# Algorytmy ewolucji różnicowej (ang. differential evolution -DE) oraz roju cząstek (ang. particle swarm optimization -PSO)

# Ewolucja różnicowa - wstęp

- Stosunkowo nowy (połowa lat 90tych) algorytm optymalizacji numerycznej (autorzy: R. Storn i K. Price).

- Osobniki są L-wymiarowymi wektorami liczb rzeczywistych. Fenotyp jest identyczny z genotypem.

- Mutacja polega na perturbacji losowo wybranego wektora o różnicę dwóch innych wektorów pomnożoną przez stały współczynnik.

- Algorytm jest bardzo prosty i działa bardzo dobrze na typowych funkcjach testowych, choć nie do końca wiadomo dlaczego.

- Algorytm wymaga tylko trzech parametrów, z których jeden (S) jest rozmiarem populacji, drugi (F) steruje mutacją, a trzeci (CR) krzyżowaniem.

# Selekcja i sukcesja

- Algorytm operuje na populacji S osobników $x_1, x_2, ..., x_S$, gdzie $x_i \in \Re^L$
- W każdym kroku algorytmu dla każdego osobnika $x_i$ tworzony jest ***osobnik próbny*** $u_i$.
  - $u_i$ powstaje poprzez zastosowanie operatorów mutacji i krzyżowania.
- Następnie dopasowanie osobnika próbnego $u_i$ jest porównywane z dopasowaniem rodzica $x_i$.
  - Jeżeli dopasowanie $u_i$ jest lepsze (mniejsze bądź większe w zależności od rodzaju problemu optymalizacyjnego) niż $x_i$, $u_i$ zastępuje $x_i$ w populacji.
  - Jeżeli dopasowanie $u_i$ jest gorsze niż $x_i$, $u_i$ jest odrzucany.
- Ten schemat możemy zaimplementować utrzymując dwie populacje, każdą składającą się z S wektorów L-elementowych: populację główną i populację próbną.

# Mutacja

- Wynikiem mutacji jest nowy wektor $v_i$ otrzymany w sposób następujący:

$$v_i = x_{r_1} + F * (x_{r_2} - x_{r_3}).$$

gdzie $0 \leq F \leq 1$ jest stałym parametrem, zwanym ***współczynnikiem amplifikacji*** (ang. amplification factor), $r_1, r_2, r_3$ to trzy losowo wygenerowane numery osobników ze zbioru {1,2,...,S}, przy czym spełniona jest zależność $i \neq r_1 \neq r_2 \neq r_3$ (trzy liczby losowe nie mogą się powtarzać i każda musi być różna od i)

- $v_i$ jest nazywany osobnikiem ***mutantem***.
- Typowa wartość F to 0.5.
- Uwaga: rodzic $x_i$ nie bierze udziału w procesie mutacji.

# Krzyżowanie

- Wynikiem krzyżowania operującego na rodzicu $x_i$ i mutancie $v_i$ jest osobnik próbny $u_i$, który następnie w procesie sukcesji zostanie porównany z $x_i$. Każdy element $u_{i,j}$ (j=1,2,..L) wektora $u_i$ jest wyznaczany w sposób następujący:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{jeżeli } rnd_j < CR \text{ lub } j = d \\ x_{i,j} & \text{w przeciwnym wypadku} \end{cases}$$

gdzie $rnd_j$ jest liczbą losową z przedziału [0,1) losowaną niezależnie dla każego j. $0 \le CR \le 1$ jest stałym parametrem algorytmu, d jest losowym numerem elementu wektora losowanym ze zbioru {1,2,...,L}.

- CR oznacza prawdopodobieństwo przejścia elementu z wektora mutanta $v_i$ do wektora próbnego $u_i$. CR=1 → wszystkie elementu wektora próbnego $u_i$ pochodzą od mutanta $v_i$. CR=0 → wszystkie (z wyjątkiem jednego, co zapewnia warunek po „lub”) elementy wektora próbnego $u_i$ pochodzą od rodzica $x_i$.
- Możemy stwierdzić że operator krzyżowania „miesza” losowo elementy rodzica $x_i$ i mutanta $v_i$, tworząc wektor próbny $u_i$.

# Inne schematy mutacji

- W literaturze stosuje się następujący schemat oznaczania wariantów algorytmu ewolucji różnicowej:

  DE/x/y/z, gdzie

  - z to schemat krzyżowania obecnie bin.
  - y to liczba różnic wektorów stosowanych w mutacji.
  - x to jest wektor poddany mutacji może być: rand (wektor losowy), best (najlepszy osobnik w populacji), current (wektor bieżący, czyli $x_i$)

- Schemat opisany przeze mnie do DE/rand/1/bin.
- Przykłady innych schematów:

  DE/best/2/bin

$$v_i = x_{Best} + F * (x_{r_1} - x_{r_2}) + F * (x_{r_3} - x_{r_4}).$$

  DE/current/1/bin

$$v_i = x_i + F * (x_{r_1} - x_{r_2}).$$

# Schemat mutacji current-to-best

- W tym schemacie mutant jest generowany w sposób następujący:

$$v_i = x_i + F * (x_{Best} - x_i) + F * (x_{r_1} - x_{r_2}).$$

a więc osobnik rodzicielski $x_i$ jest wykorzystywany w procesie mutacji.

# Dobór parametrów

- Algorytm posiada tylko trzy parametry: F, CR,S.
- Zazwyczaj $0.4 \leq F \leq 0.9$.
- W literaturze spotyka się propozycje $5*L \leq S \leq 10*L$, ale prowadziłoby to do bardzo dużych rozmiarów populacji dla funkcji wielowymiarowych. Proponuję w swoich eksperymentach przyjąć na początek $S \approx 50$..
- $0 \leq CR \leq 1$. Proponuję zacząć od bardzo małe (bliskie 0) oraz bardzo duże (bliskie 1) wartości CR.
- Zaproponowano również schematy adaptacji i samoadaptacji parametrów algorytmu. Np. Dla każdego mutanta losuj F z rozkładu N(0.5,0.3).

# Optymalizacja metodą roju cząstek (ang. particle swarm optimization - PSO)

- Algorytm wynaleziony w 1995 r (R. Kenedy i R. Eberhart).
- Algorytm optymalizacji oparty na populacji osobników podobnie jak algorytmy ewolucyjne, choć nie jest to algorytm ewolucyjny sensu stricto.
- Zainspirowany przez stadne zachowania zwierząt (ryby, ptaki).



- Indywidualni członkowie roju mogą skorzystać z odkryć (i poprzednich doświadczeń) innych członków roju.

# PSO – ogólna idea

- Algorytm operuje na ***roju*** (populacji) cząstek.
- Każda cząstka jest umieszczona na ***pozycji*** w przestrzeni rozwiązań. Pozycja cząstki (wektor z przestrzeni $R^L$) podlega ocenie przez funkcję dopasowania.
- Każda cząstka porusza się w przestrzeni cech z pewną ***prędkością*** (ang. velocity). Prędkość cząstki, to również wektor w $R^L$.
  - W każdej iteracji do wektora reprezentującego pozycję cząstki jest dodawany wektor reprezentujący jej prędkość.
- Prędkość cząstki ulega zmianie. Na jej wpływ ma pozycja o największym dopasowaniu odkryta przez cały rój oraz pozycja o największym dopasowaniu odkryta przez cząstkę.

# PSO - oznaczenia

- Dla każdej cząstki i w roju (i=1,2,...,S)
  - $x_i$ jest pozycją cząstki $x_i \in \Re^L$
  - $f(x_i)$ jest dopasowaniem cząstki
  - $v_i$ jest prędkością cząstki $v_i \in \Re^L$
  - $p_i$ jest najlepszą pozycją spośród wszystkich pozycji odwiedzonych przez cząstkę, a $pbest_i$ jest dopasowaniem dla tej pozycji.
  - g jest najlepszą pozycją spośród wszystkich pozycji odwiedzonych przez wszystkie cząstki roju, a gbest jest dopasowaniem na tej pozycji.
- U(0,φ) jest wektorem L-elementowym, w którym każdy element jest niezależnie losowaną liczbą losową z przedziału [0,φ].
- × oznacza mnożenie dwóch wektorów element po elemencie np.

$$\begin{bmatrix}1\\2\\3\end{bmatrix} \times \begin{bmatrix}1\\2\\3\end{bmatrix} = \begin{bmatrix}1\\4\\9\end{bmatrix}$$

# PSO - algorytm

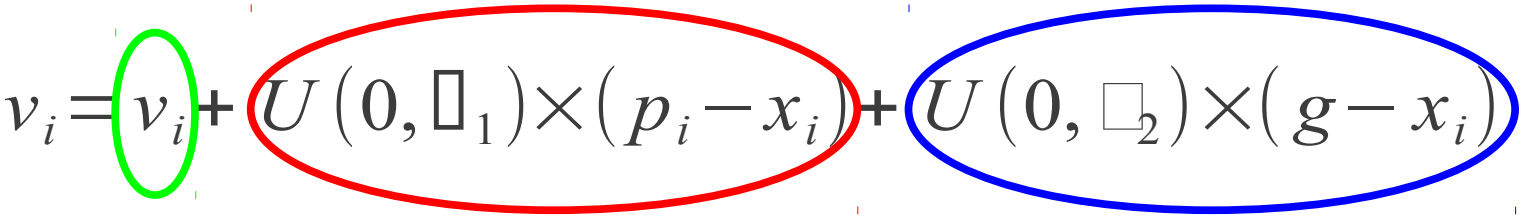
- Zainicjalizuj losowo pozycje i prędkości wszystkich cząstek.
- Dopóki nie warunek stopu
  - Dla każdej cząstki i
    - Oblicz dopasowanie $f(x_i)$.
    - Jeżeli $f(x_i)$ jest lepsze niż $pbest_i$ to $pbest_i=f(x_i)$ $p_i=x_i$.
    - Jeżeli $f(x_i)$ jest lepsze niż gbest to $gbest=f(x_i)$ $g=x_i$.
  - Dla każdej cząstki i
    - Uaktualnij prędkość cząstki wg. wzoru:

      $$v_i = v_i + U(0, \varphi_1) \times (p_i - x_i) + U(0, \varphi_2) \times (g - x_i)$$

      gdzie $\varphi_1$ oraz $\varphi_2$ są stałymi parametrami nazywanymi ***współczynnikami akceleracji***.
    - Uaktualnij pozycję cząstki wg wzoru:

      $$x_i = x_i + v_i$$
- Zwróć g jako najlepsze rozwiązanie, a gbest jako dopasowanie najlepszego rozwiązania

# Reguła uaktualniająca prędkość cząstki



- Składnik momentu – sprawia, że cząstka stara się kontynuować ruch w dotychczasowym kierunku.
- Składnik kognitywny – sprawia, że cząstka stara się powrócić do najlepszej swojej pozycji.
- Składnik socjalny – sprawia, że cząstka jest przyciągana do najlepszej pozycji odnalezionej jak dotąd przez rój.

# Dobór współczynników akceleracji

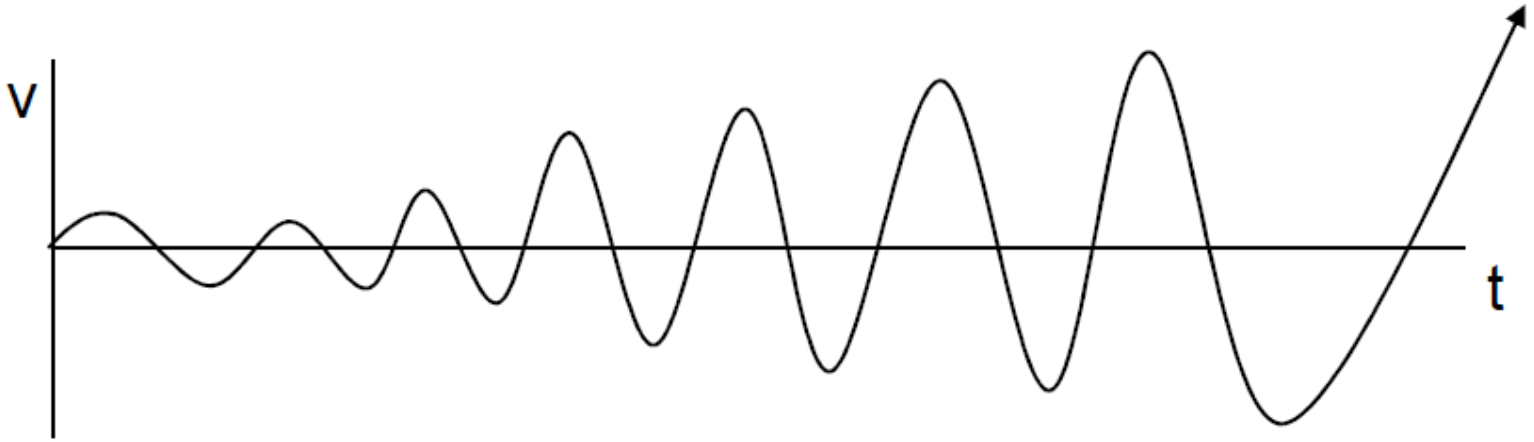
$$v_i = v_i + U(0, \square_1) \times (p_i - x_i) + U(0, \square_2) \times (g - x_i)$$

- $\varphi_1 > 0$ oraz $\varphi_2 = 0$ Każda cząstka realizuje stochastyczny algorytm wspinania się na wzgórze (ang. hill climbing).
- $\varphi_1 = 0$ oraz $\varphi_2 > 0$ Cały rój realizuje stochastyczny algorytm wspinania się na wzgórze.
- $\varphi_1 = \varphi_2 > 0$ Cząstka przyciągana jest przez średnią z $p_i$ oraz g.
- $\varphi_1 > \varphi_2$ Zalecane w przypadku problemów multimodalnych.
- $\varphi_1 < \varphi_2$ Zalecane w przypadku problemów unimodalnych.
- niskie $\varphi_1, \varphi_2$ Gładkie trajektorie cząstek.
- wysokie $\varphi_1, \varphi_2$ Gwałtowne trajektorie cząstek.

- Zaproponowano również adaptacyjny dobór współczynników akceleracji (np. zmniejszanie ich wraz z postępem algorytmu).

# Problemy z oryginalnym algorytmem

- Współczynniki akceleracji powinny być dostatecznie wysokie (w przeciwnym wypadku mamy b. wolną zbieżność).
- Jednakże duże współczynniki akceleracji prowadzą do niestabilnego zachowania systemu, przy którym prędkość „eksploduje”.




- Próbowano sobie z tym radzić ograniczając składowe wektora prędkości do przedziału $[-v_{MAX}, v_{MAX}]$, ale nie chroni to przed opuszczeniem przez cząstkę przestrzeni poszukiwań i dodaje kolejny, trudny w doborze parametr $v_{MAX}$.

# Algorytm PSO z czynikiem ściśku (ang. constriction factor)

- Zmiana prędkości odbywa się według wzoru:

$$v_i = \chi\left(v_i + U(0, \varphi_1) \times (p_i - x_i) + U(0, \varphi_2) \times (g - x_i)\right)$$

gdzie $\varphi = \varphi_1 + \varphi_2 > 4$ oraz $\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}$

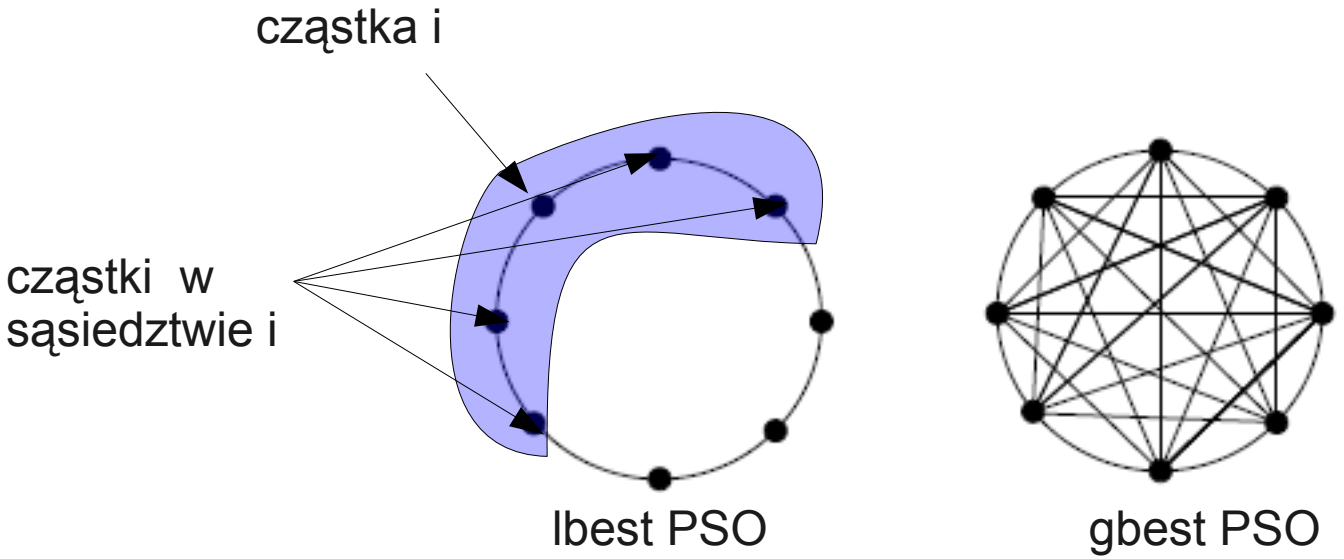
- Reguła zapobiega eksplozji prędkości cząstki, umożliwia zbieżność i pozwala na pozbycie się parametru $v_{MAX}$.

# Sąsiedztwa cząstek i algorytm lbest PSO

- Opisana przeze mnie reguła uaktualniania prędkości dotyczy strategii nazwanej gbest (globally best) PSO w której każda cząstka zna położenie i dopasowanie wszystkich cząstek w roju.
- W problemach wielomodalnych może lepiej się sprawdzać strategia lbest (locally best), w której cząstka, oprócz swojego położenia zna położenia i dopasowania cząstek w swoim sąsiedztwie (z reguły nadanym przez pewną topologię, np.. pierścienia). Reguła zmiany prędkości przyjmuje postać:

$$v_i = \chi\left(v_i + U(0, \square_1)\times(p_i - x_i) + U(0, \square_2)\times(g_i - x_i)\right)$$

gdzie $g_i$ to pozycja o największym dopasowaniu w sąsiedztwie cząstki i, przy czym sąsiedztwo zazwyczaj obejmuje 10%-15% cząstek w roju.



# Literatura dodatkowa

- R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, 11(4), s. 341-359, 1997, artykuł wynalazców DE, ponad 3000 cytowań.
- S. Das, P.N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation,15(1), s. 4-31, 2011, artykuł przeglądowy na temat najnowszych trendów w DE.
- J. Kennedy, R. Eberhart, Particle swarm optimization, IEEE International Conference on Neural Networks, 1995. Proceedings, s. 1942-1948, 1995, artykuł wynalazców PSO, ponad 14000 cytowań.
- R. Poli, J. Kennedy, T. Blackwell: Particle swarm optimization. Swarm Intelligence 1(1), s. 33-57, 2007, artykuł przeglądowy na temat PSO.
- http://www.icsi.berkeley.edu/~storn/code.html strona R. Storna o DE, kod źródłowy i applety w wielu językach programowania.
- http://www.particleswarm.info/ Strona z tutorialami i programami n.t. PSO.