AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA INFORMATYKI

Praca dyplomowa magisterska

Skuteczność algorytmu roju cząstek w wybranych problemach optymalizacji ciągłej

Effectiveness of particle swarm algorithm in selected continuous optimization problems

Autor: *Mateusz Gruszka*
Kierunek studiów: *Informatyka*
Opiekun pracy: *dr hab. inż. Marek Kisiel-Dorohinicki*

Kraków, 2015

Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję . . . tu ciąg dalszych podziękowań które zostaną uzupełnione pod koniec :)

Spis treści

1. Wprowadzenie ... 6
1.1. Cele pracy ... 6
1.2. Zakres pracy ... 6
1.3. Zawartość pracy ... 7
2. Rozdział o pyagu i reszcie ... 8
2.1. Platforma PyAGE ... 8
2.2. Sposób porównywania wyników ... 8
2.2.1. Funkcja Rastrigina ... 8
3. Algorytmy rojowe ... 9
3.1. Przegląd wiedzy ... 9
3.2. Algorytm roju cząstek ... 10
3.2.1. Opis algorytmu ... 10
3.2.2. Przemieszczenie ... 11
3.2.3. Zasada działania algorytmu ... 12
3.3. Modyfikacje algorytmu roju cząstek ... 13
3.3.1. Rozszerzenie wiedzy cząstek ... 13
3.3.2. Dodanie losowej składowej ... 13
3.3.3. Nadawanie wag parametrom ... 13
3.3.4. Zmiana prędkości ruchu ... 13
4. Algorytmy ewolucyjne ... 14
4.1. Przegląd wiedzy ... 14
4.2. Algorytm EMAS ... 15
4.2.1. Agent ... 16
4.2.2. Interakcje agentów ... 16
4.2.3. Ewolucja ... 17
4.2.4. Wyspy obliczeniowe ... 18
5. Algorytm roju cząstek na platformie PyAGE ... 19

5.1. rowne wagi ... 19
5.2. stala predkosc ... 19
5.3. dodanie losowego wspolczynnika ... 19
6. Porównanie algorytmów genetycznego i PSO ... 20
6.1. 1 wyspa ... 20
6.2. 3 wyspy ... 20
6.3. 6 wysp ... 20
6.4. 9 wysp ... 20
6.5. 12 wysp ... 20
7. Podsumowanie ... 21
7.1. zalety pso wzgledem emas ... 21
7.2. mozliwy rozwoj ... 21

Spis rysunków

3.1 Wizualizacja ruchu cząstki PSO 11
3.2 Diagram blokowy algorytmu PSO 12

4.1 Diagram blokowy klasycznego algorytmu genetycznego 15
4.2 Operacje ewolucyjne w algorytmie EMAS 17

1. Wprowadzenie

Tematem niniejszej pracy jest sprawdzenie skuteczności algorytmu roju cząstek w wybranych problemach optymalizacji ciągłej. Skuteczność rozwiązania została oceniona względem dostarczonej platformy wraz z istniejącym rozwiązaniem.

1.1. Cele pracy

Celem pracy jest realizacja różnych wariantów implementacji algorytmu roju cząstek oraz badanie ich skuteczności w wybranych problemach optymalizacji ciągłej. W implementacji wykorzystane zostanie istniejące środowisko wspierające budowę rozproszonych modeli obliczeniowych w języku Python (pyAgE). Skuteczność algorytmu roju cząstek badana będzie eksperymentalnie dla różnych wariantów konfiguracji obliczeń oraz różnych problemów benchmarkowych.

1.2. Zakres pracy

W czasie realizacji pracy, została rozszerzona platforma pyAgE o możliwość dokonywania obliczeń za pomocą algorytmu roju cząstek. Zostały zaimplementowane zarówno elementy składowe algorytmu jak i przetestowane konfiguracje pozwalające na badanie skuteczności algorytmu.

Porównanie odbywało się pomiędzy dostarczonym wraz z platformą pyAgE algorytmem genetycznym EMAS, a różnymi konfiguracjami algorytmu roju cząstek.

Algorytm roju cząstek został przystosowany aby sprawdzić jego skuteczność w kilku wersjach:

- Podstawowa wersja, zawierająca możliwość sterowania:
  - Prędkością
  - Wagami składowych
- Rozszerzoną o kroki algorytmu ewolucyjnego wykonywane razem z rojowymi
- Rozszerzoną o informację o sąsiadach, dostarczaną przez platformę pyAgE
- Rozszerzoną o wyspy obliczeniowe

1.3. Zawartość pracy

Rozdział 2 zawieraja informację o platformie pyAgE i jej komponentach. Znajdują się tam również informacje o funkcjach benchamrkowych, ze szczególnym uwzględnieniem funkcji stosowanej do porównania w niniejszej pracy.

Następną częścią pracy (rozdziały 3 i 4) jest przybliżenie mechanizmów działania zarówno algorytmów rojowych jak i ewolucyjnych, szczególnie tych, które zostały zastosowane w implementacji. W rodziale 3.3 zostały opisane częste modyfikacje algorytmu roju cząstek.

W rozdziale 5 znajdują się informację o zaimplementowanych rozwiązaniach oraz wyniki badań prowadzące do uzyskania najlepszych parametrów algorytmu roju cząstek.

Pod koniec pracy znajdują się porównania algorytmów rojowego i ewolucyjnego oraz analiza skuteczności algorytmu rojowego pod kątem postawionego problemu.

2. Rozdział o pyagu i reszcie

a na pewno o samym pyage, sposob porownania wynikow i funkcja moze w innym?

2.1. Platforma PyAGE

2.2. Sposób porównywania wyników

2.2.1. Funkcja Rastrigina

3. Algorytmy rojowe

Inteligencją stadną nazywane są techniki sztucznej inteligencji bazujące na wiedzy o społecznych zachowaniach w samo zorganizowanym systemie.

Systemy rojowe są najczęściej zbudowane z populacji prostych agentów oddziałujących na siebie nawzajem oraz na środowisko w którym się znajdują. Nie posiadają scentralizowanej struktury kierującej zachowaniem indywidualnych agentów. Agenci posiadają jedynie wiedzę o akcji jaką powinni podjąć przy danych bodźcach zewnętrznych. Wzajemne oddziaływanie pomiędzy agentami prowadzi do złożonych zachowań populacji.

Przykłady takich zachowań są bardzo liczne w naturze. Zaliczyć do nich można kolonie mrówek, klucze ptaków, ławice ryb czy roje pszczół. W przypadkach tych mamy doczynienia z licznymi osobnikami, którzy wpływając na siebie nawzajem osiągają złożone populacje zdolne realizować wielopoziomowe zadania.

3.1. Przegląd wiedzy

Pierwszy raz idea wykorzystania zachowań zaobserwowanych w naturze została zaproponowana w 1987 roku[1]. Dzięki wykorzystaniu kilku względnie prostych reguł udało się osiągnąć w animacjach symulowanie bardziej skomplikowanych, realistycznie wyglądających zachowań stada ptaków.

Pojęcie inteligencji stadnej zostało po raz pierwszy użyte w 1989 roku[2] w kontekście zrobotyzowanych systemów komórkowych, natomiast w roku 1995[3] pojawiła się pierwsza wersja algorytmu roju cząstek (ang. Particle Swarm Optimization (PSO)). Pierwotnym założeniem algorytmu PSO była sumylacja społeczeństwa, jednak problem okazał się zbyt złożony dla takiego algorytmu. Odnalazł on jednak zastosowanie w problemach optymalizacji ciągłej.

Wraz ze wzrostem zainteresowania algorytmami rojowymi, pojawiło się wiele badań i publikacji nad różnymi algorytmami. Do najpopularniejszych i najdokładniej opisanych należy zaliczyć takie jak algorytm mrówkowy (ang. Ant colony optimization (ACO))[7], czy algorytmy pszczele (ang. Bees algorithm (BA))[5] i pszczelej kolonii (ang. Artifical bee colony algorithm (ABC))[4]. Część z algorytmów mimo, że ich pierwotne wersje były dostosowane do rozwiązywania problemów dyskretnych, z czasem została zmodyfikowana również do rozwiązywania problemów ciągłych. Przykładem takiego algorytmu może być algorytm kolonii mrówek, którego dostosowanie zostało opublikowane ponad dziesięć lat od zaproponowania pierwotnej wersji[6].

Ze względu na różnorodność otaczającej przyrody kolejne algorytmy inspirowane naturą są nieustannie proponowane i rozwijane. W momencie pisania niniejszej pracy (rok 2015), jednym z nowszych zaproponowanych jest algorytm lwich mrówek (ang. Ant lion optimizer (ALO))[8], którego działanie oparte jest na mechanizmach polowania lwich mrówek.

3.2. Algorytm roju cząstek

Algorytm roju cząstek naśladuje zachowania stadne. Podczas zdobywania doświadczenia(wiedzy), cząsteczki wzajemnie na siebie oddziałując, jednocześnie przesuwając się w coraz lepszy obszar przestrzeni rozwiązań.

Optymalizacja nie jest wymagająca pod względem zasobów. Zapotrzebowanie na pamięć i czas procesora w każdej iteracji są niskie, a sama zmiana parametrów cząstki odbywa się przy wykorzystaniu podstawowych operatorów matematycznych. Algorytm roju cząstek jest jest wystarczający dla wielu problemów i nie wymaga skomplikowanych metod używanych na przykład przez algorytmy ewolucyjne.

Każdy agent w populacji posiada zestaw mechanizmów warunkujących jego ruch po przestrzeni rozwiązań. Ponadto ma pamięć o najlepszym miejscu w przestrzeni jakie odwiedził, oraz wiedzę o położeniu agenta z najlepszym rozwiązaniem z populacji.

3.2.1. Opis algorytmu

Jeśli przestrzeń poszukiwań jest D-wymiarowa, można zaprezentować i-tą cząsteczkę populacji jako D-wymiarowy wektor 3.1,

$$x_i = (x_i1, x_i2, x_i3, \ldots, x_iD)^T \tag{3.1}$$

gdzie T oznacza numer iteracji. Najlepszą odwiedzoną pozycję (particle best) i-tej cząsteczczki można oznaczyć jako 3.2.

$$p_{b_i} = (p_{b_{i1}}, p_{b_{i2}}, p_{b_{i3}}, \ldots, p_{b_{iD}}) \tag{3.2}$$

Na podobnej zasadzie, wprowadzając oznaczenie g_b (global best) oznaczona zostaje najlepsza pozycja w przestrzeni rozwiązań w stadzie. Przyjmując takie oznaczenia, każdy agent populacji przemieszcza się według równania ruchu 3.3.

$$v_{id}^{n+1} = v_{id}^n + (p_{b_{id}}^n - x_{id}^n) + (g_{b_{id}}^n - x_{id}^n) \tag{3.3}$$

co skutkuje, że w kolejnych iteracjach jego pozycja jest uaktualniana zgodnie z równaniem 3.4

$$x_{id}^{n+1} = x_{id}^n + v_{id}^{n+1} \tag{3.4}$$

gdzie:

d = 1, 2, 3, . . . , D

i = 1, 2, 3, . . . , N

N - rozmiar populacji

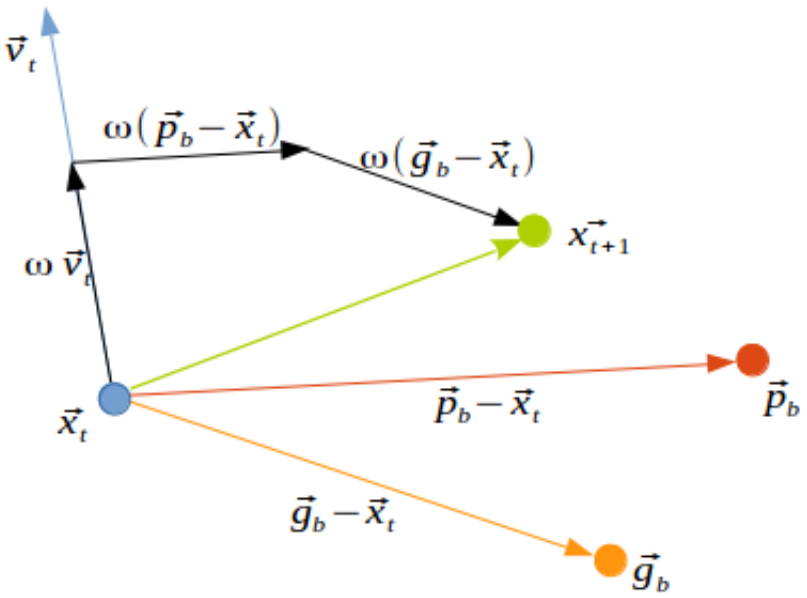
n = 1,2,3, . . . - numer iteracji

Zgodnie z równaniem 3.3, każda nowa pozycja w przestrzeni rozwiązań zależy wyłącznie od poprzednich wartości cząsteczki oraz wartości jej sąsiadów. W celu manipulowania prędkością przesunięcia w konkretnej iteracji, równanie ruchu cząsteczki zostało rozszerzone o współczynnik inercji ω (3.5).

$$v_{id}^{n+1} = \omega(v_{id}^{n} + (p_{b_{id}}^{n} - x_{id}^{n}) + (g_{b_{id}}^{n} - x_{id}^{n}) \tag{3.5}$$

3.2.2. Przemieszczenie

Wizualizację zgodnie z równaniami zamieszczonymi w rozdziale 3.2.1, można prześledzić na rysunku 3.1.

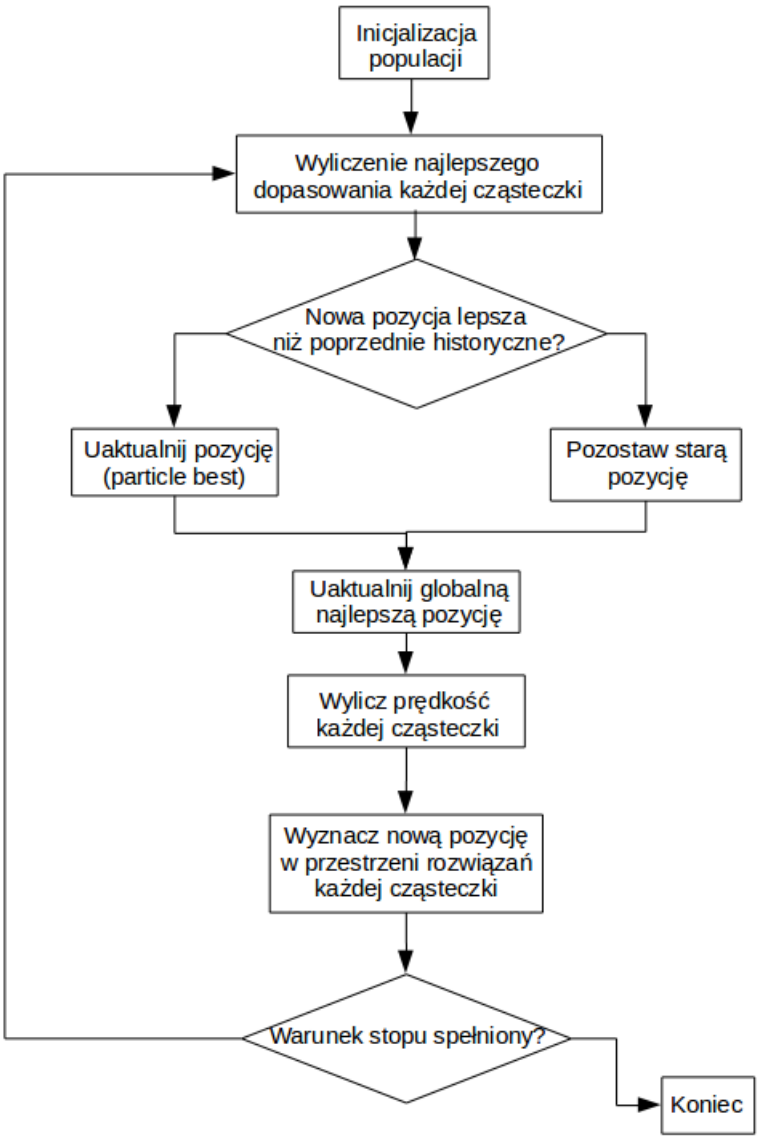


Rysunek 3.1: Wizualizacja ruchu cząstki PSO

W danej iteracji, cząsteczka znajduje się w konkretnym punkcie x_t. Zna położenie najlepszej cząsteczki z całej populacji g_b, oraz pamięta swoją najlepszą odwiedzoną do tej pory pozycję p_b. Agent wyzacza wektory przesunięcia względem tych punktów, oraz przemieszczenia z poprzedniej iteracji v_t. Każdy z wektorów jest mnożony przez współczynnik inercji ω, a następnie wszystkie wektory są ze sobą składane. Wynikiem złożenia jest nowa pozycja cząsteczki w przestrzeni rozwiązań.

3.2.3. Zasada działania algorytmu

Diagram 3.2 przedstawia pełny mechanizm działania algorytmu. Pierwszym krokiem jest zainicjalizowanie startowej populacji w losowych miejscach przestrzeni rozwiązań. Następnie dla każdego agenta liczone jest dopasowanie. Jeśli aktualne dopasowanie jest lepsze niż zapamiętane, uaktualniana jest pamiętana pozycja w przestrzeni rozwiązań. W przypadku gdy dopasowanie jest gorsze, zapamiętana informacja pozostaje bez zmian. Kolejnym krokiem jest uaktualnienie informacji najlepszej pozycji z całej populacji. Posiadając wszystkie informacje, cząsteczka wyznacza swoją prędkość w danej iteracji, a następnie przemieszcza się zgodnie z nią w przestrzeni rozwiązań. Aż do spełnienia warunku stopu (uzyskanie pożądanego dopasowania lub osiągnięcia limitu iteracji), cząsteczki ponownie wyliczają sewoje dopasowanie i powtarzają cały cykl.



Rysunek 3.2: Diagram blokowy algorytmu PSO

3.3. Modyfikacje algorytmu roju cząstek

W paragrafie 3.2 została przedstawiona podstawowa, najprostsza wersja algorytmu roju cząstek. Istnieje szereg rozszerzeń i modyfikacji algorytmu, które powodują zwiększenie jego dokładności i skuteczności.

3.3.1. Rozszerzenie wiedzy cząstek

Rozszerzając wiedzę agentów o dodatkowe informacje o otoczeniu, często okazuje się, że zwiększa się dokładność jak i prędkość w jakiej otrzymywany jest satysfakcjonujący wynik. Jednym ze sposobów, jest dodanie wiedzy o położeniu najlepszej cząsteczki w pewnym otoczeniu danego agenta[9]. Dołożenie takiego parametru, pociąga za sobą zmianę wyliczania ruchu cząstki (rys. 3.1) o dodatkowy wektor. Podejście takie zwiększa skupienie cząsteczek i pozwala na dokładniejsze przeszukiwanie przestrzeni rozwiązań w danym zakresie.

3.3.2. Dodanie losowej składowej

Uwzględnienie podczas ruchu cząstki dodatkowego składowego wektora, którego wartość oraz kierunek generowana jest losowo, daje możliwość pozbycia się pewnych potencjalnych problemów. Cząstki które poruszają się w sposób opisany w rozdziale 3.2.2, oraz te rozszerzone o informacje o sąsiedztwie, obarczone są ryzykiem wpadania w lokalne ekstrema. Dodając losową składową, wymuszany jest ruch często oddalający agenta od roju, co może skutkować wyskoczeniem z lokalnego ekstremum i dalsze przeszuiwanie przestrzeni rozwiązań w celu znalezienia ekstremum globalnego.

3.3.3. Nadawanie wag parametrom

Podstawowa wersja algorytmu roju cząstek zakłada jednakową wagę każdego z wektorów podczas wyliczania nowej pozycji cząstki. Modyfikacja wprowadzająca dla każdego wektora parametr definiujący jego wagę, pozwala na lepsze dostosowanie algorytmu dla danego problemu.

3.3.4. Zmiana prędkości ruchu

Algorytm roju cząstek w swojej pierwotnej wersji nie zakładał zmiany prędkości agentów w czasie. Rozszerzenie dające możliwość manipulowania współczynnikiem inercji ω na przestrzeni kolejnych iteracji pozwala na uniknięcie potencjalnego "przeskakiwania"poprawnego rozwiązania przez agenty. Wersja podstawowa niesie za sobą ryzyka, że po pewnej ilości iteracji cząsteczki będą krążyły wokół ekstremum, jednak długość wektora będzie zbyt duża, aby udało się im "trafić"w rozwiązanie. Zmniejszanie współczynnika ω wraz z kolejnymi iteracjami pozwoli cząstkom dokładniej przeszukać dany zakres, jednocześnie utrzymując szeroki zasięg przeszukiwań na początku działania algorytmu.

4. Algorytmy ewolucyjne

Algorytmami ewolucyjnymi nazywne są algoytmy, które w celu przeszukania przestrzeni rozwiązań wykorzystują mechanizmy zaczerpnięte ze zjawiska ewolucji biologicznej. Jest to ogólna nazwa dla metod takich jak algorytmy genetyczne, strategie ewolucyjne czy neuroewolucje.

Podobnie jak opisane w rozdziale 3 algorytmy rojowe, ewolucyjne również zawierają populację agentów wpływających nawzajem na siebie. Populacja agentów generowana jest losowo, wraz z pewnym zestawem cech dla każdego agenta - genotypem. Genotyp jest takim zestawem cech agenta, który umiejscawia go w pewnej przestrzeni rozwiązań, co umożliwia jego ewaluację. Podczas działania algorytmu, agenci poprzez krzyżowanie się, umieranie i rodzenie wpływają na swoje genotypy.

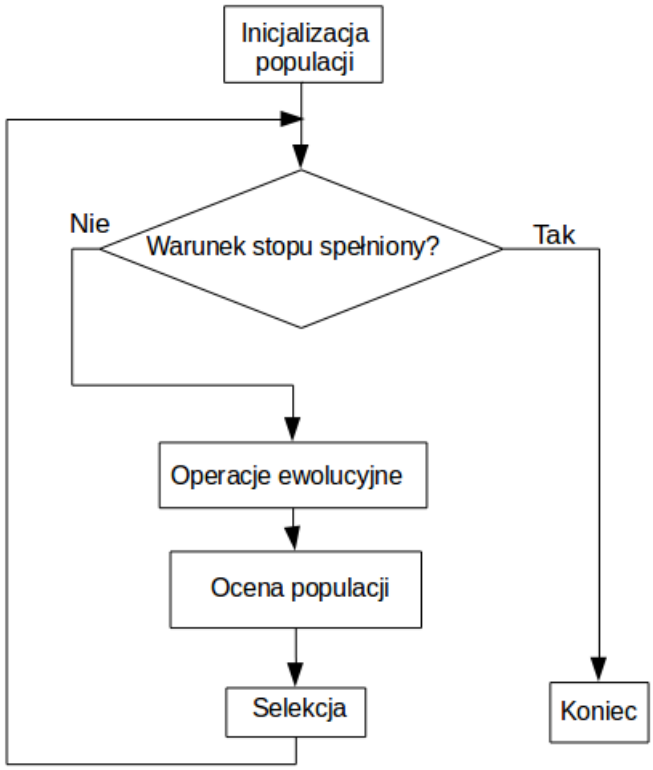
Na przestrzeni lat zostało zaproponowanych wiele algorytmów bazujących na mechanizmach genetycznych, jednak wszystkie z nich opierały się na tych samych bazowych mechanizmach. Każdy z osobników populacji mógł, zmieniając swój genotyp, przybliżyć całą populację do znalezienia optymalnego rozwiązania postawionego problemu. Większość współczesnych rozwiązań stosuje również krzyżowanie się osobników, jako drugą główną składową działania algorytmów tego typu.

4.1. Przegląd wiedzy

Początki algorytmów ewolucyjnych sięgają lat 50 XX wieku[10], jednak ich idee nie były rozwijane przez wiele lat, głównie ze względu na ograniczenia sprzętowe jak i metodologiczne. Dopiero dwadzieścia lat później[11] pojawiły się prace rozwijające modele ewolucyjne. Wtedy też zostało zaproponowane twierdzenie Hollanda o schematach, które uważane jest za podstawę wyjaśnienia algorytmów genetycznych.

Znaczącą kwestią wpływającą na tempo rozwoju algorytmów genetycznych, było użycie techniki uwzględniającej ewolucję zarówno przez mutację jak i krzyżowanie się osobników z danej populacji. Podczas kolejnych lat badań, algorytmy tego typu zostały poszerzone o kod genetyczny pozwalający reprezentować strukturę każdego problemu.

Klasyczne algorytmy ewolucyjne działają zgodnie z algorytmem przedstawionym na diagramie 4.1, lub podobnie do niego.



Rysunek 4.1: Diagram blokowy klasycznego algorytmu genetycznego

Początkowo inicjalizowana jest losowa populacja, która wykonuje między sobą zachowania ewolucyjne aż do spełnienia warunku stopu. Podczas każdej iteracji z całej populacji wybierana jest część osobników która zostanie poddana krzyżowaniu się między sobą. Następnie te osobniki poddawane są mutacji. Dla każdego z osobników wyliczana jest funkcja przystosowania, pozwalająca ocenić jakość jego genotypu.

4.2. Algorytm EMAS

Algorytm EMAS (ang. Evolutionary Multi-Agent System) jest paradygmatem obliczeniowym zaproponowanym w 1996 roku[13]. Jest to połączenie algorytmu ewolucyjnego z systemem wieloagentowym. Idea algorytmu opiera się na koncepcji, że agenci w środowisku mogą się łączyć, reprodukować i umierać.

Przeznaczony jest do pracy bez zachowania jakiejkolwiek globalnej wiedzy o problemie. Agenci są niezależni oraz zdolni do podejmowania własnych decyzji dotyczących ich akcji. Ta cecha sprawia, że algorytm jest łatwo skalowalny i pozwala na zrównoleglenie.

Dziedziczenie i selekcja to dwa główne elementy algorytmów ewolucyjnych, które w algorytmie EMAS realizowane są za pomocą zjawisk śmierci i reprodukcji. Agenci o najlepszym przystosowaniu

są zachowane i mogę produkować swoje potomstwo. Agenci o najgorszych parametrach są całkowicie usuwane z otoczenia. Takie zachowanie zmusza populację do ewolucji oraz poprawia jej parametry[14].

4.2.1. Agent

Każdy z agentów charakteryzuje się trzema parametrami:

– genotyp (ang. genotype)

– dopasowanie (ang. fitness)

– energia (ang. energy)

Genotyp agenta stanowi pojedynczą odpowiedź na zadany problem populacji. Jest to podstawa do obliczenia dopasowania. Genotyp jest cechą dziedziczoną podczas reprodukcji i ulega mutacji w procesie ewolucji. Jako zasadniczy parametr agenta jest to podstawa dla pozostałych parametrów.

Kolejną cechą jest dopasowanie, które jest liczbą reprezentującą jakość genotypu. Lepsze genotypy mają lepsze wartości dopasowania i mają większe prawdopodobieństwo aby być wybranym przy procesie reprodukcji. Sama wartość jest obliczana bezpośrednio z parametru genotypu po stworzeniu agenta. Dopasowanie danego agenta, tak samo jak jego genotyp nie zmienia sie w czasie jego życia.

Ostatnią cechą, wpływającą na sposób selekcji jest energia. Ze względu na brak globalnej wiedzy agentów, nie jest możliwa ocena ich wszystkich w tym samym czasie. Ponieważ proces ewolucji jest asynchroniczny, metody selekcji znane z klasycznych algorytmów ewolucyjnych nie mogły zostać użyte. Z tego powodu została wprowadzona energia, można opisać ten parametr jako stan agenta, który podczas interakcji może ją zyskiwać lub tracić, zależnie od jakości genotypu. Agenci z lepszym genotypem są bardziej skłonni do gromadzenia energii, jednak całkowita ilość energii w populacji jest stała.

Ponieważ agenci w algorytmie EMAS są całkowicie autonomiczni, decyzję o swoim zachowaniu podejmują na podstawie poziomu energii. Jeśli jej liczba przekracza pewien próg to będą się reprodukować, a jeśli osiągnie zero to agent umiera[15].

4.2.2. Interakcje agentów

Istnieją trzy możliwe działania agenta, które może podjąć w danym kroku iteracji:

– śmierć (ang. death)

– reprodukcja (ang. reproduction)

– walka (ang. fight)

Śmierć to usunięcie agenta z populacji, spowodowane jest najczęściej poprzez osiągnięcie przez danego agenta zerowego poziomu energii.

Reprodukcja jest procesem tworzenia nowych agentów. Wymaga wystąpienia jednego lub dwójki rodziców i skutkuje odpowiednio jednym lub dwójką nowo powstałych agentów. Jak zostało wspomniane

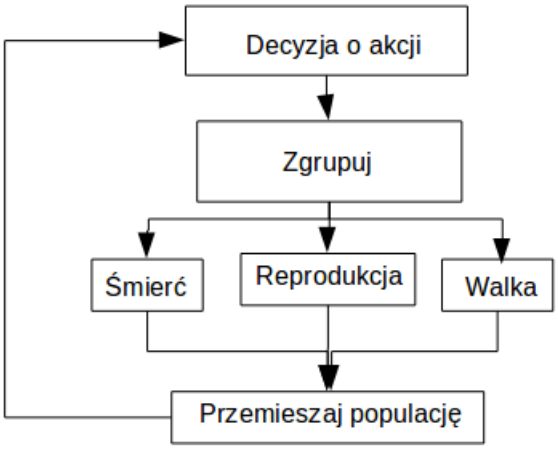
w rozdziale 4.2.1, rozwiązanie oraz dopasowanie są stałe podczas życia agenta, więc ich wartość u rodziców się nie zmienia. Jedyną zmianą parametru jest zmniejszenie ich poziomu energii, która jest przekazywana nowopowstałym agentom. Genotypy nowo narodzonych agentów są tworzone poprzez losowe zmieszanie genotypów ich rodziców. W momencie uzyskania genotypu, możliwe jest obliczenie dopasowania danego agenta.

Walka jest działaniem odpowiedzialnym za wymianę energii. Dzięki niej, agenci o lepszym genotypie są w stanie pobrać energię od tych z gorszym. Walka sprowadza się do porównaniu dopasowania dwóch agentów i w jego efekcie transferze energii. Jest to szybki sposób porównania i nagradzania najlepszych rozwiązań.

Jak zostało wspomniane wcześniej, ilość energii jest stała w całym układzie. W wyniku reprodukcji nowo narodzony agent dostaje energię od rodziców, natomiast w przypadku walki następuje wymiana pomiędzy dwoma agentami.

4.2.3. Ewolucja

Pomimo, że proces ewolucji jest asynchorniczny można zauważyć pewien schemat zachowania, przedstawiony na rysunku 4.2.



Rysunek 4.2: Operacje ewolucyjne w algorytmie EMAS

Pierwszym krokiem jest zdecydowanie jakie działania powinien podjąć agent w danej iteracji. Ponieważ nie ma żadnej wiedzy globalnej, jego decyzje opierają się wyłącznie o jego parametry. Wszyscy agenci zostają oznaczeni jaką akcję podejmą w danym kroku.

Następnie agenci są dzieleni na trzy grupy, odpowiednio dla akcji które zostaną podjęte. Ma to na celu spowodowanie współdziałania ich ze sobą. Ponieważ agenci nie wiedzą jakie działania podejmą pozostali, ten krok jest wymagany aby spowodować interakcje wewnątrz populacji.

Kolejnym krokiem jest przeprowadzenie akcji wewnątrz grup. Część agentów umiera, a część powstaje, zależnie od składu grup. W ostatnim kroku agenci są zbierani w całość, a następnie przemieszani. Ma to na celu uniknięcie ciągłej interakcji ze sobą tych samych agentów.

4.2.4. Wyspy obliczeniowe

Populacją nazywany jest cały zbiór agentów, Jej początkowy rozmiar jest parametryzowany i zmienia się w czasie wykonywania programu. Interakcje wewnątrz populacji nie odbiegają zbytnio od większości algorytmów ewolucyjnych.

Podczas pojedynczego przebiegu programu można utworzyć wiele różnych populacji nazywanych wyspami[15], zmieniających się niezależnie w tym samym czasie.

Poprzez wprowadzenie migracji między różnymi wyspami, jest możliwe eksportowanie pewnych rozwiązań pomiędzy wyspami, co ma pozytywny wpływ na ogólną wydajność. Dobre rozwiązanie opracowane w jednej populacji może być wprowadzone do innej, powodując modyfikację genotypu.

Wprowadzenie wysp obliczeniowych dodaje każdemu agentowi dodatkową akcję jaką może podjąć, a mianowicie migrację. Jak wszystkie pozostałe akcje, zależna jest ona od wartości energii danego agenta.

5. Algorytm roju cząstek na platformie PyAGE

rozdział o różnych parametrach pso, porównanie wszystkich implementacji, eksperymenty z parametrami itp.

5.1. rowne wagi

5.2. stala predkosc

5.3. dodanie losowego wspolczynnika

6. Porównanie algorytmów genetycznego i PSO

tutaj będą te wszystkie wykresy które do tej pory konsultowaliśmy, i wszystkie ich opisy itp. tutaj juz beda tylko takie pso(pso, pso+sasiedztwo, pso+wyspy) ktore w poprzednim rozdziale bylo jako najlepsze

6.1. 1 wyspa

6.2. 3 wyspy

6.3. 6 wysp

6.4. 9 wysp

6.5. 12 wysp

6.6. porównanie czasu działania

7. Podsumowanie

Rozdzial podsumowywujący wszystkie wyniki razem z jakims ogolnym komentarzem

7.1. zalety pso wzgledem emas

7.2. mozliwy rozwoj

Bibliografia

[1] C. W. Reynolds *Flocks, Herds, and Schools: A Distributed Behavioral Model*. Computer Graphics, 21(4), Lipiec 1987, str. 25-34

[2] G. Beni, J. Wang *Swarm Intelligence in Cellular Robotic Systems*. NATO Advanced Workshop on Robots and Biological Systems, Włochy, Lipiec 1989

[3] J. Kennedy, R. Eberhart *Particle Swarm Optimization*. Proceedings of IEEE International Conference on Neural Networks IV, 1995, str. 1942–1948

[4] D. Karaboga *An Idea Based On Honey Bee Swarm for Numerical Optimization*. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Turcja, 2005

[5] D.T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri , S. Rahim , M. Zaidi *The Bees Algorithm – A Novel Tool for Complex Optimisation Problems*. Technical Note, Manufacturing Engineering Centre, Cardiff University, Wielka Brytania, 2005

[6] X Hu, J Zhang, and Y Li *Orthogonal methods based ant colony search for solving continuous optimization problems*. Journal of Computer Science and Technology, Springer, Styczeń 2008, str. 2-28

[7] M. Dorigo, L.M. Gambardella *Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem*. IEEE Transactions on Evolutionary Computation, 1997, str. 53-66

[8] S. Mirjalili *The Ant Lion Optimizer*. Advances in Engineering Software 83, 2015, str. 80–98.

[9] P.N. Suganthan *Particle swarm optimiser with neighbourhood operator*. Evolutionary Computation, IEEE, Washington, 1999

[10] G.E.P. Box *Evolutionary operation: A method for increasing industrial productivity*. Appl. Statistics, vol. VI, no.2, 1957, str. 81–101

[11] J. H. Holland *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975

[12] A. Byrski, R. Dreżewski, M. Kisiel-Dorohinicki, L. Siwik *Evolutionary Multi-Agent Systems*. AGH University of Science and Technology, 2012
`https://age.iisg.agh.edu.pl/emas/emas.html`.

[13] K. Cetnarowicz, M. Kisiel-Dorohinicki, E. Nawarecki *The application of evolution process in multi-agent world (MAW) to the prediction system*. M. Tokoro, editor, Proc. of the 2nd Int. Conf. on Multi-Agent Systems (ICMAS'96). AAAI Press, 1996

[14] A. Byrski, R. Drezewski, L. Siwik, M. Kisiel-Dorohinicki *Evolutionary multiagent systems*. The Knowledge Engineering Review, 2013

[15] M. Kisiel-Dorohinicki *Agent-oriented model of simulated evolution*. In William I. Grosky and Frantisek Plasil, editors, SofSem 2002: Theory and Practice of Informatics, volume 2540 of LNCS. Springer-Verlag, 2002.