GWC

Dominika Sarkowicz

2018-03-15

Abstract

...

Contents

1 Grey Wolf Optimizer **2**
1.1 Inspiracja . . . 2
1.2 Model matematyczny . . . 2
1.2.1 Hierarchia . . . 2
1.2.2 Otoczenie ofiary . . . 2
1.2.3 Polowanie . . . 3
1.2.4 Atakowanie ofiary (eksploatacja) . . . 3
1.2.5 Poszukiwanie ofiary (eksploracja) . . . 4
1.3 Algorytm . . . 4

2 Warianty GWO **5**
2.1 Zmodyfikowane wersje GWO . . . 5
2.1.1 Mechanizm aktualizacji położenia . . . 5
2.1.2 Nowe operatory . . . 6
2.1.3 Kodowanie . . . 6
2.1.4 Modyfikacja struktury populacji oraz hierarchii . . . 6
2.2 Hybrydy GWO . . . 7
2.3 Paralelny GWO . . . 7

3 Zastosowanie GWO **7**
3.1 Uczenie maszynowe . . . 7
3.2 Problemy inżynierskie . . . 8
3.3 Pozostałe dziedziny . . . 8

4 Wielokryterialny GWO (MOGWO) **8**
4.1 Wprowadzone modyfikacje . . . 8
4.1.1 Archiwum . . . 9
4.1.2 Kontroler archiwum . . . 9
4.1.3 Siatka . . . 9
4.1.4 Selekcja najlepszego rozwiązania . . . 9
4.2 Algorytm . . . 11
4.3 Zastosowanie MOGWO . . . 11
4.3.1 Robotyka i planowanie ścieżki . . . 11
4.3.2 Planowanie . . . 11
4.3.3 Bezprzewodowa sieć czujników . . . 12

1 Grey Wolf Optimizer

1.1 Inspiracja

Algorytm GWO został zaproponowany w 2014 roku przez S. Mirjalili [1]. Inspiracją do jego powstania była unikatowa struktura hierarchii wśród wilków oraz sposób w jaki polują. Najpotężniejszym wilkiem jest osobnik α, który przewodzi całej grupie w polowaniu, migracji i karmieniu, Kiedy wilk α jest poza grupą, chory lub martwy, najsilniejszy wilk z grupy β przejmuje przywództwo. Władza i dominacja wilków δ i ω są dużo mniejsze niż wilków α i β. Drugą inspiracją jest sposób w jaki wilki polują. Głównymi fazami polowania są:

- Śledzenie, pogoń i zbliżenie się do ofiary.
- Pościg, otoczenie i napastowanie ofiary.
- Atak ofiary.

1.2 Model matematyczny

1.2.1 Hierarchia

Aby matematycznie zamodelować hierarchię wśród wilków, najlepiej dostosowane rozwiązanie uznajemy za α. Konsekwentnie drugie i trzecie najlepiej dostosowane rozwiązania to β i δ. Pozostałe rozwiązania kandydujące określane są jako ω. W GWO polowanie (optymalizacja) jest przewodzona przez α, β i δ. Pozostałe wilki ω podążają za nimi.

1.2.2 Otoczenie ofiary

Wilki w trakcie polowania okrążają ofiarę. Aby matematycznie zamodelować to zachowanie zaproponowano następujące równania:

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \tag{1}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{2}$$

gdzie t oznacza aktualną iterację. $\vec{A}$ i$\vec{C}$ są wektorami współczynników. $\vec{X}_p$ jest wektorem pozycji ofiary, a $\vec{X}$ to wektor pozycji wilka. Wektory $\vec{A}$ i $\vec{C}$ są obliczane:

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{3}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{4}$$

gdzie elementy $\vec{a}$ liniowo zmniejszają się z 2 do 0 w trakcie kolejnych iteracji, a $\vec{r}_1$ i $\vec{r}_2$ są losowymi wektorami z przedziału $[0, 1]$

Aby zobaczyć efekty równania 1 i 2 zilustrowano wektor dwuwymiarowy i kilku możliwych sąsiadów na obrazku 1(a). Tak jak można zauważyć, wilk na pozycji (X, Y) może zaktualizować swoją pozcyję zgodnie z pozycją ofiary (X^*, Y^*). Można osiągnąć różne miejsca wokoł najlepszego agenta w odniesieniu do aktualnejpozycji poprzez dostosowanie wartości wektorów $\vec{A}$ i $\vec{C}$. Na przykład, $(X^* - X, Y^*)$ moze być osiągnięte przez ustawienie $\vec{A} = (1, 0)$ i $\vec{C} = (1, 1)$. Możliwe aktualizacje pozycji agenta w przestrzeni 3D prezentuje obrazek 1(b). Należy zauważyć, że losowe wektoy $\vec{r}_1$ i $\vec{r}_2$ pozwalają wilkowi na osiągnięcie każdej pozycji pomiędzy punktami zilustrowanymi na obrazku 1. Czyli agent może zaktualizować swoją pozycję w przestrzeni dookoła ofiary w dowolnej losowej lokalizacji przy pomocy równań 1 i 2.

Ta koncepcja może zostać rozszerzona na przestrzeń przeszukiwań o n wymiarach. Agenci poruszają się w hiperkostkach wokół najlepszego rozwiązania uzyskanego do tej pory.

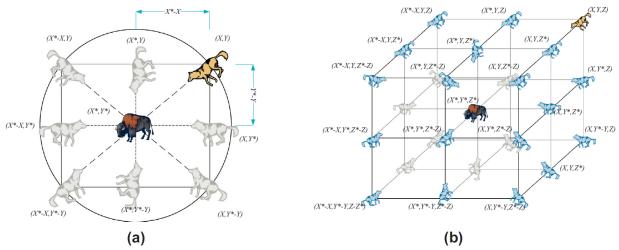


Figure 1: Wektory pozycji 2D i 3D i ich możliwe kolejne lokacje. Źródło [1]

1.2.3 Polowanie

Wilki posiadają umiejętność lokalizacji ofiary i okrążenia jej. Polowanie jest zwykle prowadzone przez α. β i δ mogą również okazjonalnie uczestniczyć w polowaniu. Jednakże w abstrakcyjnej przestrzeni przeszukiwań nie mamy informacji o lokalizacji optimum (ofiary). Aby matematycznie zamodelować proces polowania wśród wilków zakładamy, że α (najlepsze rozwiązanie kandydujące), β i δ posiadają większą wiedzę o potencjalnej lokalizacji ofiary. W związki z tym zapisujemy 3 najlepsze rozwiązania, które dotychczas otrzymaliśmy i obligujemy pozostałych agentów do aktualizacji ich pozycji na podstawie pozycji najlepszych agentów. (Liczone w każdej iteracji dla każdego agenta)

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}| \tag{5}$$

$$\vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}| \tag{6}$$

$$\vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}| \tag{7}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot (\vec{D}_\alpha) \tag{8}$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_1 \cdot (\vec{D}_\beta) \tag{9}$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_1 \cdot (\vec{D}_\delta) \tag{10}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1(t) + \vec{X}_2(t) + \vec{X}_3(t)}{3} \tag{11}$$

Algorytm GWO przyjmuje, że "najlepszą" informację o położeniu ofiary posiadają wilki α, β, δ i na podstawie ich położenia szacowane jest położenie ofiary - tak na prawdę na ich podstawie jest szacowany okrąg, w którym ta ofiara się znajduje.. Pozycja pozostałych wilków (agentów szukających) aktualizowana jest na podstawie pozycji α, β, δ. Wykorzystywany jest tutaj wcześniej opisany mechanizm otoczenia ofiary - nowa pozycja każdego z agentów to losowe miejsce w okół oszacowanej pozycji ofiary - czyli znajduje się w okręgu, który szacują α, β, δ.

Poniżej (wydaje mi się, że lepiej) opisany obrazek:

1.2.4 Atakowanie ofiary (eksploatacja)

Jak wcześniej wspomniano, wilki kończą polowanie atakiem ofiary, kiedy przestaje się ona poruszać (w rzeczywistości/naturalnym środowisku). Aby matematycznie zamodelować ten proces, zmniejszamy liniowo wartość $\vec{a}$. Należy zauważyć, że wartości $\vec{A}$ także zmniejszają się wraz z $\vec{a}$. W innych słowach wartości $\vec{A}$ są wartościami losowymi należącymi do przedziału $[-2\vec{a}, 2\vec{a}]$, gdzie wartości $\vec{a}$ zmniejszają się z 2 do 0 z każdą iteracją. Kiedy wartości $\vec{A}$ należą do $[-1, 1]$, następną pozycją agenta może być każda pozycja między aktualną pozycją, a pozycją ofiary.

Dotąd zaproponowane operatory pozwalają agentom na aktualizację pozycji zgodnie z pozycjami α, β, δ i atakować ofiarę. Jednakże algorytm ten nie jest odporny na stagnację w rozwiązaniach lokalnych.

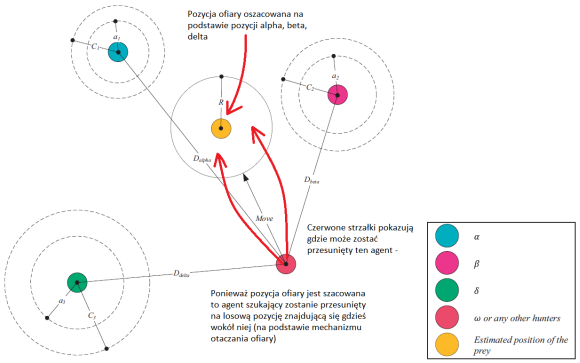


Figure 2: Aktualizacja pozycji. Źródło [1]

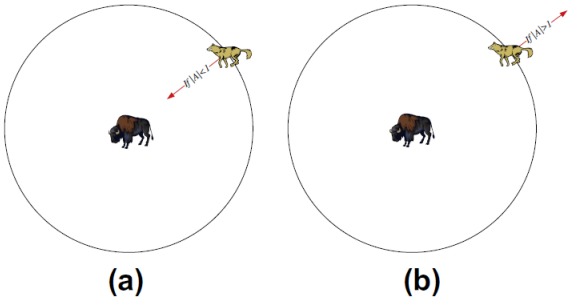


Figure 3: a) Atakowanie ofiary b) Poszukiwanie ofiary. Źródło [1]

1.2.5 Poszukiwanie ofiary (eksploracja)

Wilki zwykle poszukują ofiary zgodnie z pozycjami α, β, δ. Oddalają się od siebie w poszukiwaniu zdobyczy a później zbierają się by ją zaatakować. Aby matematycznie zamodelować oddalanie się wykorzystujemy losowe wartości $|\vec{A}| > 1$, aby wymusić na agencie oddalenie się od zdobyczy. Uwydatnia to proces eksploracji i pozwala GWO na globalne przeszukiwanie przestrzeni. Obrazek 3 również to pokazuje.

Kolejnym komponente faworyzującym eksplorację jestt $\vec{C}$. Tak jak pokazuje 4, $\vec{C}$ zawiera losowe wartosci z przedziału $[0, 2]$. Ten komponent zapewnia losowe wagi dla ofiary, aby stochastycznie podkreślić ($C > 1$) lub zmniejszyć znaczenie ($C < 1$) wpływ ofiary na definiowanie odległości w równaniu 1. Pozwala to GWO na bardziej losowe zachowanie w trakcie optymalizacji, sprzyjające eksploracji i unikaniu lokalnych optimów. Warto tutaj zaznaczyć, że w porónaniu do A, C nie jest liniowo zmniejszane. Celowo wymagamy, żeby C zapewniało losowe wartości przez cały czas do podkreślenia eksploracji nie tylko podczas początkowych iteracji, ale także końcowych. Ten komponent jest bardzo przydatny w przypadku stagnacji (lokalne optima), w szczególności w ostatnich iteracjach.

Wektor $\vec{C}$ może być uważany za imitację przeszkód, które są napotykane podczas polowania. Ogólnie mówiąc, przeszkody w naturalny sposób pojawiają się i uniemożliwiają szybkie i wygodne podejście do ofiary. I właśnie za to odpowiada $\vec{C}$. W zależności od pozycji wilka, może losowo nadać ofiarze wagę i utrudnić wilkom dotarcie do niej i vice versa.

1.3 Algorytm

Aby zobaczyć jak GWO jest teoretycznie w stanie rozwiązać probl

- Zaproponowana hierarchia pomaga GWO w zapisaniu najlepszych dotąd otrzymanych rozwiązań w trakcie kolejnych iteracji

- Zaproponowany mechanizm okrążania definiuje sąsiedztwo w kształcie koła wokół rozwiązań, które może być rozwinięte do liczniejszych wymiarów jako hiper-sfery.
- Losowe wartosci A i C pomagają rozwiązaniom kandydującym do posiadania hiper-sfer z różnymi losowymi promieniami
- Zaproponowany mechanizm polowania pozwala rozwiązaniom kandydującym na zlokalizowanie prawdopodobnej pozycji ofiary
- Eksploracja i eksploatacja są zapewnione przez adaptacyjne wartości a i A
- Adaptacyjne wartości a i A pozwalają GWO na płynne przejście między ekspolarcją i eksploatacją
- Ze wzrastającym A, połowa iteracji jest poświęcona eksploracji ($|A| \geq 1$) a druga połowa eksploatacji ($|A| < 1$)
- GWO posiada tylko 2 główne parametry, które wymagają dostrojenia - a i C

Algorytm 1 GWO

Losowa inicjalizacja populacji $X_i (i = 1, 2, ..., n)$ Osobnik jest reprezentowany przez n-wymiarowy wektor liczb rzeczywistych, gdzie n to liczba argumentów optymalizowanej funkcji.
Inicjalizacja a, A, C
Obliczenie wartości funkcji dopasowania dla każdego agenta
X_α = osobnik o najlepszym dostosowaniu
X_β = osobnik o drugim najlepszym dostosowaniu
X_δ = osobnik o trzecim najlepszym dostosowaniu
while t < maksymalna liczba iteracji **do**
  for all agent **do**
    Aktualizacja pozycji zgodnie z równaniem (11)
  end for
  Aktualizacja a, A, C
  Obliczenie wartości funkcji dopasowania dla każdego agenta
  Aktualizacja X_α, X_β, X_δ
  $t = t + 1$
end while
return X_α

2 Warianty GWO

W związku ze złożoną naturą rzeczywistych problemów optymalizacyjnych, algorytm GWO został zmodyfikowany tak, aby odpowiadał przestrzeniom przeszukiwania złożonych dziedzin. Poniżej zaprezentowano kilka znaczniejszych modyfikacji, które zostały zebrane w artykule zbiorczym [3].

2.1 Zmodyfikowane wersje GWO

2.1.1 Mechanizm aktualizacji położenia

W tych badaniach zajęto się poprawą równowagi między eksploracją i eksploatacją. Badania podzieliły się na dwa kierunki: pierwsze z nich próbowało dynamicznie aktualizować parametry GWO, a drugi z nich porponuje różne strategie aktualizacji osobników. Najważniejsze propozycje to:

- W pracy [4] zaproponowano zmniejszanie wartości $\vec{a}$ z wykorzystaniem zanikającej funkcji wykładniczej, a nie liniowej:

$$a = 2\Big(1 - \frac{iter^2}{maxIter^2}\Big) \tag{12}$$

  Autorzy przetestowali to podejście na 27 funkcjach testowych, a wyniki porównali z PSO, BA, CS i klasycznym GWO. Badania pokazały, że osiągnęli oni lepszą eksplorację.

- W pracy [5] zaproponowano zmniejszanie wartości $\vec{a}$ nieliniowo, gdzie μ jest nieliniowym wektorem modulacji w interwale $(0, 3)$

$$a = \left(1 - \frac{iter}{maxIter}\right) \cdot \left(1 - \mu \frac{iter}{maxIter}\right)^{-1} \tag{13}$$

  Przeprowadzone badania na licznych ograniczonych funkcjach pokazały, że tak zmodyfikowany GWO osiąga lepszą równowagę między eksploracją, a eksploatacją.

- W pracy [6] zaproponowano aktualizację pozycji agentów oparta na wprowadzeniu kroku, który jest proporcjonalny do dostosowania agenta w obecnej iteracji. Plusem tej modyfikacji jest mniejsza liczba parametrów i brak konieczności inicjalizacji parametrów początkowych.

$$X_i^{t+1} = \left(\frac{1}{t}\right)^{[(bestf(t) - f_i(t))/(bestf(t) - worstf(t))]} \tag{14}$$

  Przeprowadzone badania na 27 funkcjach testowych pokazały, że tak zmodyfikowany GWO osiąga szybszą zbieżność, a wyniki są lepsze w porównaniu do klasycznego GWO.

2.1.2 Nowe operatory

- W pracy [7] autorzy wprowadzili prosty operator krzyżowania między dwoma losowymi agentami. Jego celem jest dzielenie się informacjami w populacji. Przeprowadzone badania na 6 funkcjach testowych i porównanie wyników z klasycznym GWO pokazały, że operator krzyżowania ulepsza działanie, jakość rozwiązania i prędkość zbieżności.

- W pracy [8] wprowadzono kolejny typ operatora ewolucyjna dynamika populacji (ang. evolutionary population dynamics, EPD). Został on wprowadzony do eliminacji najgorszych osobników w populacji i zmiany ich pozycji w pobliżu wilków dowodzących. Autorzy twierdzą, że wprowadzenie tego operatora ulepsza medianę populacji w kolejnych generacjach oraz ma pozytywny wpływa na eksplorację i eksploatację. Porównanie wyników dla 13 funkcji testowych potwierdzają zalety tego wariantu.

- W pracy [9] zintegrowano algorytm optymalizacyjny Powell'a jako operator lokalnego przeszukiwania, a modyfikację nazwano PGWO. Algorytm Powell'a jest metodą znajdowania minimum funkcji, które nie musi być różniczkowalne, a pochodne nie są konieczne. Taka modyfikacja sukcesywnie wykonuje dwukierunkowe przeszukiwanie wśród rozwiązań kandydujących. Algorytm został przetestowany tylko na 7 funkcjach. Wyniki porównano z GWO, GA, PSO, GGSA, ABC i CS. Pokazały one poprawę działania w porównaniu do GWO i bardzo konurencyjne wyniki w porównaniu do reszty algorytmów.

2.1.3 Kodowanie

W pracy [10] zaproponowano zmianę kodowania osobników. Zamiast typowego kodowania rzeczywistego wykorzystano kodowanie zespolone. Agent posiada gen o dwóch częściach - część rzeczywistą oraz urojoną. Autorzy twierdzą, że taka reprezentacja może rozszerzyć pojemność informacji oraz zwiększyć różnorodność populacji. Rezultaty porównano z klasycznym GWO, ABC i GGSA. Na podstawie 6 funkcji testowych pokazano, że zaproponowana wersja algorytmu posiada bardzo konkurencyjne wyniki.

2.1.4 Modyfikacja struktury populacji oraz hierarchii

GWO posiada unikatową hierarchiczną strukturę populacji. Tak jak wcześniej wyjaśniono, istnieją 4 różne typy osobników. Trzy z nich posiada tylko jednego reprezentanta, a reszta osobników należy do czwartego typu. Ta wyróżniająca struktura zmotywowała badaczy do zbadania, czy zmiana w strukturze będzie miała jakieś efekty. W pracy [11] zaproponowano inną hierarchię przywództwa. Populacja jest podzielona na dwie niezależne subpopulacje. Pierwsza z nich nazwana jest kooperatywną grupą polującą, a druga losową grupą zwiadowczą. Zadaniem grupy zwiadowczej jest przeprowadzenie szerokiej eksploracji, natomiast grupy polującej jest wykonanie głębokiej eksploatację. W odróżnieniu od klasycznego GWO, wilki δ podzielone są na dwa typy: δ_1, który reprezentuje wilki-łowców i δ_2, które reprezentują wilki-zwiadowców. Autorzy tej modyfikacji zastosowali ją do dostrojenia parametrów sterowników generatora indukcyjnego opartego na turbinie wiatrowej. Wyniki porównali

z wynikami otrzymanymi z GA, PSO, GWO i MFO. Ich wyniki pokazały lepsze wartości dopasowania i wyższą stabilność.

2.2 Hybrydy GWO

Ogólnie mówiąc hybrydyzacja metaheurystyk polega na połączeniu ze sobą co najmniej dwóch algorytmów, aby uwydatnić i wykorzystać mocne strony każdego z nich. W literaturze można znaleźć przykład hybrydyzacji GWO z PSO w stylu sekwencyjnym do optymalizacji problemu o jednostkowy obszarze [12]. Wygenerowane wyniki były konkurencyjne, jednakże to podejście posiada wolniejszą zbieżność z powodu długiego, sekwencyjnego czasu wykonywania. Inna propozycja to połączenie GWO z DE. Dwie prace pokazały bardzo konkurencyjne wyniki w porównaniu do DE czy PSO [13] [14].

2.3 Paralelny GWO

Paralelizm może być efektywnie stosowany w metaheurystykach opartych na populacji. Populacja może zostać podzielona na mniejsze, a każda z nich może ewoluować na innym procesorze maszyny. Paralelizm algorytmu może efektywnie zmniejszyć czas wykonywania algorytmu oraz poprawić jakość rozwiązania. W jednej z prac zaproponowano paralelną wersję GWO, gdzie podgrupy ewoluują niezależnie i co określoną liczbę iteracji wymieniają się miedzy sobą najlepszymi rozwiązaniami. [15] Badania na 4 różnych funkcjach testowych pokazały, ze zmodyfikowana wersja algorytmu poprawia znacząco wydajność pod względem jakości rozwiązania oraz czasu wykonywania.

3 Zastosowanie GWO

W związku z impresywnymi zaletami GWO, algorytm został wykorzystany do rozwiązania wielu różnorodnych problemów. Poniższy diagram przedstawia zastosowanie GWO w różnych dziedzinach w czerwcu 2017r.

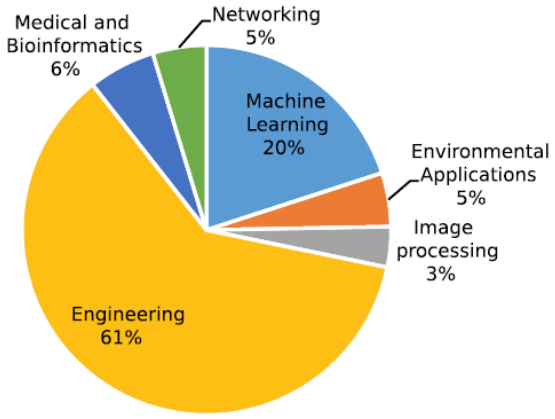


Figure 4: Zastosowanie GWO w różnych dziedzinach. Źródło [3]

3.1 Uczenie maszynowe

Można znaleźć zastosowanie w takich dziedzinach uczenia maszynowego jak:

- Ekstrakcja cech
- Trenowanie sieci neuronowych
- Optymalizacja SVM

- Klasteryzacja

3.2 Problemy inżynierskie

Algorytm GWO znalazł zastosowanie w między innymi:

- Projektowaniu i dostrajaniu sterowników
- Problemach dotyczących przesyłanie energii elektrycznej
- Robotyka i planowanie ścieżek
- Planowanie harmonogramów

3.3 Pozostałe dziedziny

Znalazł zastosowanie także w:

- Bezprzewodowej sieci czujników
- Modelowaniu środowiska
- medycynie i bioinformatyce

4 Wielokryterialny GWO (MOGWO)

Algorytm GWO jest w stanie rozwiązać problem o jednym kryterium. Autor tego algorytmu zaproponował jego wielokryterialną wersję do rozwiązywania problemów posiadających wiele kryteriów [2]. MOGWO korzysta z tego samego mechanizmu aktualizującego położenia. W związku z istnieniem wielu najlepszych rozwiązań - tak zwanych rozwiązań Pareto optymalnych - wprowadzono kilka modyfikacji.

4.1 Wprowadzone modyfikacje

Wprowadzono archiwum do zarządzania rozwiązaniami Pareto optymalnymi. Jest nie tylko magazynem, ale także selektorem osobników α, β, δ. Kontroler archiwum wybiera lidera z mniej zaludnionych regionów oszacowanego optymalnego frontu Pareto i zwraca je jako α, β, δ. Ten mechanizm został zaprojektowany, aby poprawić rozmieszczenie (pokrycie) optymalnych rozwiązań Pareto dla wszystkich kryteriów.

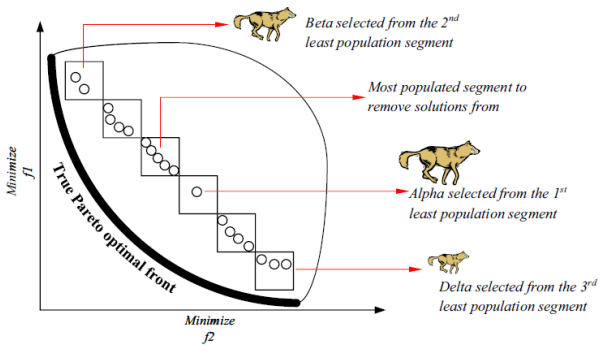


Figure 5: Selekcja lidera z najmniej zaludnionych obszarów. Źródło [3]

Kolejnym mechanizmem poprawy rozmieszczenia rozwiązań jest proces czyszczenia archiwum. Kiedy archiwum jest przepełnione, rozwiązania powinny zostać usunięte tak, aby pomieścić nowe niezdominowane rozwiązania.

MOGWO usuwa rozwiązania z najbardziej zaludnionych regionów. Dzięki temu prawdopodobieństwo poprawy rozwiązań rośnie wraz z kolejnymi iteracjami.

Zbieżność MOGWO jest podobna do GWO z powodu wykorzystania tego samego mechnizmu aktualizacji pozycji. Rozwiązania napotykają nagłe zmiany w pierwszej połowie iteracji i stopniowe fluktuacje w pozostałej części. Ze względu na wybór α, β, δ z najsłabiej zaludnionych regionów, wybrani liderzy będą pochodzili z różnych regionów. Pogarsza to zbieżność w kierunku jednego najlepszego rozwiązania, ale jednak jest to wymagane, aby utrzymać rozkład rozwiązań zgodnie z kryteriami.

Poniżej "wytłumaczono" te mechanizmy na podstawie publikacji [19].

4.1.1 Archiwum

Zadaniem archiwum jest przechowywanie niezdominowanych rozwiązań, zwanych populacją zewnętrzną. Są to najlepsze dotąd znalezione rozwiązania.

4.1.2 Kontroler archiwum

Zadaniem kontrolera archiwum jest podjęcie decyzji, czy obecne rozwiązanie powinno zostać dodane do archiwum. Czynność ta wygląda w każdej iteracji następująco :

Algorytm 2 Dodawanie rozwiązań do archiwum

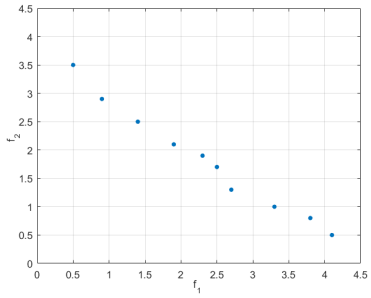
```
NS = niezdominowane rozwiązania znalezione w danej iteracji
if archiwum jest puste then
  Dodanie rozwiązań z NS do archiwum
else
  for all rozwiązanie ns z NS do
    if rozwiązanie ns jest zdominowane przez rozwiązanie w archiwum then
      Odrzucenie rozwiązania ns
    else
      Dodanie rozwiązania ns do archiwum
      if rozwiązanie ns dominuje nad innymi rozwiązaniami then
        Usunięcie rozwiązań zdominowanych z archiwum
      end if
    end if
  end for
end if
if archiwum jest pełne then
  Wykonanie mechanizmu adaptacji siatki
end if
```

4.1.3 Siatka

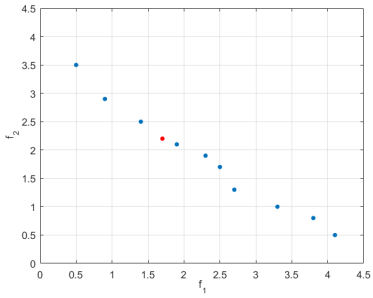
Algorytm wykorzystuje siatkę adaptacyjną, aby front Pareto został równomiernie rozmieszczony. Podstawową ideą jest użycie archiwum do przechowywania paretooptymalnych rozwiązań z wzięciem pod uwagę jego zawartości. W archiwum przestrzeń jest podzielona na regiony, tak jak na rysunku 6. Jeżeli dodawany osobnik leży poza obecnymi granicami siatki, należy ją zaktualizować. Siatka adaptacyjna tak naprawdę jest przestrzenią uformowaną przez hiperkostki, które posiadają tyle wymiarów, ile jest funkcji kryterialnych.

4.1.4 Selekcja najlepszego rozwiązania

Selekcja najlepszego osobnika w archiwum odbywa się w dwóch krokach:

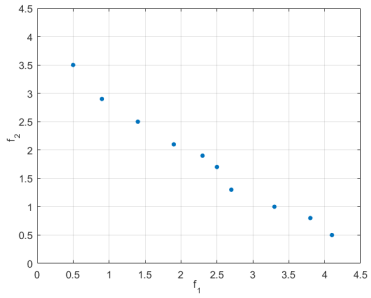


(a) Siatka przed dodaniem osobnika

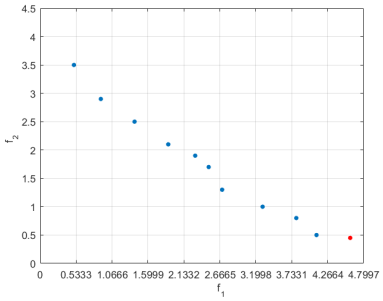


(b) Siatka po dodaniu osobnika

Figure 6: Graficzna reprezentacja dodawania nowego osobnika, który znajduje się w granicach siatki



(a) Siatka przed dodaniem osobnika



(b) Siatka po dodaniu osobnika

Figure 7: Graficzna reprezentacja dodawania nowego osobnika, który znajduje się poza granicami siatki

1. Selekcja hiperkostki z wykorzystaniem selekcji ruletkowej.
   Do kalibracji ruletki wykorzystujemy poniższe prawdopodobieństwo dla każdej hiperkostki:

$$P_i = \frac{c}{N_i},$$

   gdzie c to stała liczba większa od 1 oraz N_i to liczba rozwiązań należących do danej hiperkostki.

2. Selekcja losowego osobnika z wybranej kostki.

4.2 Algorytm

Algorytm 3 MOGWO

Losowa inicjalizacja populacji $X_i(i = 1, 2, ..., n)$ Osobnik jest reprezentowany przez n-wymiarowy wektor liczb rzeczywistych, gdzie n to liczba argumentów optymalizowanego problemu.
Inicjalizacja a, A, C
Oblicz wartości kryteriów dla każdego agenta
Znajdź niezdominowane rozwiązania i zainicjalizuj nimi archiwum
$X_\alpha = WybierzLidera(archiwum)$
Wyłącz α tymczasowo z archiwum, aby nie wybrać tego samego lidera
$X_\beta = WybierzLidera(archiwum)$
Wyłącz β tymczasowo z archiwum, aby nie wybrać tego samego lidera
$X_\delta = WybierzLidera(archiwum)$
Dodaj α, β do archiwum
while t < maksymalna liczba iteracji **do**
  for all agent **do**
    Aktualizacja pozycji
  end for
  Aktualizacja a, A, C
  Oblicz wartości kryteriów dla każdego agenta
  Znajdź niezdominowane rozwiązania
  Zaktualizuj archiwum zgodnie z otrzymanymi niezdominowanymi rozwiązaniami
  if Archiwum jest pełne **then**
    Usuń jedno z aktualnych rozwiązań w archiwum
    Dodaj nowe rozwiązanie do archiwum
  end if
  if jakiekolwiek z nowo dodanych rozwiązań jest zlokalizowane poza hiperkostkami **then**
    Zaktualizauj siatkę tak, aby pokrywała wszystkie rozwiązania (uruchom mechanizm siatki)
  end if
  $X_\alpha = WybierzLidera(archiwum)$
  Wyłącz α tymczasowo z archiwum, aby nie wybrać tego samego lidera
  $X_\beta = WybierzLidera(archiwum)$
  Wyłącz β tymczasowo z archiwum, aby nie wybrać tego samego lidera
  $X_\delta = WybierzLidera(archiwum)$
  Dodaj α, β do archiwum
  $t = t + 1$
end while
return $archiwum$

4.3 Zastosowanie MOGWO

4.3.1 Robotyka i planowanie ścieżki

W pracy [16] autorzy wykorzystali MOGOW do optymalizacji planowania ścieżki dla robota. Użyto dwóch kryterów do minimalizacji, które odpowiadają za długość oraz płynność ścieżki. Przeprowadzono wiele simulacji w różnych statycznych środowiskach. Wyniki pokazują, że zaproponowane podejście poprawnie zapewnia optymalną ścieżkę do celu bez kolidowania z przeszkodami.

4.3.2 Planowanie

W pracy [17] autorzy zaproponowali dyskretną wersję MOGWO do optymalizacji planowania rzeczywistego procesu spawania. Jako kryteria, które należy zminimalizaować, wybrano długość trwania całego procesu oraz obciążenie maszyn. Badania i testy statystyczne pokazały przewagę zaproponowanego rozwiązania nad NSGA-II i SPEA2.

4.3.3 Bezprzewodowa sieć czujników

W pracy [18] autorzy zaproponowali MOGOW do rozwiązywania problemu lokalizacji wierzchołków. Zaznaczyli oni, że podejście wielokryterialne może być bardziej wydajny w porównaniu do klasycznego podejścia jednokryterialnego. Wykorzystali oni dwa kryteria - odległość wierzchołków oraz geometrię topologiczną. Badania pokazały, że w ten sposób można efektywnie zmniejszyć średni błąd lokalizacji.

References

[1] Mirjalili, S., Mirjalili, S.M. and Lewis, A., 2014. Grey wolf optimizer. Advances in engineering software, 69, pp.46-61.

[2] Mirjalili, S., Saremi, S., Mirjalili, S.M. and Coelho, L.D.S., 2016. Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. Expert Systems with Applications, 47, pp.106-119.

[3] Faris, H., Aljarah, I., Al-Betar, M.A. and Mirjalili, S., 2017. Grey wolf optimizer: a review of recent variants and applications. Neural Computing and Applications, pp.1-23.

[4] Mittal N, Singh U, Sohi BS (2016) Modified grey wolf optimizer for global engineering optimization. Appl Comput Intell Soft Comput 2016:8

[5] Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge

[6] Dudani AR, Chudasama K (2016) Partial discharge detection in transformer using adaptive grey wolf optimizer based acoustic emission technique. Cogent Eng 3(1):1256083

[7] Kishor A, Singh, PK (2016) Empirical study of grey wolf optimizer. In: Proceedings of fifth international conference on soft computing for problem solving. Springer, pp 1037–1049

[8] Saremi S, Mirjalili SZ, Mirjalili SM (2015) Evolutionary population dynamics and grey wolf optimizer. Neural Comput Appl 26(5):1257–1263

[9] Zhang S, Zhou Y (2015) Grey wolf optimizer based on Powell local optimization method for clustering analysis. Discret Dyn Nat Soc 2015:17

[10] Luo Q, Zhang S, Li Z, Zhou Y (2015) A novel complex-valued encoding grey wolf optimization algorithm. Algorithms 9(1):4

[11] Yang B, Zhang X, Yu T, Shu H, Fang Z (2016) Grouped grey wolf optimizer for maximum power point tracking of doubly-fed induction generator based wind turbine. Energy Convers Manag 133:427–443

[12] Kamboj VK (2016) A novel hybrid pso-gwo approach for unit commitment problem. Neural Comput Appl 27(6):1643–1655

[13] Jitkongchuen D (2015) A hybrid differential evolution with grey wolf optimizer for continuous global optimization. In: 2015 7th international conference on information technology and electrical engineering (ICITEE). IEEE, pp 51–54

[14] Zhu A, Xu C, Li Z, Wu J, Liu Z (2015) Hybridizing grey wolf optimization with differential evolution for global optimization and test scheduling for 3d stacked soc. J Syst Eng Electron 26(2):317–328

[15] Pan TS, Dao TK, Chu SC, et al (2015) A communication strategy for paralleling grey wolf optimizer. In: International conference on genetic and evolutionary computing. Springer, pp 253–262

[16] Tsai PW, Dao TK, et al (2016) Robot path planning optimization based on multiobjective grey wolf optimizer. In: International conference on genetic and evolutionary computing. Springer, pp 166–173

[17] Lu C, Xiao S, Li X, Gao L (2016) An effective multi-objective discrete grey wolf optimizer for a real-world scheduling problem in welding production. Adv Eng Softw 99:161–176

[18] Hai Phong Private Universty (2016) Estimation localization in wireless sensor network based on multi-objective grey wolf optimizer. In: Advances in information and communication technology: proceedings of the international conference, ICTA 2016, vol 538. Springer, p 228

[19] Coello, C.A.C., Pulido, G.T. and Lechuga, M.S., 2004. Handling multiple objectives with particle swarm optimization. IEEE Transactions on evolutionary computation, 8(3), pp.256-279.