Obliczenia z wykorzystaniem sztucznej inteligencji

wykład IV

Particle Swarm Optimization

Joanna Kołodziejczyk

2016

Plan wykładu

Eberhart and Kennedy

Swarm czyli rój

Słowo to zostało użyte, by opisać rodzinę procesów socjologicznych. Odnosi się do zdezorganizowanych ruchów poruszających się obiektów, głównie owadów, przemieszczających się nieregularnie, chaotycznie. Inteligencja roju może mieć miejsce w wielowymiarowej przestrzeni.

Particle

Cząstka, pojedynczy obiekt w roju.

Intelligence

Odnosi się do inteligencji społecznej w grupie organizmów koegzystujących ze sobą.

PSO w odniesieniu do EA

Podobieństwo do EA

PSO używa populacji kandydatów, by ewoluować optymalne, lub bliskie optymalnego rozwiązanie problemu optymalizacyjnego. Jakość optymalizacji mierzona jest zadaną przez użytkownika funkcją przystosowania.

Różnice do EA

PSO używa jako elementy populacji „cząstki", które przemierzają przestrzeń problemu. Kiedy populacja jest inicjalizowana losowo „cząstki" mają też stochastycznie przypisane wartości prędkości. W każdej iteracji prędkość cząstki jest zmieniana

- w kierunku poprzedniej najlepszej pozycji i
- w kierunku najlepszej pozycji swoich sąsiadów.

PSO w odniesieniu do CA

Podobieństwo do automatów komórkowych

Trzy najważniejsze cechy CA to:

1. pojedyncze komórki są aktualizowane równolegle
2. wartość każdej komórki zależy tylko od wartości jej samej i jej najbliższych sąsiadów
3. wszystkie komórki są aktualizowane na podstawie tych samych reguł.

Cząstki w PSO mogą być widziane jako komórki w CA, których stany zmieniają się symultanicznie w wielu kierunkach.

Dlaczego PSO

„So why, after all, did we call our paradigm a “particle swarm?” Well, to tell the truth, our very first programs were intended to model the coordinated movements of bird flocks and schools of fish. As the programs evolved from modeling social behavior to doing optimization, at some point the two-dimensional plots we used to watch the algorithms perform ceased to look much like bird flocks or fish schools and started looking more like swarms of mosquitoes. The name came as simply as that.”

Pięć głównych zasad w PSO

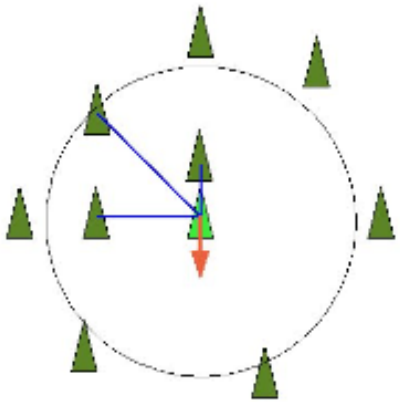
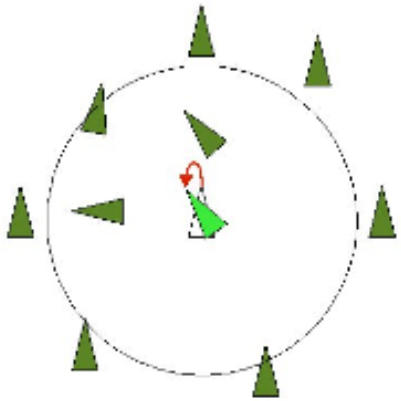
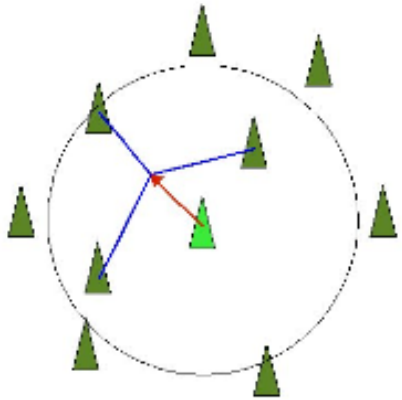
1. *Zasada bliskości:* populacja powinna mieć możliwość przeprowadzania łatwych obliczeń w czasie i przestrzeni.
2. *Zasada jakości:* populacja powinna odpowiadać na czynniki jakości w środowisku.
3. *Zasada zróżnicowanych odpowiedzi:* populacja nie powinna działać w zbyt wąskich kanałach.
4. *Zasada stabilności:* Populacja nie powinna zmieniać swojego zachowania za każdym razem, gdy zmienia się środowisko.
5. *Zasada adaptacyjności:* Populacja powinna móc modyfikować swoje zachowanie, gdy tylko pozwoli to na zysk.

Wzorce z zachowań stada

Model behawioralny Reynoldsa

Według niego agent działa zgodnie z trzema zasadami:

1. *Separacji:* każdy obiekt stara się odsunąć od innego obiektu, gdy jest za blisko.
2. *Wyrównania:* obiekty starają się wybrać kierunek zgodny ze średnim kierunkiem jego sąsiadów.
3. *Spójności:* każdy obiekt stara się poruszać w kierunku średniej pozycji swoich sąsiadów.

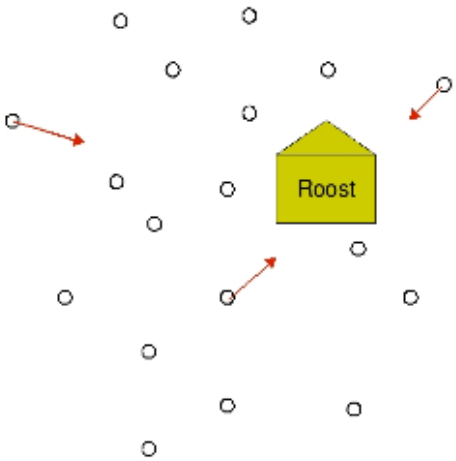




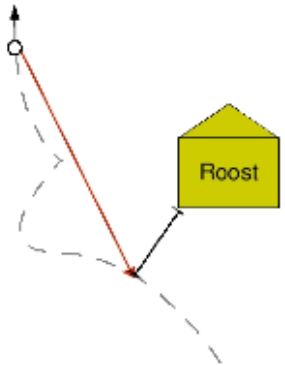
Modyfikacja Kennedy and Eberhart

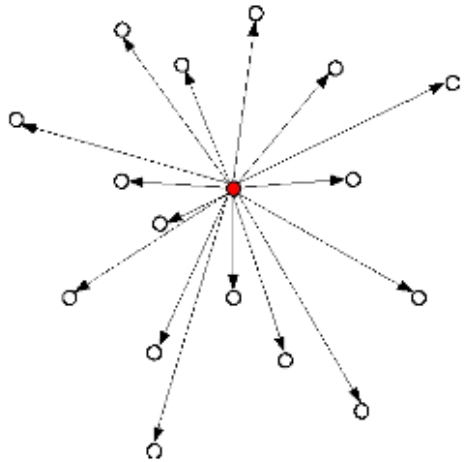
Dodanie elementu „grzęda" z modelu Heppnera

Taki agent zachowuje się zgodnie z zasadami:

1. Każdy obiekt jest przyciągany przez obecność i w kierunku „grzędy".
2. Każdy obiekt pamięta gdzie było bliżej do „grzędy".
3. Każdy agent dzieli informację o tym gdzie było najbliżej „grzędy" z innymi agentami (pierwotnie ze wszystkimi).







Ławice ryb E. O. Wilson

Kolejna inspiracja

In theory at least, individual members of the school can profit from the discoveries and previous experience of all other members of the school during the search for food. This advantage can become decisive, outweighing the disadvantages of competition for food items, whenever the resource is unpredictably distributed in patches.

Zdanie sugeruje dzielenie informacji o pokarmie w społeczności organizmów, co daje przewagę ewolucyjną. To stało się podstawą PSO.

Plan wykładu

Nearest-neighbor velocity matching

Zgodność prędkości na podstawie najbliższych sąsiadów

1. Populacja ptaków była losowo inicjalizowana z pozycją na torycznej siatce z prędkością X i Y.
2. W pętli dla każdego agenta (ptaka) szukano najbliższego sąsiada i przypisywano jego prędkości X i Y do prędkości badanego agenta.
3. Uzyskano synchronizację ruchów, ale też nienaturalną stabilność. Stado nie zmieniało kierunku.

Craziness

Odrobina szaleństwa

1. W każdej iteracji pewne losowe zmiany został dodane do losowo wybranych prędkości X i Y.
2. Okazało się to być wystarczającą modyfikacją.
3. Uzyskano w miarę naturalne zachowanie stada.

Heppner's bird simulations

„Grzęda"

Symulacja ta pokazywała, że grzęda przyciąga ptaki tak długo, aż na niej wylądują. W rzeczywistości ptaki lądują gdziekolwiek, gdzie jest jedzenie. Jak znajdują jedzenie?

Modyfikacja „cornfield vector"

Dwuwymiarowy wektor współrzędnych XY w przestrzeni. Każdy agent został zaprogramowany do oceny obecnej pozycji w odniesieniu do równania: tak, że na pozycji (100, 100) wartość wynosiła zero. Każdy agent pamięta najlepszą wartość i pozycję XY, dla której tą wartość uzyskano. Wartości te oznaczono jako pbestx[] i pbesty[] (dla każdego agenta). Dzięki takiej modyfikacji dużo łatwiej było korygować prędkości.

Nowe zasady zmiany prędkości

1. Jeżeli pozycja była na prawo od pbestx, to prędkość X oznaczona vx była pomniejszana o losową wartość ważoną pewnym parametrem systemu $vx[] = vx[] - rand() * p_{increment}$
2. Jeżeli pozycja była na lewo od pbestx, to analogicznie $vx[] = vx[] + rand() * p_{increment}$
3. Dla Y z nadwyżką: $vy[] = vy[] - rand() * p_{increment}$
4. Dla Y z niedomiarem: $vy[] = vy[] + rand() * p_{increment}$

Globalne dopasowanie

Rój zna też najlepszą pozycję znalezioną przez rój i jej wartość. Przypisano indeks tablicy z najlepszymi wartościami do zmiennej o nazwie *gbest*, tak aby *pbestx*[*gbest*] była najlepszą pozycją w X, i *pbesty*[*gbest*] najlepszym Y, a informacje były dostępne do wszystkich członków stada.
Modyfikacje przeprowadza się zgodnie z regułami:

1. if $presentx[] > pbestx[gbest]$ then $vx[] = vx[] - rand() * g_{increment}$
2. if $presentx[] < pbestx[gbest]$ then $vx[] = vx[] + rand() * g_{increment}$
3. if $presenty[] > pbesty[gbest]$then $vy[] = vy[] - rand() * g_{increment}$
4. if $presenty[] < pbesty[gbest]$ then $vy[] = vy[] + rand() * g_{increment}$

Wyniki ze stosowania cornfield vector

1. Agenci krążyli wokół punktu (100,100), aż znaleźli symulowane pole kukurydzy.
2. Z dużymi wartościami $p_{increment}$ i $g_{increment}$ rój wydawał się gwałtownie osadzać w polu. W małej liczbie iteracji prawie cały rój gromadził się w małym kręgu wokół punktu docelowego.
3. Z małymi wartościami $p_{increment}$ i $g_{increment}$ rój wirował wokół celu powoli się do niego zbliżając. Obiekty grupowały się nieznacznie i grupy przesuwały niezależnie, ale ostatecznie wszystkie wpadały do celu.

Modyfikacje usprawniające

Dopasowanie systemu

- Usunięcie procedury craziness.
- Usunięcie porównania najbliższych sąsiadów. Zachowanie wizualnie się zmieniło, ale cel nadal osiągany i to w krótszym czasie.
- *pbest* i *gbest* zachowano jako ważne.
- *pbest* to pamięć autobiograficzna, w której każdy osobnik pamięta fakt ze swoich doświadczeń. Zmianę prędkości na jej podstawie nazwano „simple nostalgia".
- *gbest* symuluje wiedzę publiczną lub grupowe normy i standardy.

Pierwsza aplikacja

Przeprowadzono testy na uczeniu wielowarstwowego perceptronu.
Założenia:

- Trzy warstwy: 2,3,1.
- XOR problem: dwa wejścia, jedno wyjście.
- 13 parametrów sieci.
- Agenci przeszukiwali przestrzeń wag, aż osiągnęli minimum błędu.

Wyniki: w czasie średnio 30.5 iteracji przy populacji 20 agentów znaleziono rozwiązanie, gdzie $e < 0.05$.

Zmiana w dopasowaniu prędkości

Pomimo skuteczności w zachowaniu systemu było coś dziwnego. Przyczyną były nierówności:

- jeśli bestx> presentx pomniejsz prędkość
- jeśli bestx <presentx powiększ prędkość.

Zmiana

Zamiast testować znak, prędkości zostały dopasowane według ich różnicy od najlepszej lokacji w danym wymiarze.

$$vx[][] = vx[][] + rand() * p_{increment} * (pbestx[][] - presentx[][])$$

Dla bestx stosuje się parametr $g_{increment}$.

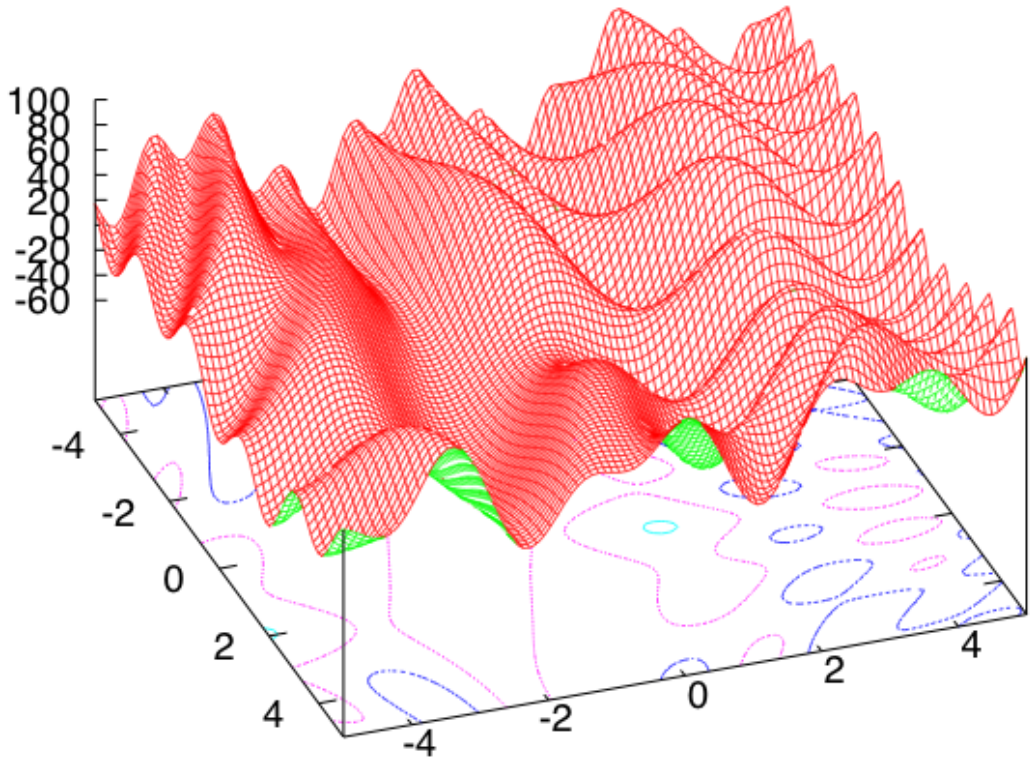
Plan wykładu

Zadanie optymalizacji w przestrzeni ciągłej

Optymalizacja w $\mathbb{R}^n$

Znaleźć $\mathcal{X}^* \subseteq \mathcal{X} \subseteq \mathbb{R}^n$ takie, że:

$$\mathcal{X}^* = \textit{argmin}\ f(\mathbf{x}) = \{\mathbf{x}^* \in \mathcal{X} : f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad \forall x \in \mathcal{X}\}$$



Algorytm w skrócie

Particle Swarm Optimization

Input: f — funkcja optymalizowana
Input: ps — rozmiar populacji
Dane: pop — populacja cząstek
Dane: i — indeks cząstek
Output: $\mathbf{x}^*$ — znalezione quasi/optimum

1. Wygeneruj pop o liczności ps
2. Powtarzaj dopóki nie jest spełniony warunek zakończenia:
   2.1 Dla każdej i-tej cząsteczki w populacji uaktualnij parametry cząstki.
3. Podaj najlepszy z populacji jako rozwiązanie $\mathbf{x}^*$

Algorytm kroki

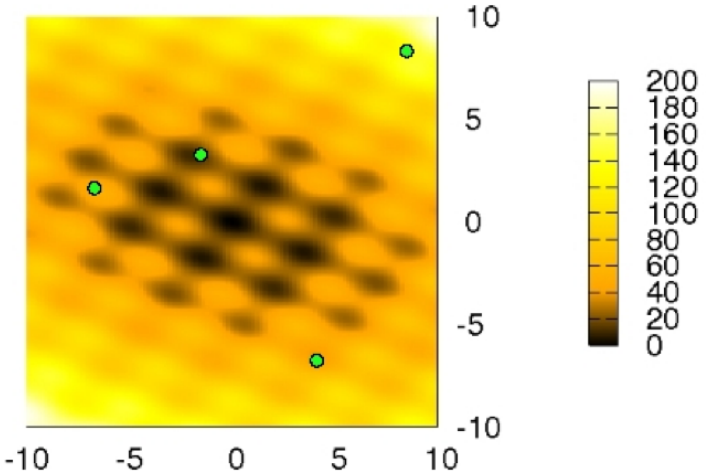
1. Utwórz populację agentów z rozkładem jednostajnym w przestrzeni $\mathcal{X}$.
2. Oszacuj każdą cząstkę zgodnie z funkcją oceny.
3. Jeżeli bieżąca pozycja cząsteczki jest lepsza niż poprzednia, to ją uaktualnij.
4. Określ najlepszą cząsteczkę, według poprzednich pozycji cząsteczek.
5. Uaktualnij prędkość cząsteczki zgodnie ze wzorem:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \varphi_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{x}_i^t) + \varphi_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{x}_i^t)$$

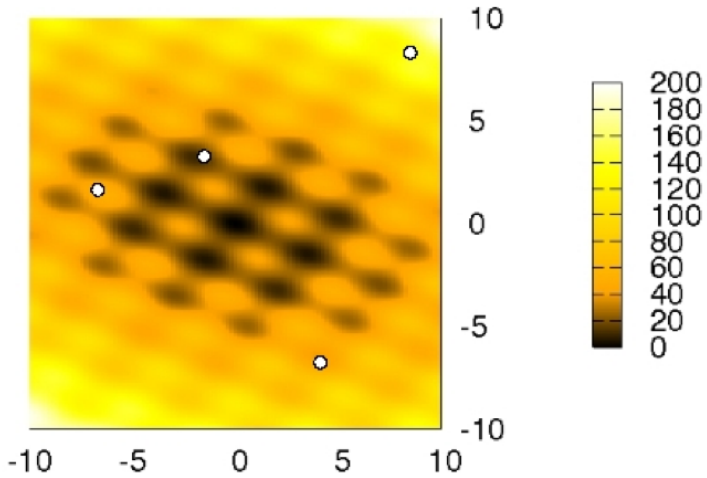
6. Przesuń cząsteczki do nowych pozycji zgodnie z regułą $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}$.
7. Idź do kroku drugiego, aż spełniony będzie warunek stopu.

Algorytm kroki wizualizacja

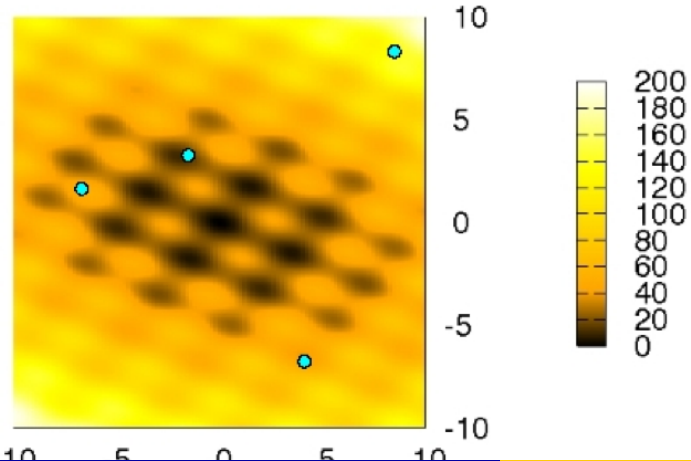
Krok 1



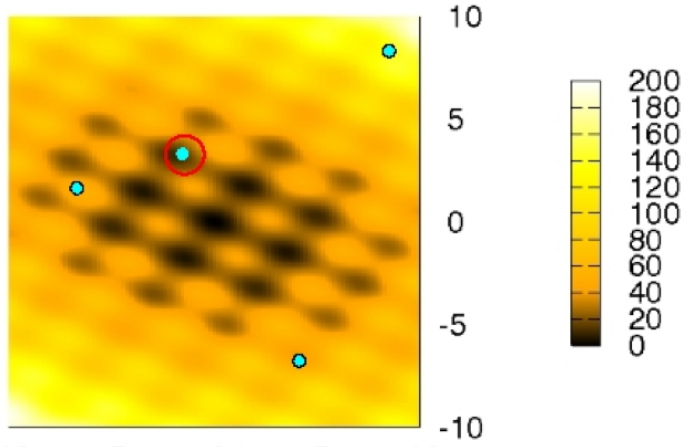
Krok 2



Krok 3

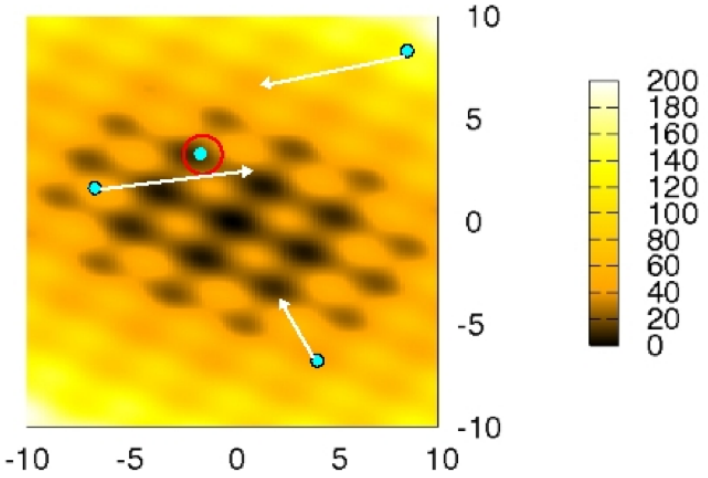


Krok 4

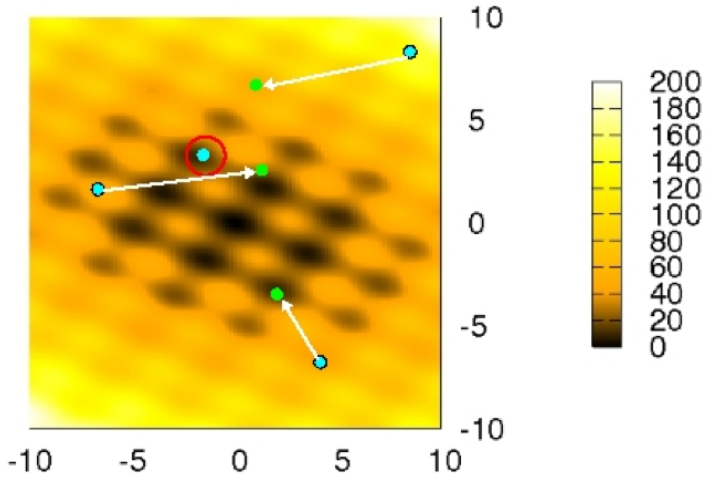


Algorytm kroki wizualizacja

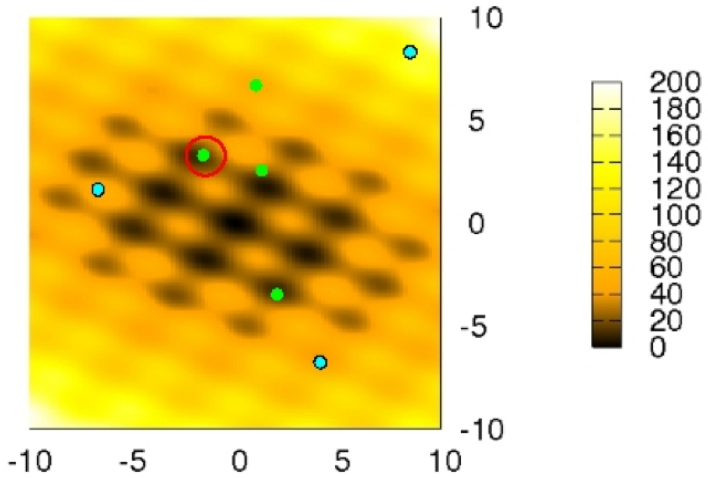
Krok 5



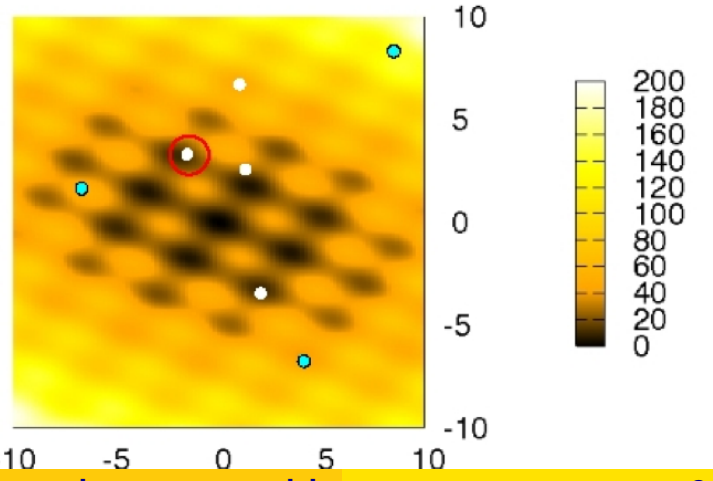
Krok 6



Krok 7

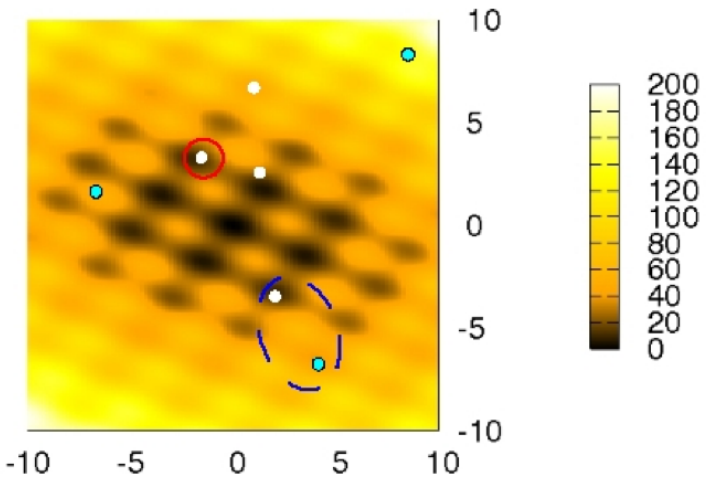


Krok 2

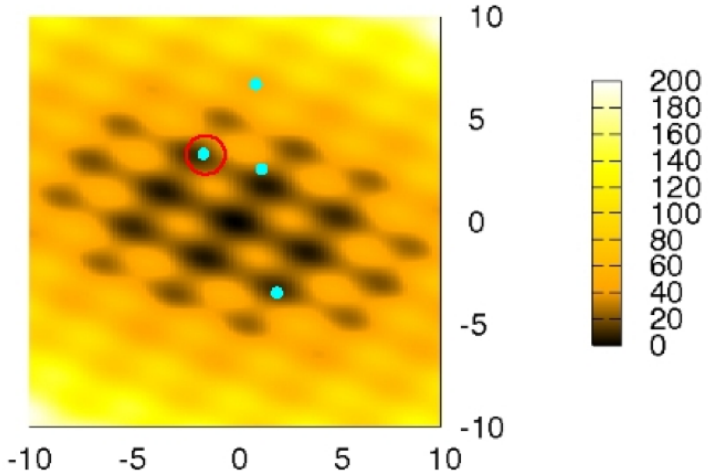


Algorytm kroki wizualizacja

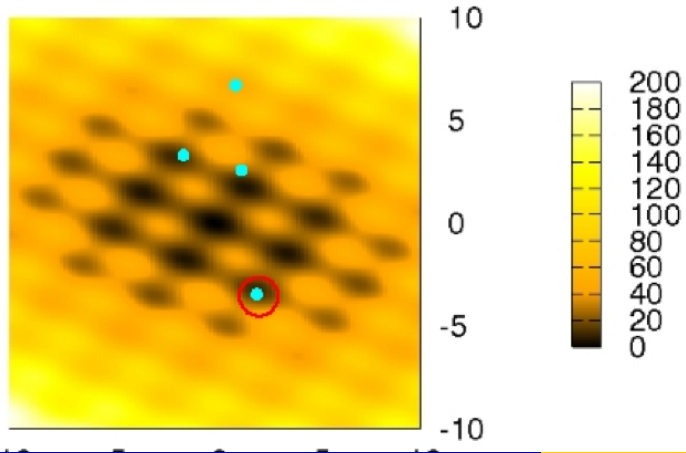
Krok 3



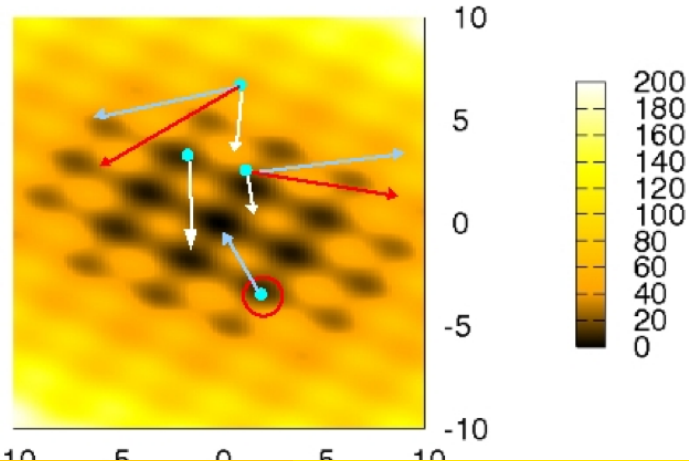
Krok 3



Krok 4

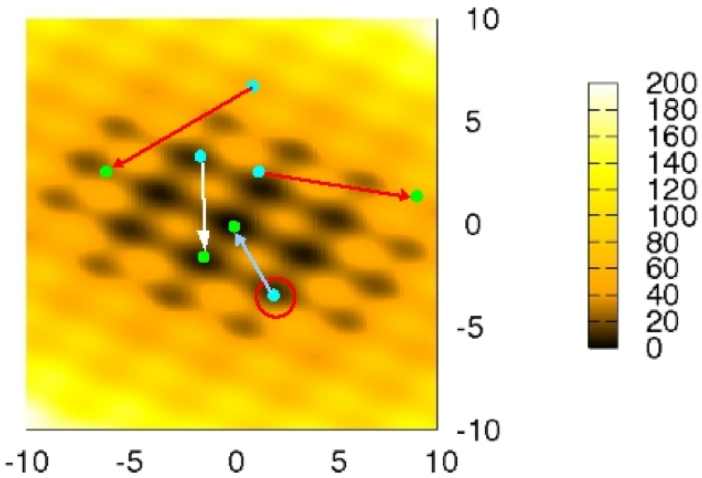


Krok 5

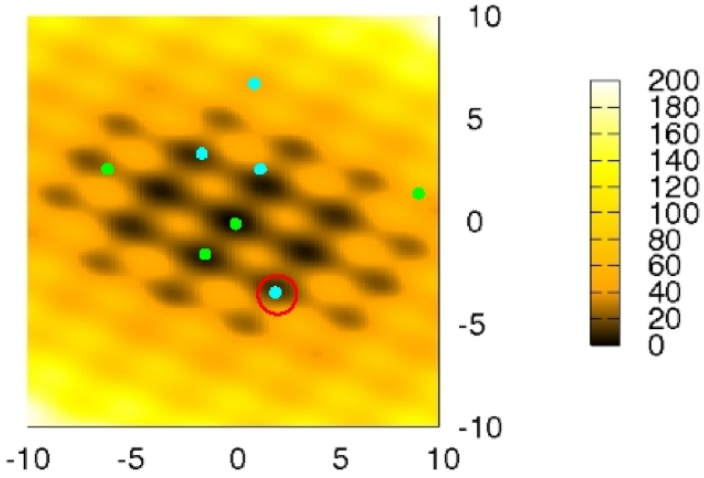


Algorytm kroki wizualizacja

Krok 6



Krok 7



Warianty algorytmu

Algorytmy różnią się ze względu na modyfikowanie prędkości:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{inertia} + \underbrace{\varphi_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{x}_i^t)}_{personal\ \ influence} + \underbrace{\varphi_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{x}_i^t)}_{social\ \ influence}$$

Warianty

Lokalne sąsiedztwo

Każda cząsteczka i ma sąsiedztwo N_i, wówczas

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \varphi_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{x}_i^t) + \varphi_2 \mathbf{U}_2^t(\mathbf{lb}^t - \mathbf{x}_i^t)$$

Ważona inercja

$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + \varphi_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{x}_i^t) + \varphi_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{x}_i^t)$$

Dynamiczna waga inercji

Waga inercji zmniejsza się w czasie od wartości zadanej max do min.

Warianty sąsiedztwa

- Pełne sąsiedztwo: połączenie każdy z każdym.
- Topologia von Neumanna $|N_i| = 4$.
- Ring każdy ma dwóch sąsiadów.

Warianty

Kanoniczne PSO

$$\mathbf{v}_i^{t+1} = \chi[\mathbf{v}_i^t + \varphi_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{x}_i^t) + \varphi_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{x}_i^t)]$$

gdzie χ nazywany jest „constriction factor" i jest stały.

Fully informed PSO

$$\mathbf{v}_i^{t+1} = \chi[\mathbf{v}_i^t + \sum_{p_k \in N_i} \varphi_k \mathbf{U}_k^t(\mathbf{pb}_k^t - \mathbf{x}_i^t)]$$

cząsteczka jest przyciągana przez każdą ze swojego sąsiedztwa.

Warianty

Przykłady innych modyfikacji:

- dynamicznie zmieniające się topologie sąsiedztwa
- zwiększające różnorodność
- zmieniające reguły określania nowych wartości prędkości.
- do rozwiązywania zadań optymalizacji dyskretnej
- dodające elementy innych algorytmów
- ...

Plan wykładu

Kierunki badań naukowych nad PSO

- Dopasowanie algorytmu (jego komponentów) do problemu.
- Zastosowanie do rozwiązywania różnych problemów (kombinatorycznych, dynamicznych, stochastycznych).
- Dobór parametrów (ile cząstek, jaka topologia).
- Odkrycie najlepszego PSO (porównywanie wariantów).
- Nowe warianty (modyfikacje i hybrydy).
- Aspekty teoretyczne (zachowanie cząstek, stagnacja).

Zalety i wady

- Bardzo łatwy algorytm, który skutecznie rozwiązuje problem optymalizacji różnych funkcji.
- Narzędzie blisko związane z metodami A-life i ewolucyjnymi.
- Dużą zaletą jest pamiętanie stanu poprzedniego, co pozwala na eksploracje znanej okolicy, a jednoczenie nie hamuje eksplorowania przestrzeni dalszych.
- Zakres eksploracji zależy od parametrów stochastycznych.

Plan wykładu

Wykorzystana literatura (do samodzielnego studiowania)


J. Kennedy and R. Eberhart
Swarm Intelligence
Morgan Kaufmann, San Francisco, CA, 2001


Marco A. Montes de Oca
PArticle Swarm Optimization
http://iridia.ulb.ac.be/~mmontes/slidesCIL/slides.pdf


J. Kennedy and R. Eberhart
Particle Swarm Optimization
http://www.engr.iupui.edu/~shi/Coference/psopap4.html