.), *Genetic algorithms and simulated annealing.* Research Notes in Artificial Intelligence, London, Pitman, 1987.

[7] Dorigo M., StützleT., *Ant Colony Optimization*. MIT Press, Cambridge, 2004.

[8] Dorigo M., Maniezzo V., Colorni A., *Ant System: Optimization by a Colony of Cooperating Agents*. IEEE Transactions on Systems, Man, and Cybernetics–Part B, 26 (1), 1996, 29–41.

[9] Findeisen W., Szymanowski J., Wierzbicki A., *Teoria i metody obliczeniowe optymalizacji*. Wyd. 2, PWN, Warszawa, 1980.

[10] Fogel L.J., Owens A.J., Walsh M.J., *Artficial Intelligence through Simulated Evolution*. Wiley, New York, 1966.

[11] Fogel D.B., *Evolutionary Computation*: *Toward a New Philosophy of Machine Intelligence*. IEEE Press Series on Computational Intelligence, Wiley-IEEE Press, 2006.

[12] Galar R., *Handicapped individua in evolutionary processes*. Biol. Cybern., 51, 1985, 1–9.

[13] Galar R., *Evolutionary search with soft selection*. Biological Cybernetics, 60, 1989, 357–364.

[14] Goldberg D.E., *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley, Reading, MA, 1989.

[15] Holland J.H., *Adaptation in natural and artificial systems.* Ann Arbor: The University of Michigan Press, 1975.

[16] Kennedy J., Eberhart R.C., Shi Y., *Swarm intelligence*. Morgan Kaufmann Publishers, San Francisco, 2001.
[17] Kirkpatrick S., Gellat C.D., Vecchi M.P., *Optimisation by simulated annealing*. Science 220, 1983, 671–680.
[18] Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, 3rd ed., 1996.
[19] Obuchowicz A., Patan K., *O pewnych wariantach algorytmu poszukiwań ewolucyjnych z miękką selekcją*. „Algorytmy ewolucyjne i optymalizacja globalna", Materiały II krajowej konferencji, Rytro 1997, 193–200.
[20] Obuchowicz A., *The evolutionary search with soft selection and Deterioration of the Objective Function*. Intelligent Information Systems VI: Proceedings of the workshop, Zakopane, IPI PAN, 1997, 288–295.
[21] Obuchowicz A., *Evolutionary algorithms for global optimization and dynamic system diagnosis*. Zielona Góra, Lubusky Scientific Society, 2003.
[22] Rechenberg, I., *Cybernetic solution path of an experimental problem*. Roy. Aircr. Establ., libr., 1965.
[23] Schwefel H.P., *Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik*. Technical University of Berlin, 1965 (Diploma thesis).
[24] Wojda P., *Wykłady z programowania liniowego*. Wydz. Matematyki Stos. AGH, http://wms.mat.agh.edu.pl/~wojda/Prog_lin/Pl2.