# Optymalizacja systemów

https://dlaczegologia.pl/kurs/optym/




Wykład 7. Algorytmy inspirowane naturą

# Heurystyki

- metoda rozwiązywania problemów, dla której nie ma gwarancji znalezienia rozwiązania optymalnego
- korzysta się z nich gdy pełny algorytm jest zbyt kosztowny lub nieznany
- najczęściej stosowane w przypadku zbadanych i znanych problemów
- w przeciwieństwie do algorytmów dokładnych
  - czas działania zazwyczaj nie jest dowolnie długi (w algorytmach dokładnych może być)
  - jedynie w specyficznych przypadkach znajduje rozwiązanie dokładne
  - może w ogóle nie znaleźć rozwiązania
- nie daje nam gwarancji przybliżenia do rozwiązania dokładnego (w przeciwieństwie do algorytmów aproksymacyjnych)
- stosowane w przypadku problemów NP-trudnych i NP-zupełnych
- może być rozwiązaniem początkowym dla innego algorytmu.
- sami stosujemy ją na co dzień.

# Metaheurystyki

- heurystyka, którą można stosować do rozwiązania dowolnego problemu, który da się opisać za pomocą pewnych zdefiniowanych przez algorytm pojęć
- najważniejsza różnica:
  - heurystyki często rozwiązują konkretny problem
  - metaheurystyki rozwiązują szersze spektrum problemów
- zazwyczaj opisują sposób przechodzenia między możliwymi rozwiązaniami
- często możliwe jest dostrajanie ich działania za pomocą różnych parametrów (nie istnieją uniwersalne wartości tych parametrów)

# Metody optymalizacji inspirowane naturą

- są często metaheurystykami lub heurystykami
- wykorzystują analogię do procesów zachodzących w naturze w procesie przeszukiwania przestrzeni rozwiązań
- między innymi
  - błądzenie losowe (*random walk*)
  - symulowane wyżarzanie (*simulated annealing*)
  - algorytmy genetyczne
  - strategie ewolucyjne
  - algorytm mrówkowy (*ant colony optimization*)
  - algorytm pszczeli (*artificial bee colony*)
  - algorytm roju cząsteczek (*particle swarm optimization*)
  - algorytm świetlika (*firefly algorithm*)
  - algorytm kukułki (*cuckoo search*)
  - algorytm nietoperza (*bat algorithms*)
  - algorytm zapylania kwiatów (*flower pollination algorithms*)

# Błądzenie losowe

Dane: $F(x), x_0, D_x, N$

Krok 0: n=0, $x^* = x_n$

Krok 1: Generujemy punkt $x_{n+1}$ zbiorze $D_x$ wg rozkładu jednostajnego

Krok 2: Jeżeli $F(x_{n+1}) < F(x^*)$ podstaw $x^* = x_{n+1}$ „down hill”

Krok 3: Jeżeli $n < N$ podstaw $n = n+1$ idź do kroku 1

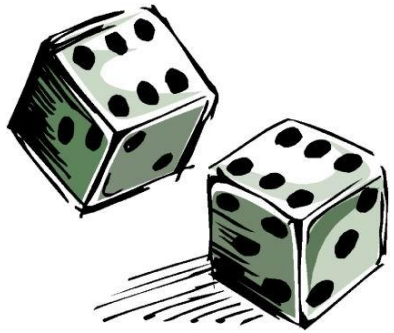
Krok 4: $x^* = x_N$

Zmieniamy krok 2 „Algorytm wspinaczkowy”

Krok 2: Jeżeli $F(x_{n+1}) > F(x^*)$ podstaw $x^* = x_{n+1}$

# Algorytm losowy
# Zalety i wady

- ☑ Zwraca stosunkowo dobre rozwiązanie w krótkim czasie.
- ☑ Dobry do wstępnego przeszukania przestrzeni rozwiązań.
- ! Losowy wybór może być problematyczny, w zależności od dziedziny.
- ! Nie jest skalowalny (rozmiar/wymiary).



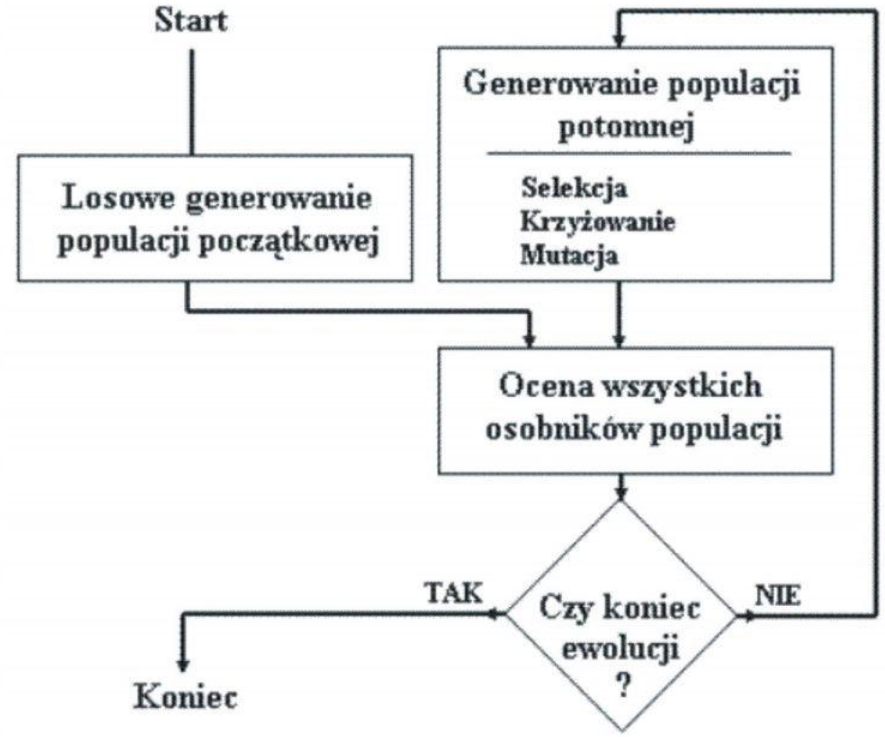
# Algorytm genetyczny

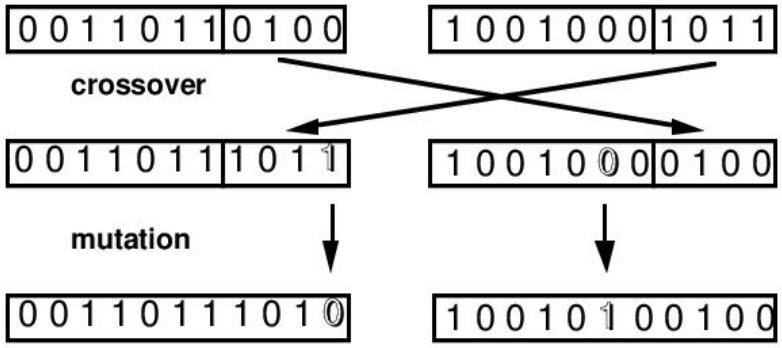
Ewolucja - ciągły proces, polegający na stopniowych zmianach cech gatunkowych kolejnych pokoleń wskutek eliminacji przez dobór naturalny części osobników z bieżącej populacji.

## Mapa pojęć

- Osobnik – jedno z rozwiązań problemu
- Populacja – podzbiór rozwiązań problemu
- Funkcja celu

# Klasyczny algorytm genetyczny



# Algorytm genetyczny

## Etapy algorytmu

### Inicjacja

- Losowy wybór populacji początkowej
  - Algorytmy heurystyczne

### Generowanie populacji potomnej

- Selekcja
- Krzyżowanie
- Mutacja

# Generowanie populacji potomnej

Selekcja - ocena osobników wg funkcji celu

Wybór osobników do reprodukcji – metody:

## Ruletka

- Wielkość pól odpowiadająca wartości funkcji celu
  - Osobniki lepiej przystosowane – większa szansa na reprodukcje

## Ranking

- Sortowanie osobników wg wartości funkcji celu
  - Niekoniecznie wybór osobników najlepszych

## Turniej

- Podział populacji na małe podzbiory
  - Wybór osobnika z podzbioru (wg funkcji celu, losowy)

# Generowanie populacji potomnej

## Krzyżowanie (generowanie populacji potomnej)

- Wymaga 2 osobników rodzicielskich
- Cechy dziedziczone po obu rodzicach
- Różne metody:
  - Dwupunktowe
  - Wielopunktowe

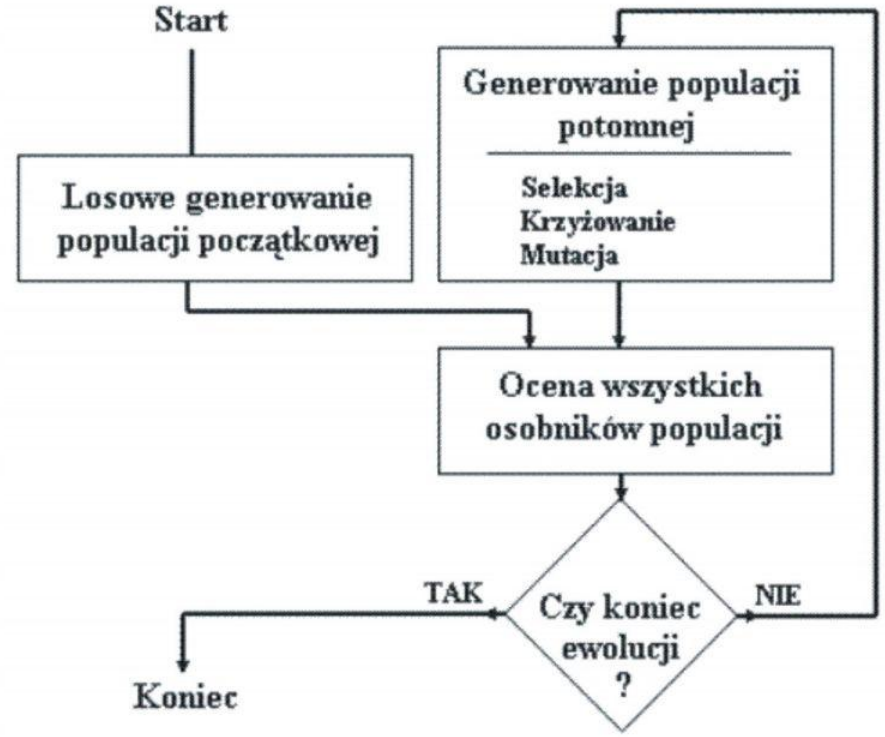
## Mutacja

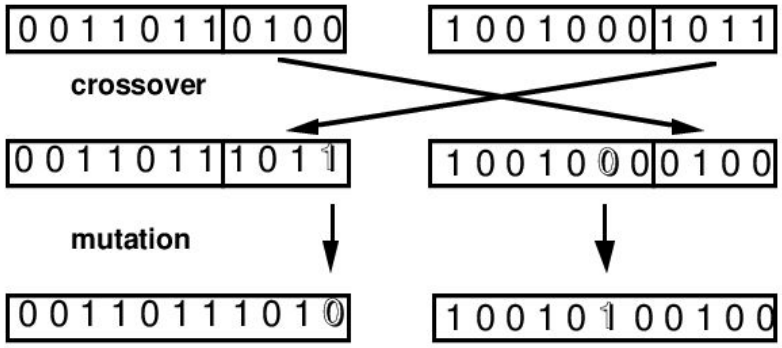
- Wprowadzenie do genotypu osobnika losowych zmian
- Wytworzenie osobnika o cechach nieobecnych w populacji
- Sposób na maksima lokalne

## Warunek stopu

- Liczba iteracji
- Brak poprawy jakości rozwiązania w kolejnych „k” iteracjach

# Klasyczny algorytm genetyczny

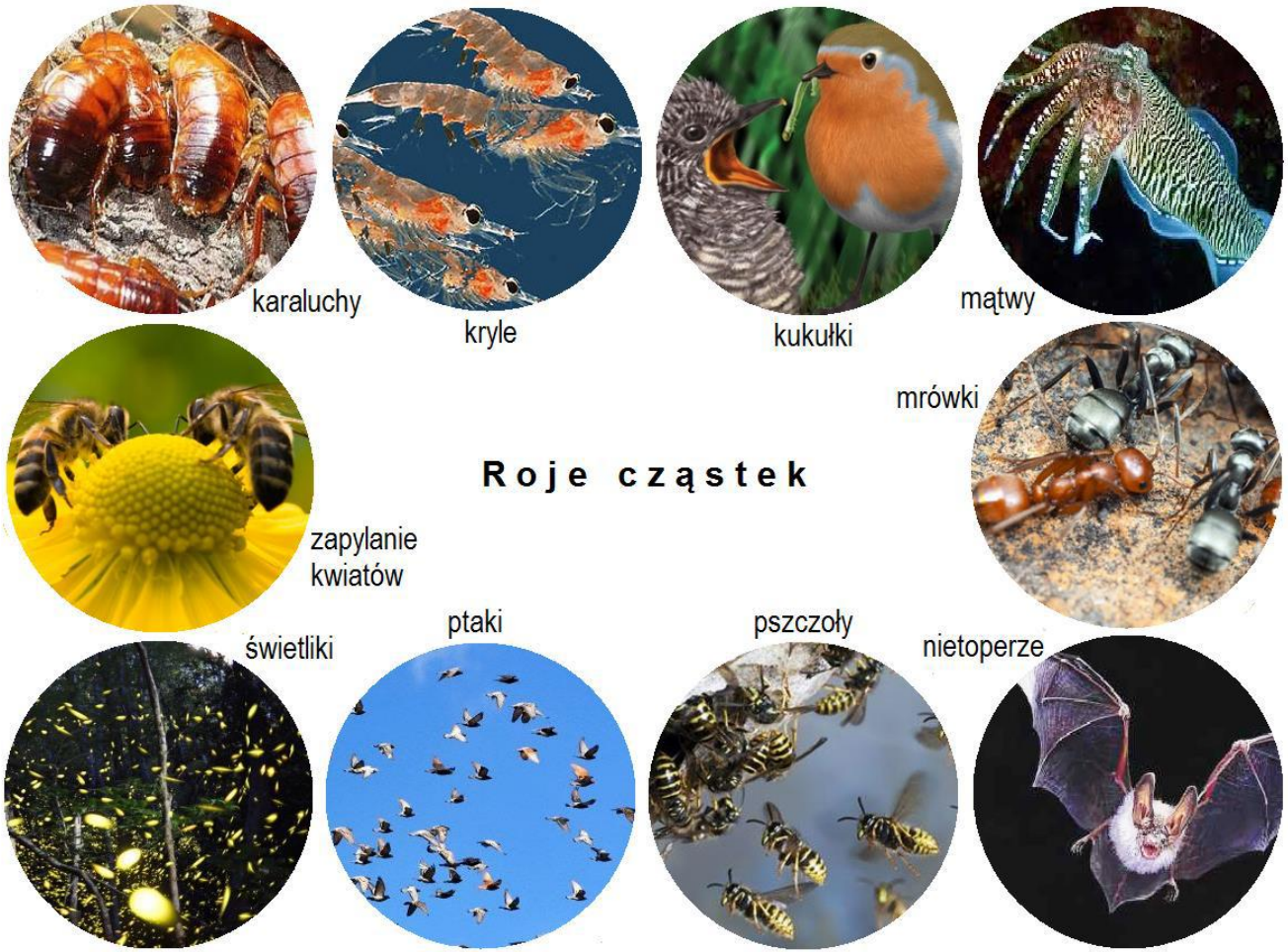


# Algorytm genetyczny - zalety i wady

- zwraca dobre rozwiązania w stosunkowo krótkim czasie
- wymaga wyliczenia oceny każdego rozwiązania w populacji
- nie działa dobrze dla bardziej skomplikowanych problemów
  - wraz ze wzrostem rozmiaru reprezentacji, rośnie przestrzeń rozwiązań
  - rzadkie zmiany w mutacji występują wystarczająco często, by *psuć* rozwiązanie
- często utyka w optimach lokalnych
  - przez brak innych mechanizmów wyjścia z optimum lokalnego niż mutacja
  - mutacja z jednej strony nie może być za duża, żeby nie stracić dobrych rozwiązań
  - z drugiej strony zbyt mała mutacja uniemożliwia wyjście z optimów lokalnych

# Roje cząstek




**Józef LISOWSKI, METODY ROJU CZĄSTEK W OPTYMALIZACJI PROCESÓW TRANSPORTOWYCH I LOGISTYCZNYCH,** Akademia Morska Wydział Elektryczny Katedra Automatyki Okrętowej

# Algorytmy roju cząstek

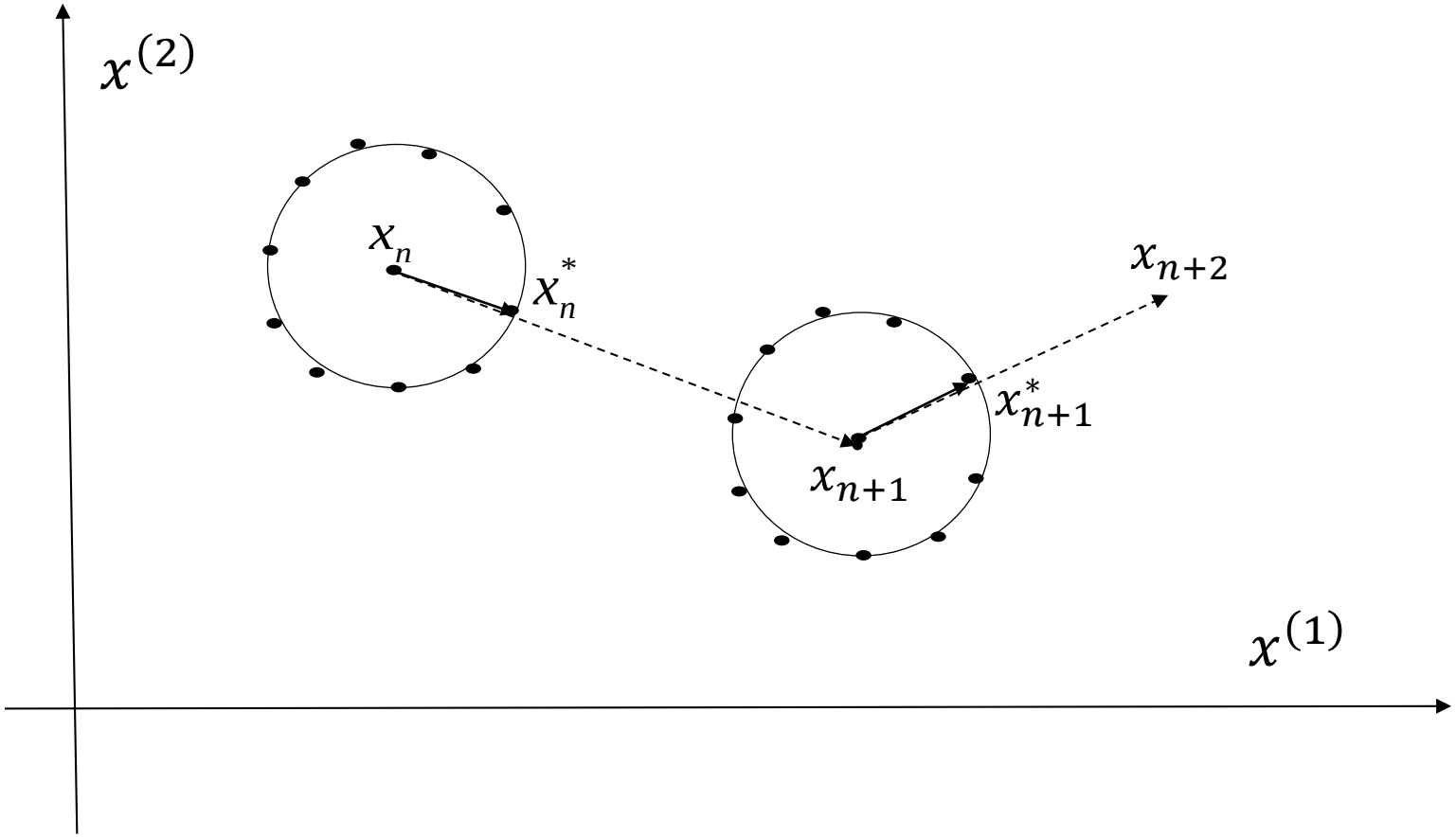
Inteligencję roju cząstek można przedstawić poprzez następujące ich właściwości:

- bliskość – zdolność do obliczania czasu i odległości,
- jakość – zdolność oceny oddziaływania z otoczeniem,
- zróżnicowanie reakcji – zdolność tworzenia różnych reakcji,
- stabilność – zdolność do tworzenia zachowania o umiarkowanej reakcji na zmiany w otoczeniu,
- adaptacja – zdolność do zmian w zachowaniu, jeśli wymusza to otoczenie.

# Algorytm optymalizacji rojem cząstek

- zainspirowany ruchem cząstek w przestrzeni
- każda cząstka reprezentuje pojedyncze rozwiązanie
- cząstka ma przypisane prędkość oraz kierunek, w jakim się porusza
- dla każdej z cząstek zapamiętywane jest jej najlepsze rozwiązanie (lokalne)
- zapamiętywane jest także globalnie najlepsze rozwiązanie
- kierunek poruszania się cząstki w każdej iteracji aktualizowany jest na podstawie najlepszego rozwiązania lokalnego i globalnego

# Poszukiwania losowe – losowy wybór kierunku



# Oznaczenia:

- populacja roju $S=\{x_1, x_2, ..., x_M\}$,
- cząstka $x_i$ należy do $N$-wymiarowej przestrzeni poszukiwań $D$,
- położenie cząstki $x_i=\{x_{i1}, x_{i2}, ..., x_{iN}\}$,
- przemieszczenie cząstki $v_i=\{v_{i1}, v_{i2}, ..., v_{iN}\}$,
- pamięć roju $P$, jako położenie każdej z cząstek o najlepszej wartości funkcji celu $F$: $P=\{p_{i1}, p_{i2}, ..., p_{iN}\}$,
- indeks $g$ numeru cząstki o najlepszej wartości funkcji celu w danej chwili.

Przemieszczanie się roju cząstek można opisać dla $k$-tej iteracji za pomocą następujących zależności:

$$v_{ij}(k) = v_{ij}(k-1) + c_1 R_{1j}\left[p_{ij}(k-1) - x_{ij}(k-1)\right] + c_2 R_{2j}\left[p_{gj}(k-1) - x_{ij}(k-1)\right] \quad (1)$$

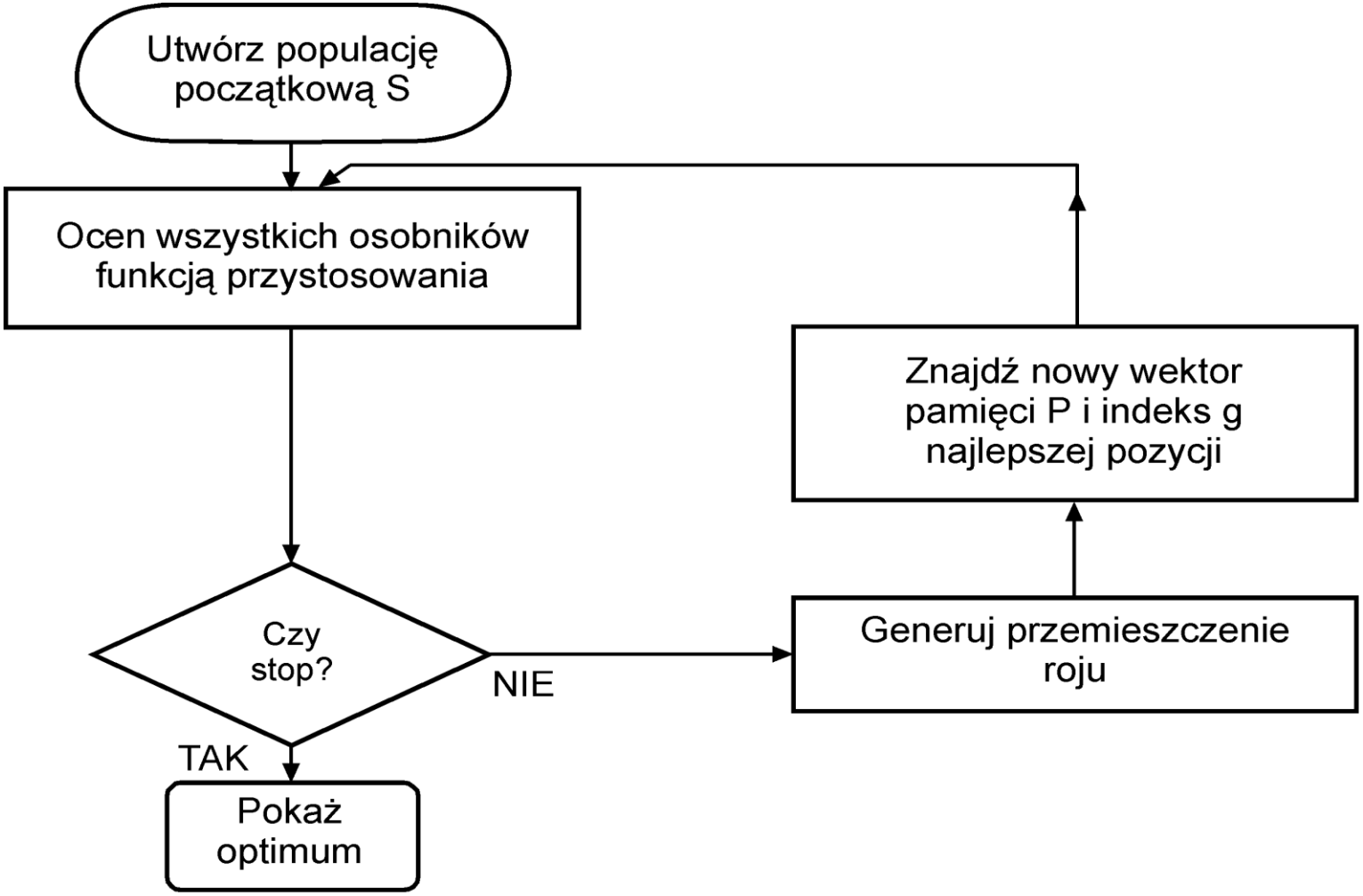
$$x_{ij}(k) = x_{ij}(k-1) + v_{ij}(k) \quad (2)$$

gdzie:

$R_{1j}$, $R_{2j}$ – liczby losowe z przedziału [0;1], liczone dla każdej składowej oddzielnie,

$c_1$, $c_2$ - współczynniki wrażliwości na zmiany funkcji celu optymalizacji $F(k)$.

# Schemat blokowy działania algorytmu roju cząstek



# Algorytm mrówkowy - motywacja

- zainspirowany zachowaniem mrówek poszukujących pożywienia
  - mrówki poszukując pożywienia poruszają się w sposób losowy
  - gdy znajdują pożywienie, wracają do swojej kolonii pozostawiając ślad z feromonów
  - gdy inna mrówka natknie się na ten ślad, przestaje poruszać się losowo i podąża za śladem
  - po pewnym czasie feromony wyparowują, więc ich siła maleje
  - im dłuższa trasa do pożywienia, tym więcej jest czasu, by feromony odparowały
  - krótsze trasy zapewniają zatem, że siła działania feromonów będzie większa
  - ostatecznie wszystkie mrówki poruszać się będą jedną, najlepszą drogą, a pozostałe drogi wyparują

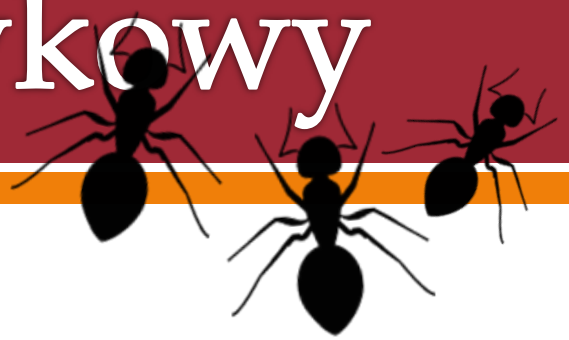
- wykorzystywany do szukania dobrych dróg w grafach
  - zazwyczaj w problemie komiwojażera i pochodnych
  - siła feromonu zależy od jakości znalezionej drogi (np. jej długości)

# Algorytm mrówkowy



1. Mrówki przeszukują przestrzeń losowo.
2. Jeśli natrafiają na „pożywienie”, tj. rozwiązanie – wracają do punktu startu zostawiając ślad feromonu.
3. Natrafiając na ślad feromonu podążają wyznaczoną ścieżką tym samym zwiększając ilość feromonu.
4. Feromon z czasem ulatnia się. Długie dystanse szybciej tracą intensywność feromonu.

# Algorytm mrówkowy



$$P_{i,j} \leftarrow \frac{\tau_{i,j}^{\alpha} \times \eta_{i,j}^{\beta}}{\sum_{k=1}^{c} \tau_{i,k}^{\alpha} \times \eta_{i,k}^{\beta}}$$

,gdzie:

$\tau_{i,j}^{\alpha}$ - wartość feromonu dla ścieżki i - j

$\alpha$ - współczynnik wpływu wartości feromonu ~ 1

$\eta_{i,j}^{\beta}$ - np. 1 / długość ścieżki (dla problemu komiwojażera)

$\beta$ - współczynnik heurystyki $\in$ (2 , 5)

C - rozmiar zbioru rozwiązań

# Algorytm mrówkowy



Zanikanie ilości feromonu na ścieżkach:

Aktualizacja lokalna:

$\tau_{i,j} \leftarrow (1-\sigma) \times \tau_{i,j} + \sigma \times \tau_{i,j}^{0}$ , gdzie:

$\sigma$ - współczynnik historyczny ~ 0.1

$\tau_{i,j}^{0}$ - poprzednia wartość feromonu

Aktualizacja globalna:

$\tau_{i,j} \leftarrow (1-\rho) \times \tau_{i,j} + \rho \times \Delta\tau i, j$

$\rho$ - współczynnik zanikania ~ 0.1

$\Delta\tau i, j$ - nagroda za bycie częścią najlepszej ścieżki globalnej ( =0 dla pozostałych ścieżek)

# Algorytm mrówkowy

1. Inicjalizacja:
   a. wszystkie ścieżki w grafie inicjalizowane są niewielką ilością feromonu
   b. mrówki umieszczane są losowo w różnych punktach (miastach)
2. Mrówki wędrują między punktami (miastami):
   a. wyznaczanie ścieżek
      i. każda mrówka wędruje do sąsiadującego punktu (miasta) wybierając losowo drogę na podstawie prawdopodobieństwa wyznaczanego bazując na ilości feromonu danej krawędzi
      ii. mrówki posiadają pamięć, która nie pozwala im odwiedzić danego punktu (miasta) ponownie
      iii. mrówki kończą wędrowanie po osiągnięciu celu
   b. aktualizacja feromonów na poszczególnych krawędziach
      i. dodawana jest ilość feromonu proporcjonalna do długości drogi przebytej przez mrówkę
      ii. odejmowana jest ilość feromonu związana z odparowywaniem
   c. jeśli niespełniony jest warunek stopu, kroki są powtarzane

# Algorytm mrówkowy - zalety i wady

- stosowany w różnych problemach (nie tylko do poruszania się po grafie):
  - problem plecakowy
  - wykrywanie krawędzi na obrazie
- dostosowuje się do zmiennych warunków
  - może służyć do wyznaczania tras w przypadku zamknięć dróg
  - w tym zastosowaniu nie zapomina poprzednio znalezionych dobrych rozwiązań
- stosunkowo duża liczba parametrów

# Algorytm pszczeli

- zainspirowany sposobem wyszukiwania pokarmu przez pszczoły
  - pszczoły robotnice poszukują źródeł pokarmu wybierając losowy kierunek i obszar poszukiwań
  - po znalezieniu źródła pożywienia, wracają do ula, by poinformować o tym innych
  - pszczoły chcąc przekazać informację o innym, wykonują specyficzny taniec (*waggle dance*), który jest dłuższy i bardziej ekspresyjny im pokarmu jest więcej
  - pozostałe pszczoły oceniają taniec zwiadowców i decydują się podążać za najlepiej tańczącym
  - także pszczoły eksplorujące poprzednio inne źródło, mogą w pewnym momencie zmienić zdanie, gdy uznają, że inny obszar rokuje na większą ilość pokarmu niż ten, który obecnie eksploatują
- wyraźnie wyznacza dwa etapy: eksploracji i eksploatacji
- analogie zastosowane w algorytmie:
  - lokalizacja pokarmu - pojedyncze rozwiązanie problemu
  - ilość pokarmu - jakość danego rozwiązania (wartość funkcji oceny)
  - liczba pszczół - liczba rozwiązań w populacji

# Algorytm pszczeli

- inicjalizacja - generowanie losowych rozwiązań
- pracownice (*employed bees*)
  - dla każdej pszczoły generowane są losowe rozwiązania w sąsiedztwie aktualnego (sąsiedztwo zdefiniowane jako rozwiązania, których reprezentacja wektorowa jest bliska)
  - pszczoła porzuca poprzednie rozwiązanie jeśli nowe jest lepsze
- obserwatorzy (*onlooker bees*) - eksploatacja
  - pszczoły wybierają jedno z rozwiązań proponowanych przez pracownice
  - wybór rozwiązania dokonywany jest losowo, lecz proporcjonalnie do jego oceny (lepsze rozwiązania mają większe prawdopodobieństwo wyboru)
  - każda pszczoła ewaluuje jedno rozwiązanie w sąsiedztwie wybranego przez siebie (sąsiedztwo zdefiniowane identycznie jak w przypadku pracownic)
- zwiadowcy (*scout bees*) - eksploracja
  - pracownice, które nie zauważyły poprawy w otoczeniu swojego rozwiązania stają się zwiadowcami
  - generują one całkowicie losowe rozwiązanie

# Algorytm pszczeli



Wyróżniamy grupy:

- *Zwiadowców*
- *Obserwatorów*
- *Pracownice*

# Algorytm pszczeli



***Zwiadowcy*** – przeszukują rozwiązania w sposób losowy

***Pracownice*** – przynoszą informacje o jakości rozwiązań do których są obecnie przypisani.

***Obserwatorzy*** – wybierają następne miejsce na podstawie informacji od *zatrudnionych*.

# Algorytm pszczeli



*Wyznaczanie źródeł pożywienia:*

$x_{ij} = x_{\min j} + rand[0,1](x_{\max j} - x_{\min j})$ , gdzie

$X_{ij}$ – źródło pożywienia

i = 1,…,P (P – liczba źródeł pożywienia)

j = 1,…,D (D – rozmiar przestrzeni rozwiązań)

# Algorytm pszczeli



*Poszukiwanie pożywienia (pszczoły pracujące):*

$x'_{ij} = x_{ij} + rand[-1,1](x_{ij} - x_{kj}), i=1,\ldots,P,\ k \neq i$

, gdzie:

$k \in \{1, 2, \ldots, P\}$ oraz $j \in \{1, 2, \ldots, D\}$

są dwoma losowo wyznaczonymi indeksami

Jeśli $F(x_i') > F(x_i)$ to i-ta pszczoła zmienia źródło pożywienia

# Algorytm pszczeli



Poszukiwanie pożywienia (obserwatorzy):

$$p_i = \frac{F(x_i)}{\sum_{i=1}^{P} F(x_i)}, \qquad i = 1, \ldots, P$$

Obserwatorzy wybierają swój cel na podstawie informacji od pszczół pracujących, a następnie przeprowadzają lokalne przeszukiwanie w jego sąsiedztwie.

Analogicznie do symulowanego wyżarzania- sąsiedztwo jest zawężane wraz z czasem działania algorytmu.

# Algorytm pszczeli



☑ Szybszy niż algorytmy genetyczny czy mrówkowy dla wielu wymiarów.

❗ Stworzony dla funkcji ciągłych

❗ Podobieństwo do zrównoleglonego algorytmu losowego – ryzyko wielokrotnego sprawdzania tego samego rozwiązania.

# Algorytm świetlika

- zainspirowany zjawiskiem bioluminescencji robaczków świętojańskich
  - jaśniej świecące świetliki przyciągają te o słabszym świetle
  - jeżeli w otoczeniu nie ma osobników o jaśniejszym świetle, wtedy porusza się losowo
- świetlik reprezentuje pojedyncze rozwiązanie
- jasność świecenia jest proporcjonalna do jakości rozwiązania
- bardzo zbliżony w swym działaniu do algorytmu optymalizacji rojem cząstek
  - obie cząstki poruszają się w kierunku globalnego rozwiązania
  - cząstka dodatkowo porusza się w kierunku swojego najlepszego lokalnego rozwiązania
  - świetlik może poruszać się w kierunku najlepszego rozwiązania innego świetlika

# Symulowane wyżarzanie



1. Losowo wybiera sąsiadującą wartość.
2. Zmniejsza temperaturę (symuluje upływ czasu).
3. Gdy wylosowana wartość jest lepsza, zastępuje aktualną.
4. Jeśli nie jest lepsza oblicza P(e,e’,T)= $\exp(\frac{e-e'}{T})$, gdzie:

e - energia systemu w obecnym stanie (*)

e’ - energia systemu w stanie wylosowanym

T - temperatura

* energia systemu = wartość funkcji celu

5. Obecna wartość jest zastępowana kandydującą, jeśli P(e,e’,T) > x, gdzie:

x- liczba losowa z przedziału (0,1)

# Symulowane wyżarzanie - informacje

- Przy współczynniku T = 0 algorytm staje się zachłannym.
- Początkowo chętniej akceptuje gorsze wartości.
- Wraz z wygaszaniem T->0 algorytm staje się wymagający, częściej akceptuje lepsze wartości.
- Opcjonalnie można resetować algorytm, by cyklicznie powracał do aktualnie najlepszego punktu.
- Zakres sąsiadujących próbek warto zmniejszać z czasem.



# Symulowane wyżarzanie – zalety i wady

- ☑ Istnieje wariant algorytmu dla optymalizacji funkcji ciągłych.
- ☑ Istnieje wiele ulepszeń m.in. zrównoleglone wyżarzanie lub resetowanie temperatury z zapamiętaniem najlepszego rozwiązania.

- ! W najgorszym przypadku algorytm gorszy niż pełne przeszukanie.
- ! Problem z oszacowaniem okresu wyżarzania T.



# Bibliografia


- *Clever Algorithms: Nature-Inspired Programming Recipes*, Jason Brownlee
- *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning , Shumeet Baluja,* School of Computer Science Carnegie Mellon University Pittsburgh,1994
- The Bees Algorithm – A Novel Tool for Complex Optimisation Problems, D.T. Pham, A. Ghanbarzadeh, E. Koç et. al, Cardiff University, 2006
- Zastosowanie Algorytmów Rojowych do Optymalizacji Parametrów w Modelach Układów Regulacji, Mirosław Tomera, Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej Nr 46, 2015
- Automatic Tuning of a Retina Model for a Cortical Visual Neuroprosthesis Using a Multi-Objective Optimization Genetic Algorithm, Antonio Martínez-Álvarez, Rubén Crespo-Cano, Ariadna Díaz-Tahoces et. al., International Journal of Neural Systems 26/7, 2016

# Dziękuję za uwagę

