Algorytmy ewolucyjne

Dr inż. Michał Bereta
p. 144 / 10, Instytut Modelowania Komputerowego
mbereta@pk.edu.pl
beretam@torus.uck.pk.edu.pl
www.michalbereta.pl

Algorytmy ewolucyjne

Warunki zaliczenia:

Wykład – 18 h : 4 x 4 h + Egzamin

Laboratoria – 9h (dr inż. Adam Mrozek)

Algorytmy ewolucyjne

Literatura:

„Algorytmy genetyczne + struktury danych = programy ewolucyjne”, MICHALEWICZ ZBIGNIEW

„Jak to rozwiązać czyli nowoczesna heurystyka”, MICHALEWICZ ZBIGNIEW

„Wykłady z algorytmów ewolucyjnych”, Jarosław Arabas

Algorytmy ewolucyjne

Zakres tematów:

- Podstawowe pojęcia algorytmów ewolucyjnych
- Klasyczny algorytm genetyczny
- Podejście ewolucyjne w rozwiązywaniu problemów
- Projektowanie algorytmów ewolucyjnych
- Rodzaje algorytmów ewolucyjnych
- Problemy z ograniczeniami

Algorytmy ewolucyjne

Zakres tematów:

- Optymalizacja wielokryterialna
- Optymalizacja kombinatoryczna
- Systemy koewolucyjne
- Zastosowania algorytmów ewolucyjnych
- Nowoczesne algorytmy ewolucyjne: algorytmy rojowe, sztuczne systemy immunologiczne, …

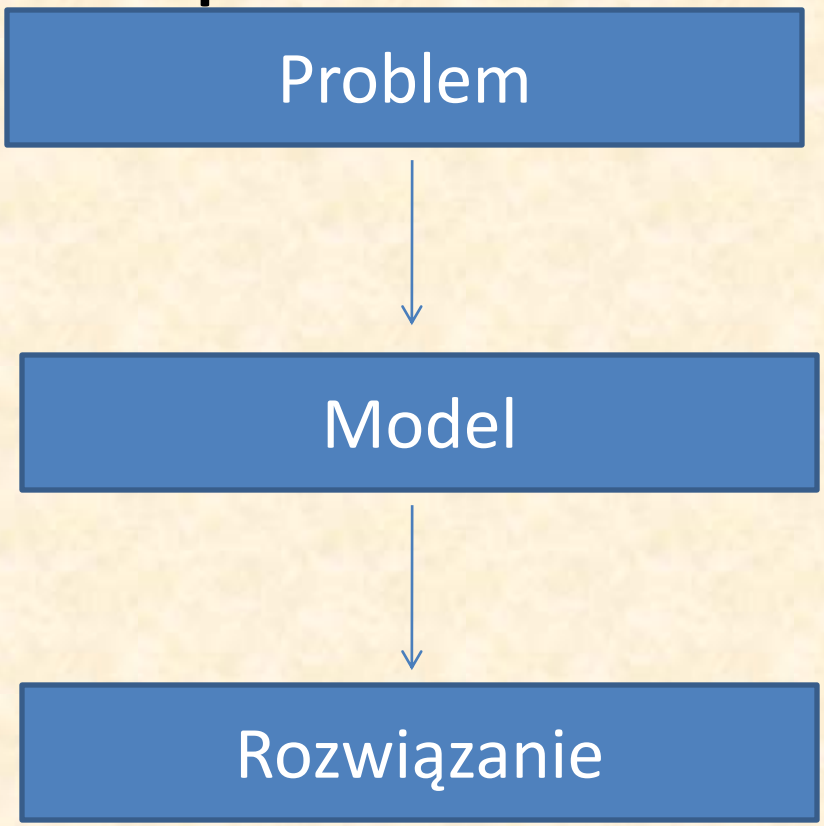
Trudności w rozwiązywaniu problemów

- Liczba możliwych rozwiązań w przestrzeni poszukiwania jest tak duża, że nie można zastosować przeszukiwania wyczerpującego
- Problem jest tak skomplikowany, że aby otrzymać jakiekolwiek rozwiązanie musimy stosować tak uproszczone modele, że żadne otrzymane rozwiązanie nie jest praktycznie użyteczne

Trudności w rozwiązywaniu problemów

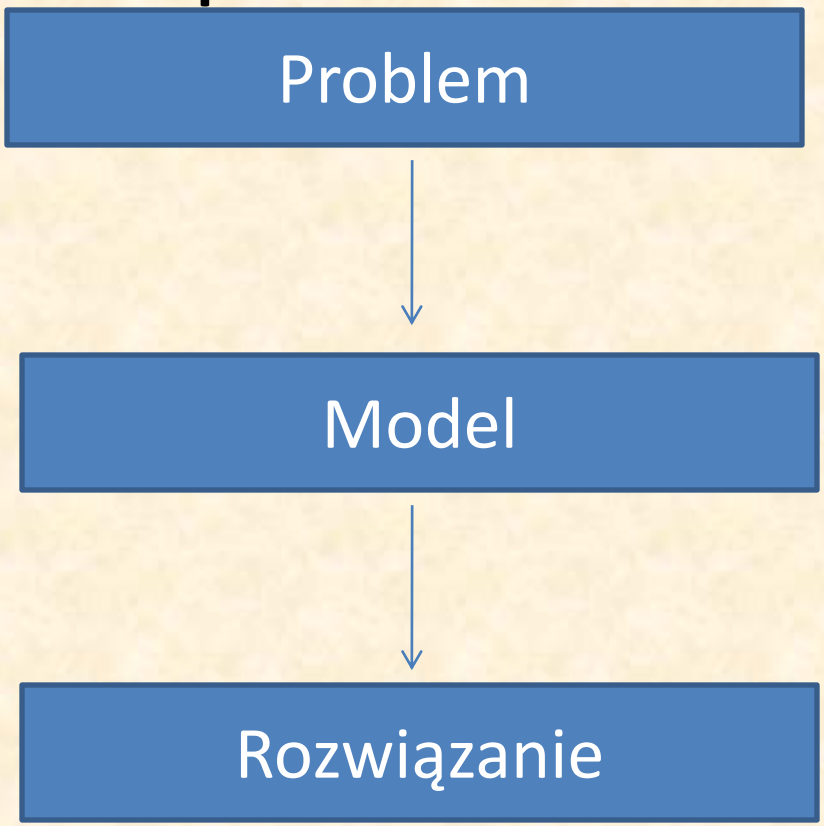
- Funkcja oceny, która opisuje jakość dowolnego rozwiązania jest zakłócana lub zmienia się w czasie – wymagany jest szereg rozwiązań
- Możliwe rozwiązania są mocno ograniczone – trudno wygenerować choć jedno dopuszczalne rozwiązanie, a co dopiero optymalne
- Psychologiczne – nawyki myślenia pewnymi kategoriami, schematy, itp.

Trudności w rozwiązywaniu problemów



Jak bardzo uproszczony może być model, by rozwiązanie było wciąż użyteczne?

Trudności w rozwiązywaniu problemów



Co jest lepsze:

-Dokładne rozwiązanie przybliżonego (czyli prostszego) modelu

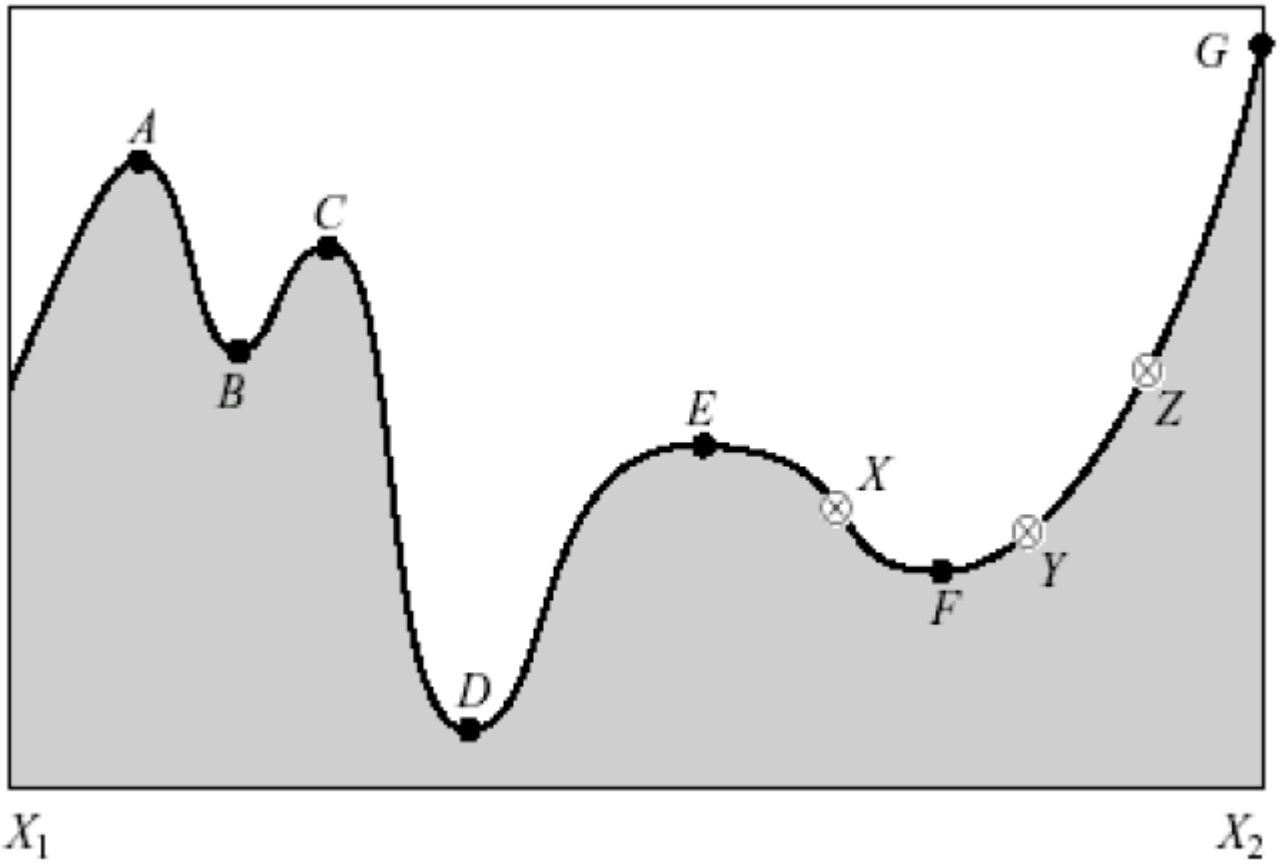
czy

- Przybliżone rozwiązanie dokładnego modelu?

Trudności w rozwiązywaniu problemów

- **<u>Rozwiązania optymalne nie są wymagane!</u>**
- W praktyce priorytetem jest posiadanie dobrego, lecz niekoniecznie optymalnego rozwiązania, lecz odpowiednio szybko – np. przed konkurencją

Problemy optymalizacji

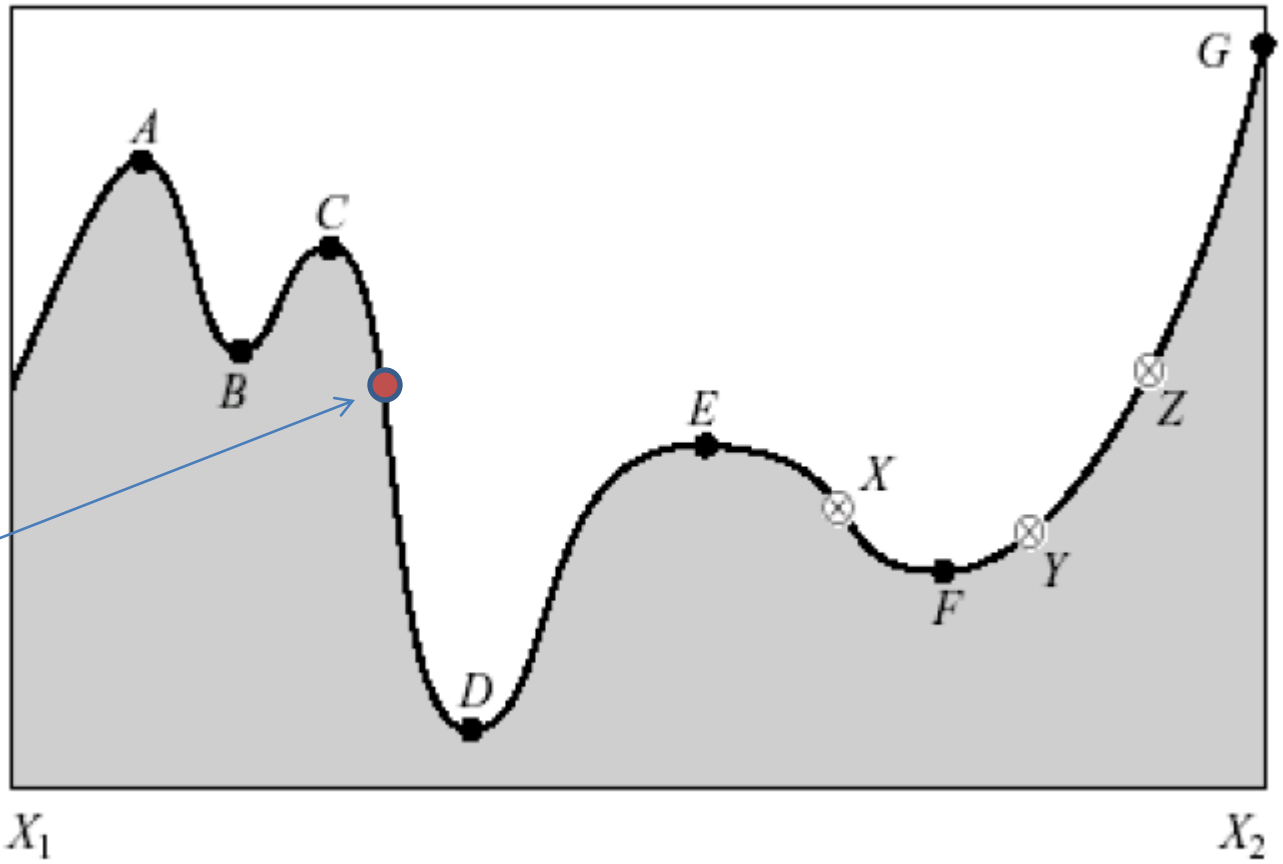


Rysunek 1. Przykład funkcji z oznaczonymi ekstremami [7]

D - minimum globalne

F – minimum lokalne

Problemy optymalizacji



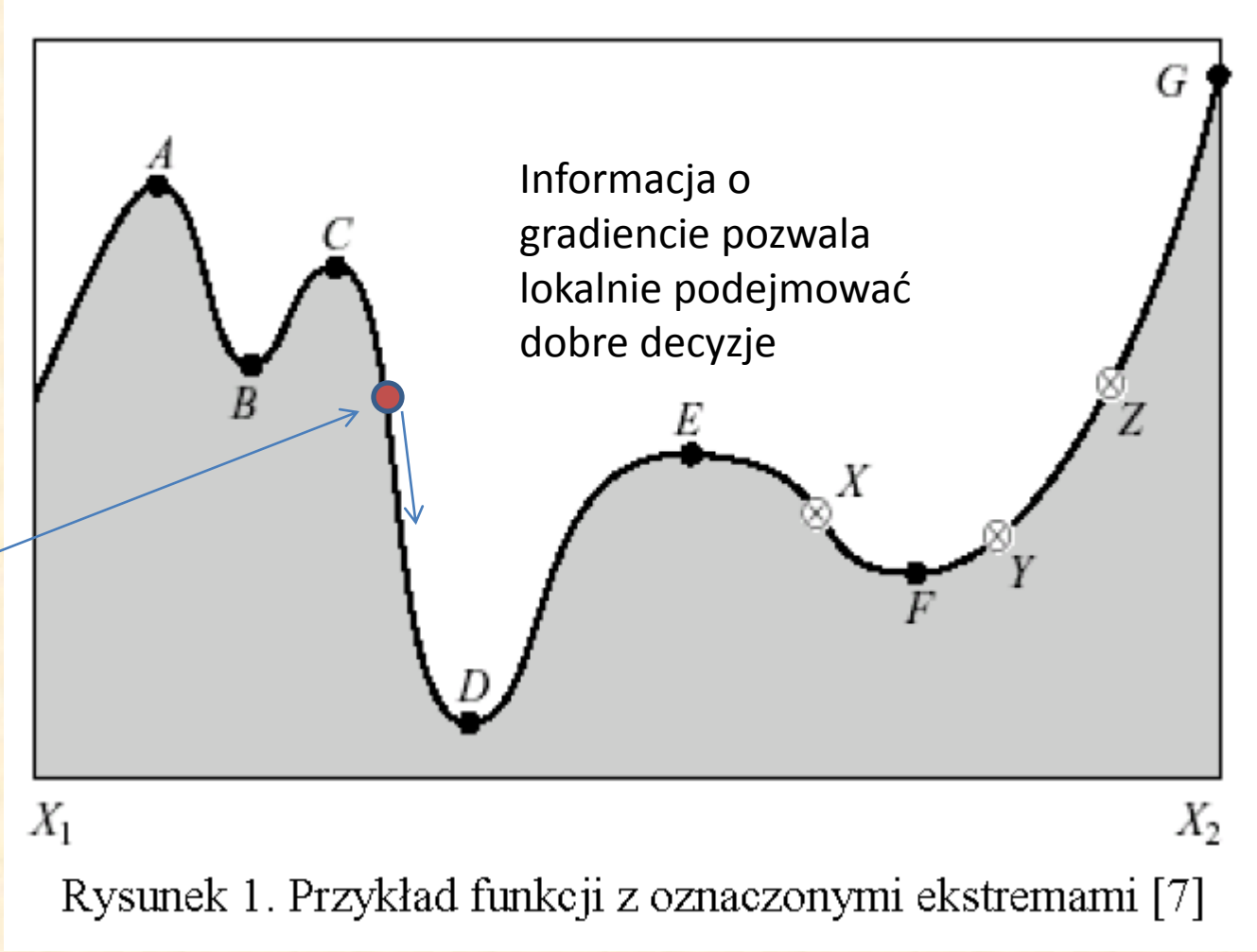
Rysunek 1. Przykład funkcji z oznaczonymi ekstremami [7]

Początkowe, losowe rozwiązanie

D - minimum globalne

F – minimum lokalne

Problemy optymalizacji

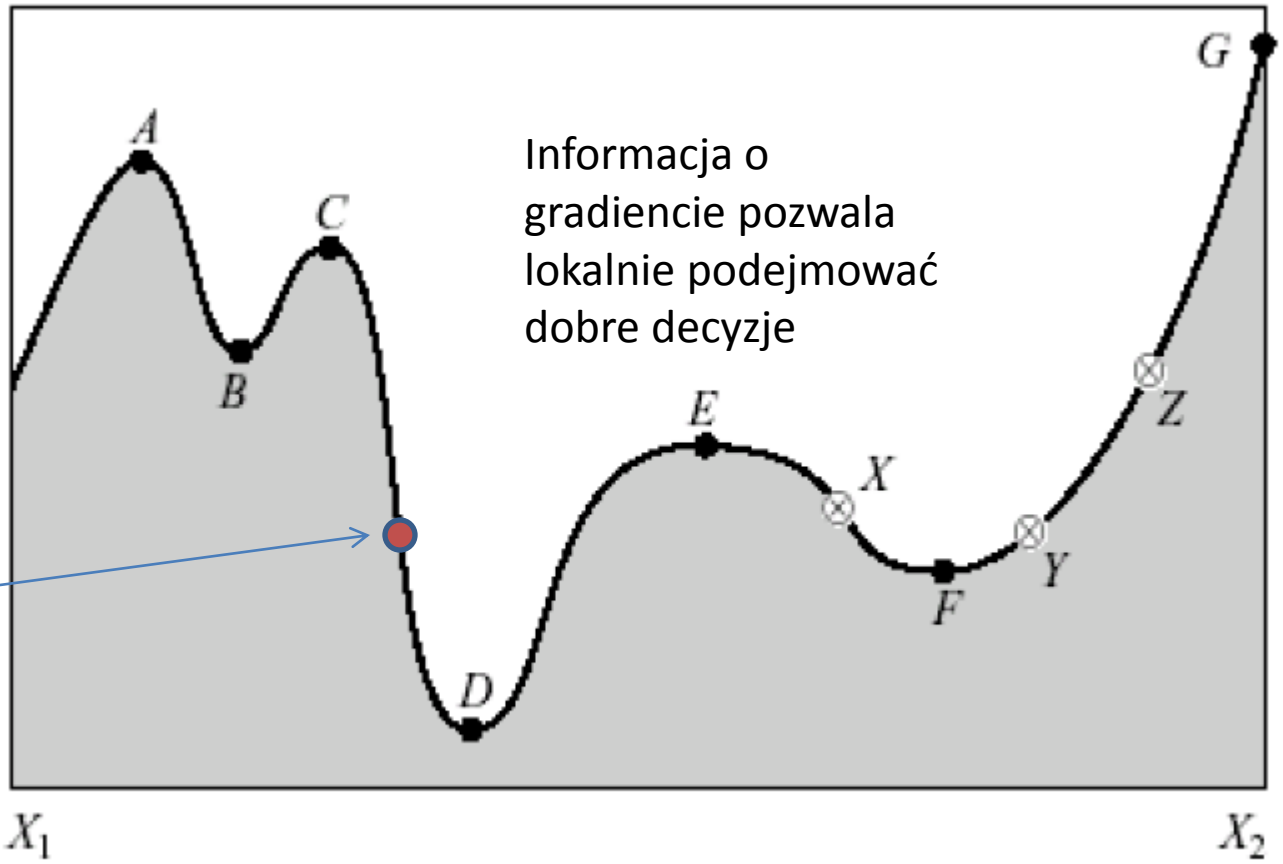


Rysunek 1. Przykład funkcji z oznaczonymi ekstremami [7]

D - minimum globalne

F – minimum lokalne

Problemy optymalizacji



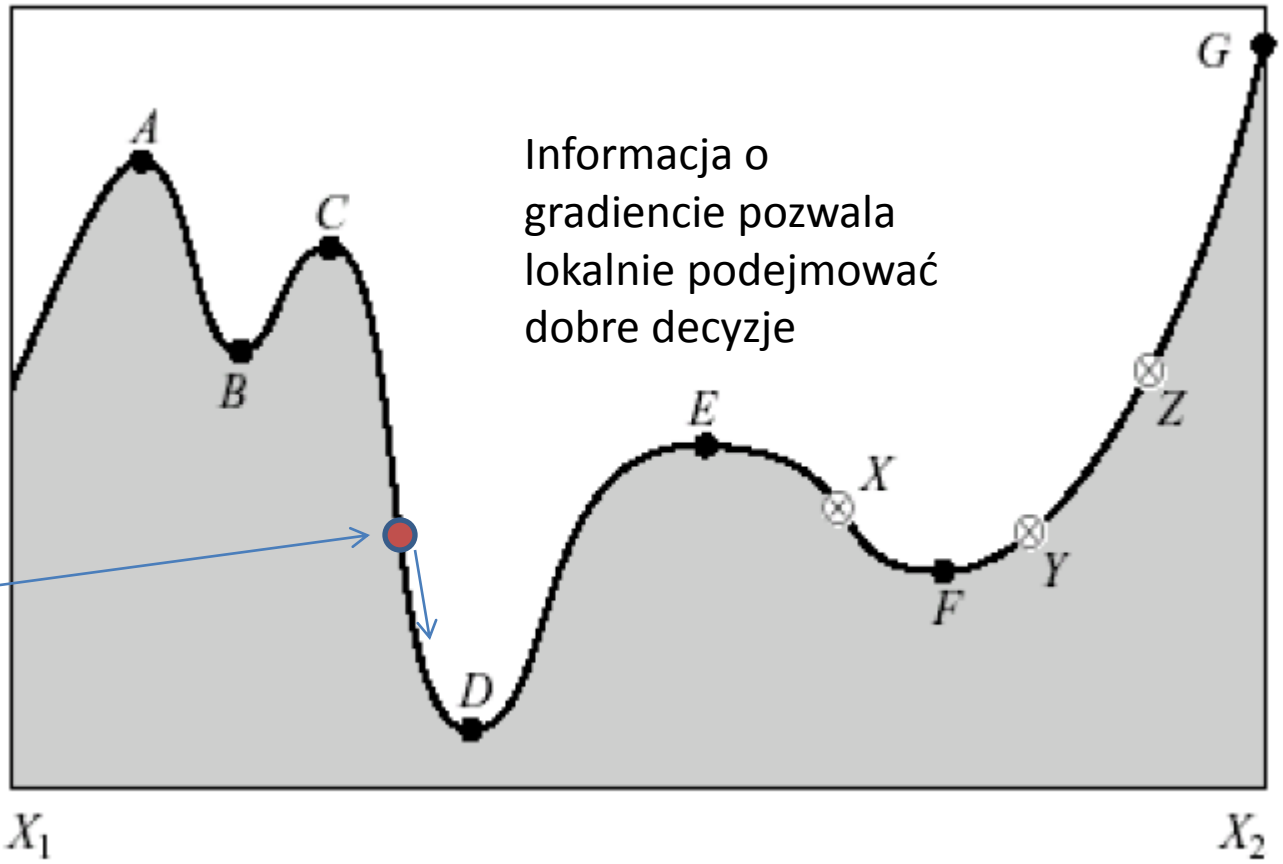
Rysunek 1. Przykład funkcji z oznaczonymi ekstremami [7]

Kolejne, ulepszone rozwiązanie

D - minimum globalne

F – minimum lokalne

Problemy optymalizacji



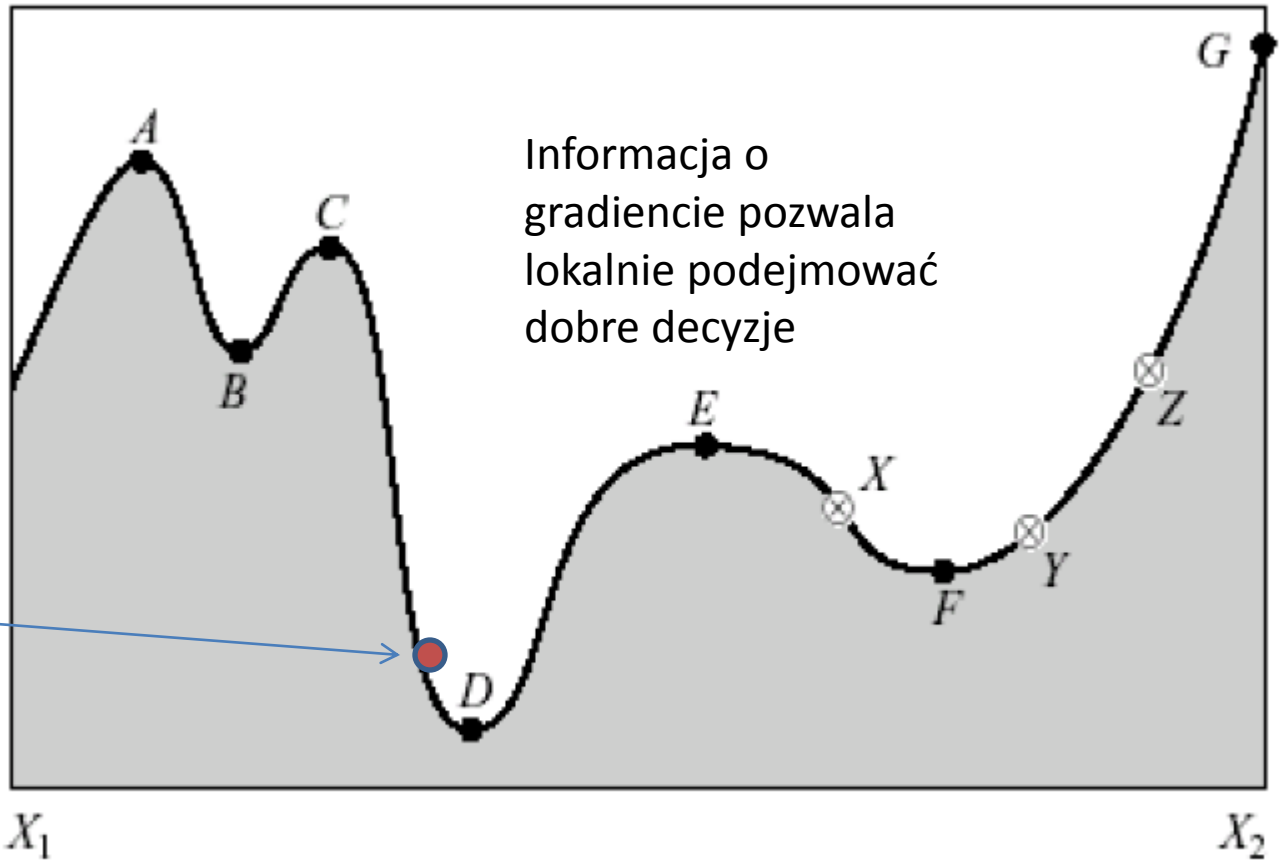
Rysunek 1. Przykład funkcji z oznaczonymi ekstremami [7]

Kolejne, ulepszone rozwiązanie

D - minimum globalne

F – minimum lokalne

Problemy optymalizacji



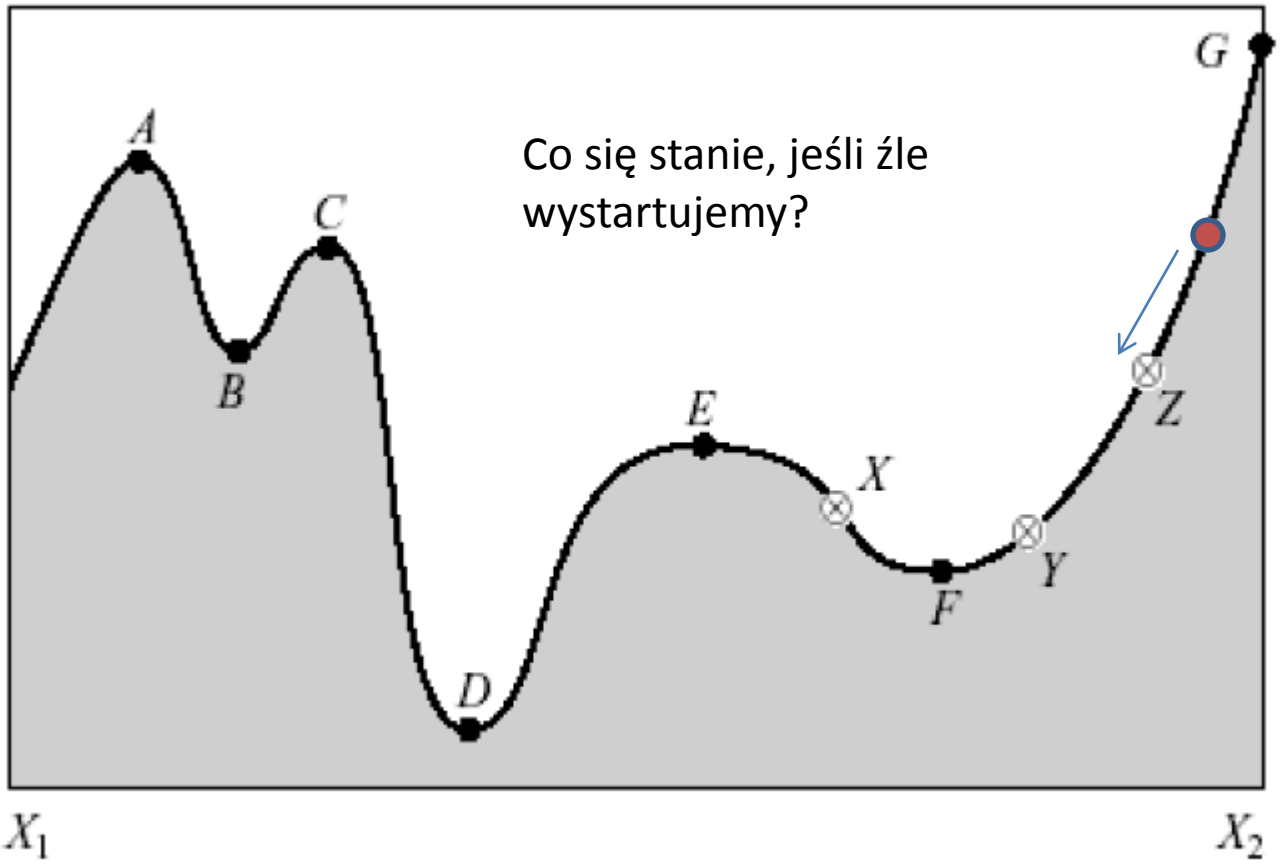
Rysunek 1. Przykład funkcji z oznaczonymi ekstremami [7]

Kolejne, ulepszone rozwiązanie

D - minimum globalne

F – minimum lokalne

Problemy optymalizacji

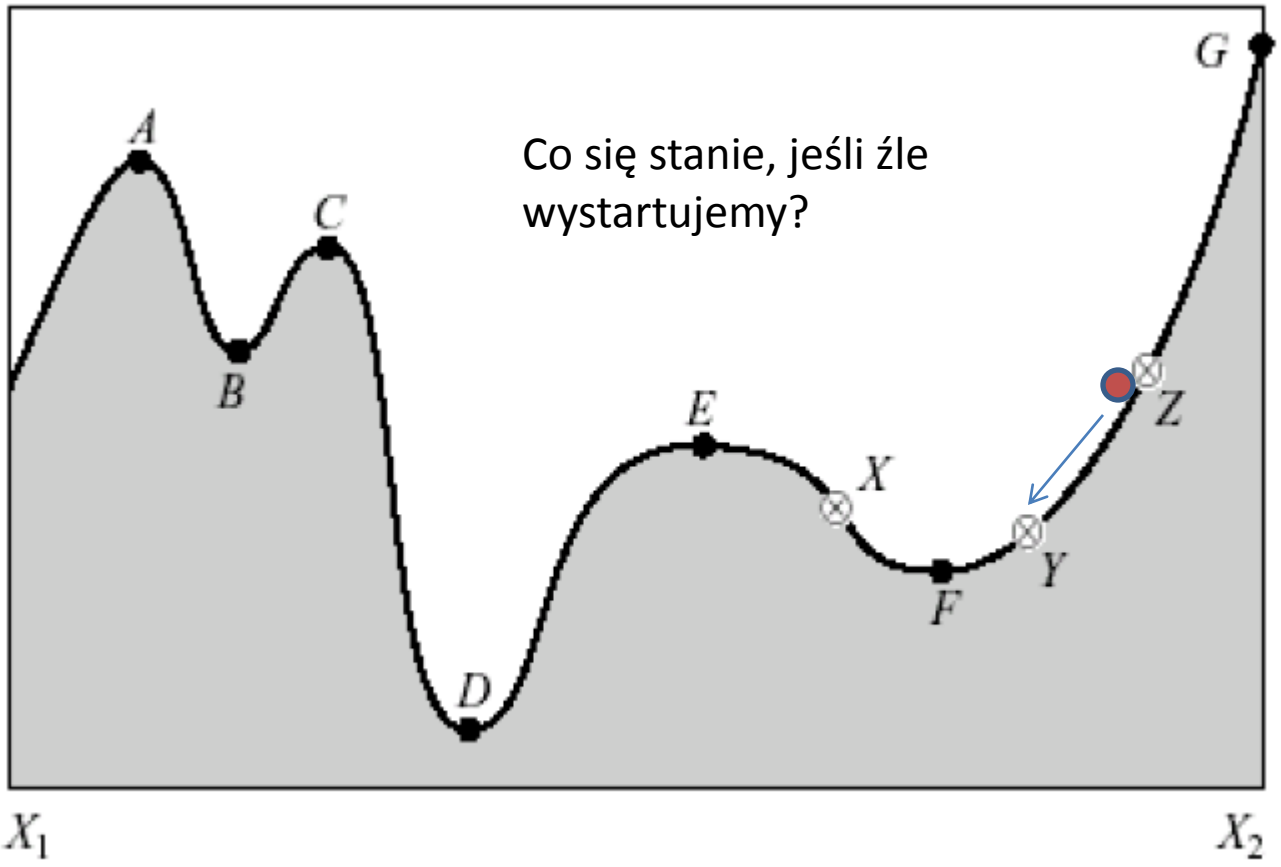


Rysunek 1. Przykład funkcji z oznaczonymi ekstremami [7]

D - minimum globalne

F – minimum lokalne

Problemy optymalizacji



Rysunek 1. Przykład funkcji z oznaczonymi ekstremami [7]

Algorytmy zachłanne mają skłonność do „utykania” w minimach lokalnych.

Algorytmy zachłanne

Algorytm zachłanny przetwarza **jedno** rozwiązanie, które iteracyjnie jest ulepszane – poszukiwania odbywają się jedynie w jego bezpośrednim otoczeniu (sąsiedztwie).

Wielokrotne uruchomienie algorytmu zachłannego lub równoległe uruchomienie wielu instancji tego algorytmu **nie jest równoważne przetwarzaniu populacyjnemu.**

Algorytmy zachłanne

Algorytmy zachłanne:

- **Symulowane wyżarzanie**
- **Algorytmy k-optymalne**

Definicja Algorytmu Ewolucyjnego

Algorytm ewolucyjny to algorytm przetwarzający populację rozwiązań poprzez wykorzystanie mechanizmów selekcji oraz operatorów różnicowania.

Definicja Algorytmu Ewolucyjnego

Wnioski:

- Rozwiązania w populacji oddziałują na siebie

Definicja Algorytmu Ewolucyjnego

Wnioski:

- Rozwiązania w populacji oddziałują na siebie
- Lepsze mają większe szanse na „przeżycie”

Definicja Algorytmu Ewolucyjnego

Wnioski:

- Rozwiązania w populacji oddziałują na siebie
- Lepsze mają większe szanse na „przeżycie”
- By dokonać selekcji istnieje potrzeba oceny rozwiązań – funkcja oceny, jakości, przystosowania

Definicja Algorytmu Ewolucyjnego

Wnioski:

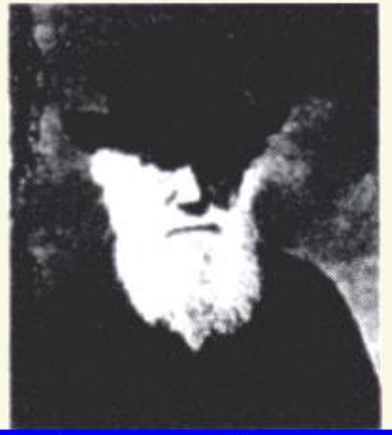
- Rozwiązania w populacji oddziałują na siebie
- Lepsze mają większe szanse na „przeżycie”
- By dokonać selekcji istnieje potrzeba oceny rozwiązań – funkcja oceny, jakości, przystosowania
- Kolejne rozwiązania (pokolenia), po zastosowaniu mechanizmów różnicowania, powinny zawierać część informacji rodziców.

Algorytmy Ewolucyjne

- Algorytmy Genetyczne
- Strategie ewolucyjne
- Programowanie genetyczne
- Komitety klasyfikatorów
- Algorytmy mrówkowe, mrowiskowe
- Algorytmu rojowe
- Sztuczne systemy immunologiczne
- Inne…

Syntetyczna teoria ewolucji

Gregor Mendel (1822-1884)



Charles Darvin (1809-1882)

GENETYKA

EWOLUCJA

Zasady dziedziczenia

Zasady doboru naturalnego

NEODARWINIZM

DNA

SYNTETYCZNA TEORIA EWOLUCJI

Historia

Pojęcie „**genu**” wprowadzono w **1910** roku na określenie abstrakcyjnej jednostki dziedziczenia, odpowiedzialnej za jakąś cechę danego gatunku.

W **1869r**. J. F. **Miescher** wyizolował z jąder komórkowych kwas deoksyrybonuleinowy - **DNA**.

W **1953** roku na podstawie dostępnych fizycznych i chemicznych informacji J. **Watson** i F. **Crick** ustalili, że **DNA** ma strukturę podwójnej **helisy**.

DNA jest **polimerem** składającym się z czterech cząsteczek zwanych nukleotydami: **adenina** (A), **tymina** (T), **guanina** (G), **cytozyna** (C).

Informacja genetyczna zapisywana jest w **DNA** za pomocą kolejności ułożenia cząstek.

Cechy algorytmów ewolucyjnych

- Przetwarzanie populacyjne
- Nie jest wymagana informacja o gradiencie
- Zdolność do radzenia sobie z pułapkami lokalnych optimów – zależy od konkretnego zadania i dobrego doboru parametrów algorytmu
- Podejście bardzo ogólne i elastyczne
- Mogą działać na dowolnej reprezentacji

Cechy algorytmów ewolucyjnych

- Łatwość implementacji
  - Stworzyć populację
  - Ocenić osobniki
  - Wprowadzić nacisk selekcyjny
  - Zastosować operatory różnicowania
  - Powtarzać „pętlę ewolucyjną”

Cechy algorytmów ewolucyjnych

- Pojęcie bardzo szerokie – inne algorytmy mogą być rozważane w kontekście AE (np. symulowane wyżarzanie – populacja z jednym osobnikiem, różnicowanie parametryzowane temperaturą)
- Ustalanie odpowiednich wartości parametrów algorytmu ewolucyjnego może być częścią samego algorytmu – np. strategie ewolucyjne

Cechy algorytmów ewolucyjnych

- Możliwość rozwiązywania problemów zmieniających się w czasie – są to algorytmy **adaptacyjne**
- Możliwość tworzenie modeli koewolucyjnych – osobniki nie muszą być oceniane za pomocą funkcji przystosowani, a jedynie są porównywane na zasadzie „lepszy - gorszy”
- Możliwość łączenia z innymi metodami

Cechy algorytmów ewolucyjnych

- Równoległość – możliwości wydajnych implementacji
- Możliwości rozszerzania o nowe koncepcje:
  - Wiek osobnika
  - Płeć
  - Pamięć (tzw. Algorytmy kulturowe)
  - Rodziny
  - Uczenie społeczne
  - Konkurencja między populacjami
  - Migracje osobników między nimi
  - Efekt Baldwina – uczenie się osobników w trakcie swego życia i wpływ tego uczenia na jakość rozwiązania

Cechy algorytmów ewolucyjnych

- Eksploracja vs. Eksploatacja

Trzy problemy

- Problem spełnialności formuł boolowskich – ***SAT***
- Problem komiwojażera - ***TSP***
- Programowanie nieliniowe - ***NLP***

Problem spełnialności formuł boolowskich – SAT

$$F(\mathbf{x}) = (x_{12} \vee \overline{x}_{22} \vee x_{31}) \wedge (x_{22} \vee \overline{x}_{42}) \wedge (x_{12} \vee x_{22} \vee x_{41} \vee \overline{x}_{77})\ldots$$

$$x_i \in \{0, 1\}$$

Dla jakich $x_1 \ldots x_n$ wyrażenie $F(x)$ jest spełnione?

Problem spełnialności formuł boolowskich – SAT

Rozwiązując problem (nie tylko za pomocą algorytmów ewolucyjnych) należy:

• dobrać sposób reprezentacji rozwiązań (kodowanie osobników)

Problem spełnialności formuł boolowskich – SAT

$$F(\mathbf{x}) = (x_{12} \vee \overline{x}_{22} \vee x_{31}) \wedge (x_{22} \vee \overline{x}_{42}) \wedge (x_{12} \vee x_{22} \vee x_{41} \vee \overline{x}_{77})\ldots$$

$$x_i \in \{0, 1\}$$

Naturalną formą reprezentacji rozwiązania w tym problemie jest ciąg bitów o długości ***n***.

(***n*** – liczba zmiennych).

Problem spełnialności formuł boolowskich – SAT

Rozwiązując problem (nie tylko za pomocą algorytmów ewolucyjnych) należy:

- dobrać sposób reprezentacji rozwiązań (kodowanie osobników)
- rozważyć przestrzeń możliwych rozwiązań

Problem spełnialności formuł boolowskich – SAT

$$F(\mathbf{x}) = (x_{12} \vee \overline{x}_{22} \vee x_{31}) \wedge (x_{22} \vee \overline{x}_{42}) \wedge (x_{12} \vee x_{22} \vee x_{41} \vee \overline{x}_{77})\ldots$$

$$x_i \in \{0, 1\}$$

Liczba potencjalnych rozwiązań: 2^n
gdzie n – liczba zmiennych

$2^{100} \approx 10^{30}$

Sprawdzając 1000 ciągów na sekundę, od Wielkiego Wybuchu (15 miliardów lat) sprawdzilibyśmy jedynie **1%** możliwych rozwiązań!

Problem spełnialności formuł boolowskich – SAT

Rozwiązując problem (nie tylko za pomocą algorytmów ewolucyjnych) należy:

- dobrać sposób reprezentacji rozwiązań (kodowanie osobników)
- rozważyć przestrzeń możliwych rozwiązań
- dobrać funkcję oceny jaką należy użyć aby oceniać rozwiązania

Problem spełnialności formuł boolowskich – SAT

$$F(\mathbf{x}) = (x_{12} \vee \overline{x}_{22} \vee x_{31}) \wedge (x_{22} \vee \overline{x}_{42}) \wedge (x_{12} \vee x_{22} \vee x_{41} \vee \overline{x}_{77})\ldots$$

$$x_i \in \{0, 1\}$$

Jak oceniać rozwiązania ?

Funkcja oceny powinna dawać wskazówki odnośnie jakości rozwiązania – rozwiązania bliższe optymalnego powinny mieć lepszą funkcję oceny.

Tutaj funkcja ***F*** daje jedynie odpowiedź ***TRUE*** lub ***FALSE***.

Dla rozwiązania problemu otrzymamy odpowiedź ***TRUE***, jednak dla wszystkich innych otrzymamy ***FALSE*** – nawet jeśli tylko jeden bit dzieli nas od rozwiązania problemu!

Problem spełnialności formuł boolowskich – SAT

$$F(\mathbf{x}) = (x_{12} \vee \overline{x}_{22} \vee x_{31}) \wedge (x_{22} \vee \overline{x}_{42}) \wedge (x_{12} \vee x_{22} \vee x_{41} \vee \overline{x}_{77})\ldots$$

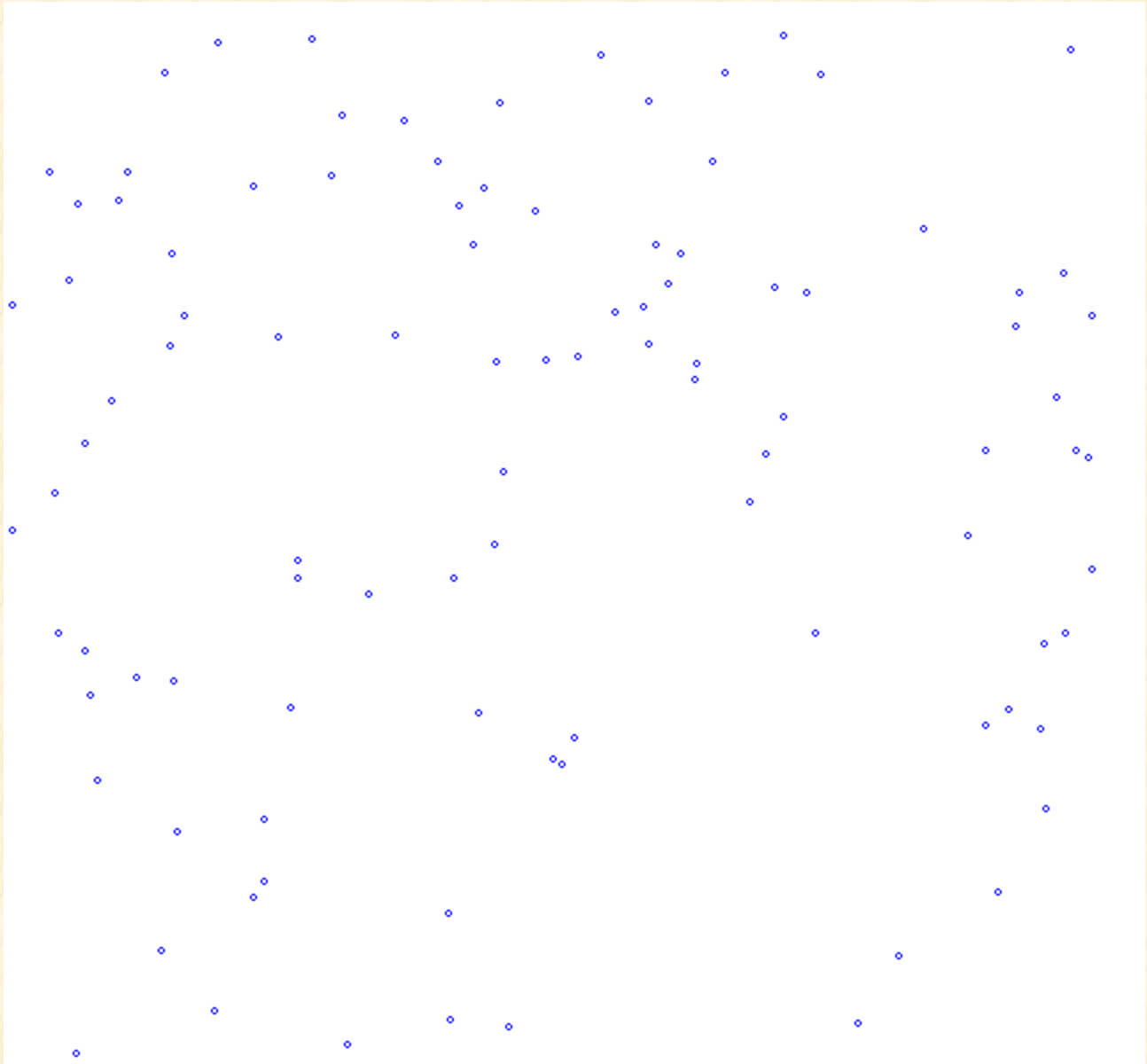
$$x_i \in \{0, 1\}$$

Jak ocenić, które rozwiązania są lepsze?

Problem komiwojażera - TSP

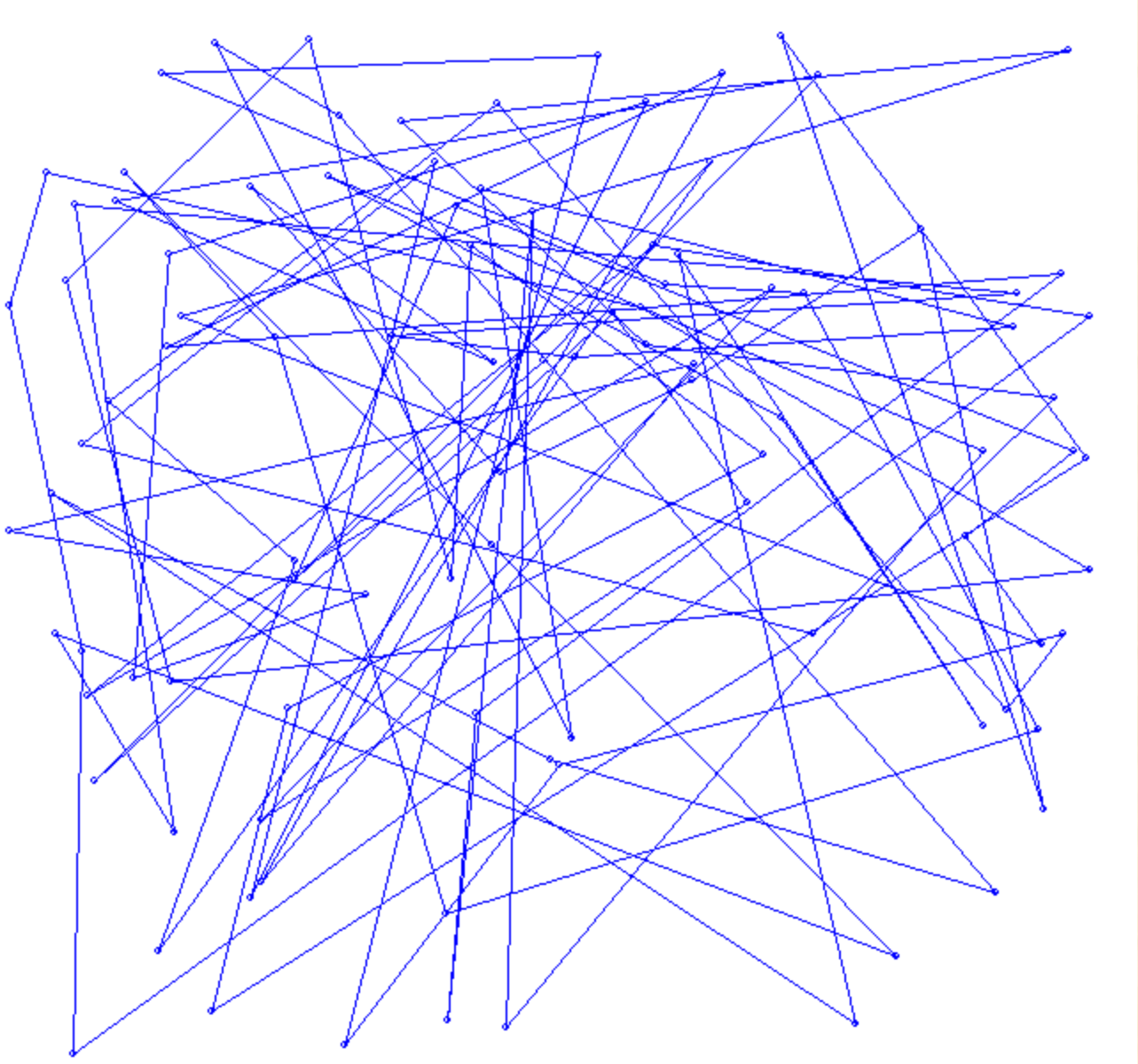
- Komiwojażer ma za zadanie odwiedzić wszystkie ***n*** miast, każde dokładnie jeden raz, i powrócić do miasta początkowego.
- W języku teorii grafów jest to poszukiwanie cyklu Hammiltona w grafie.
- Problem ***TSP***
  - Symetryczny
  - Niesymetryczny

Problem komiwojażera - TSP



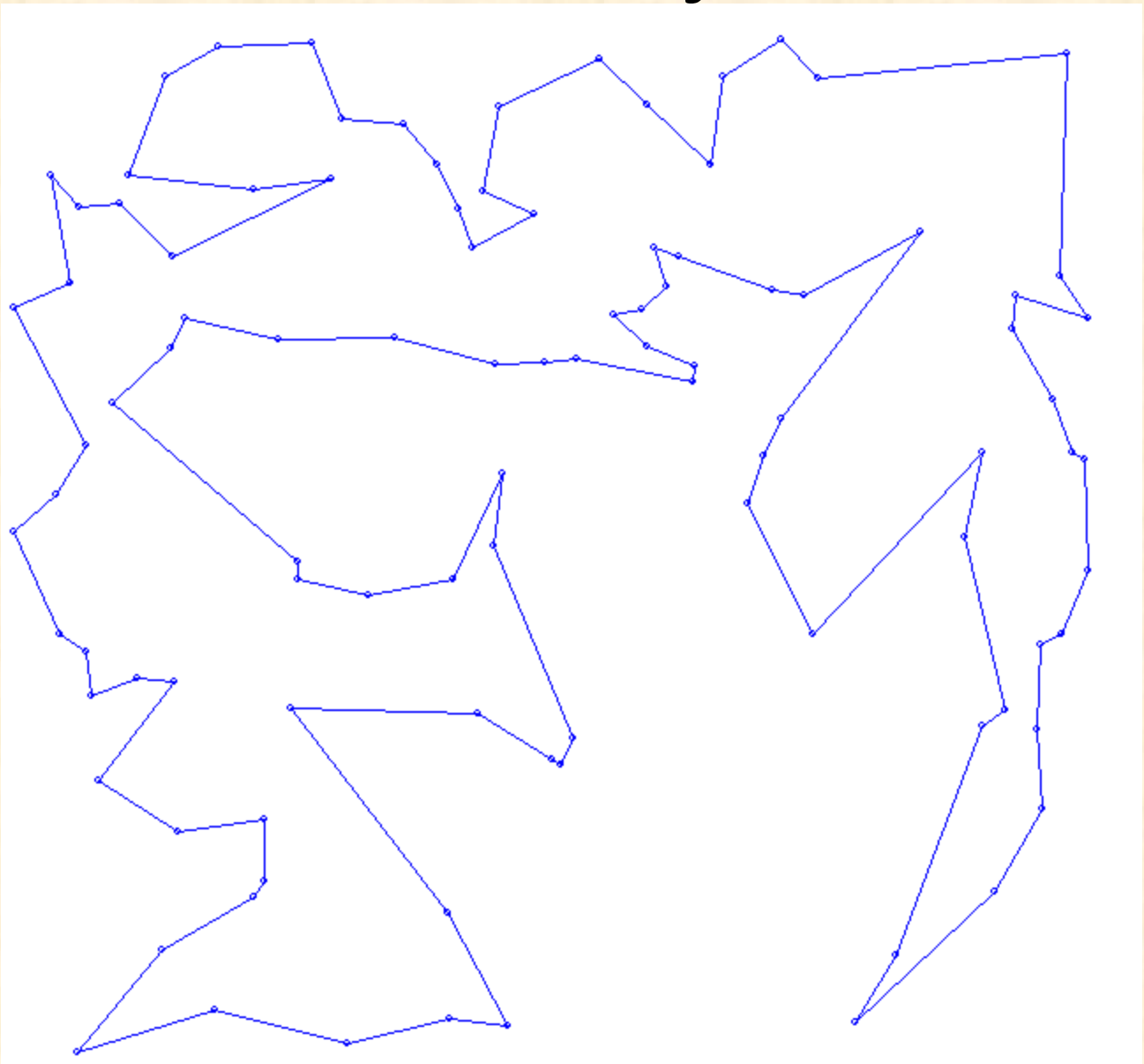
N=100

Problem komiwojażera - TSP



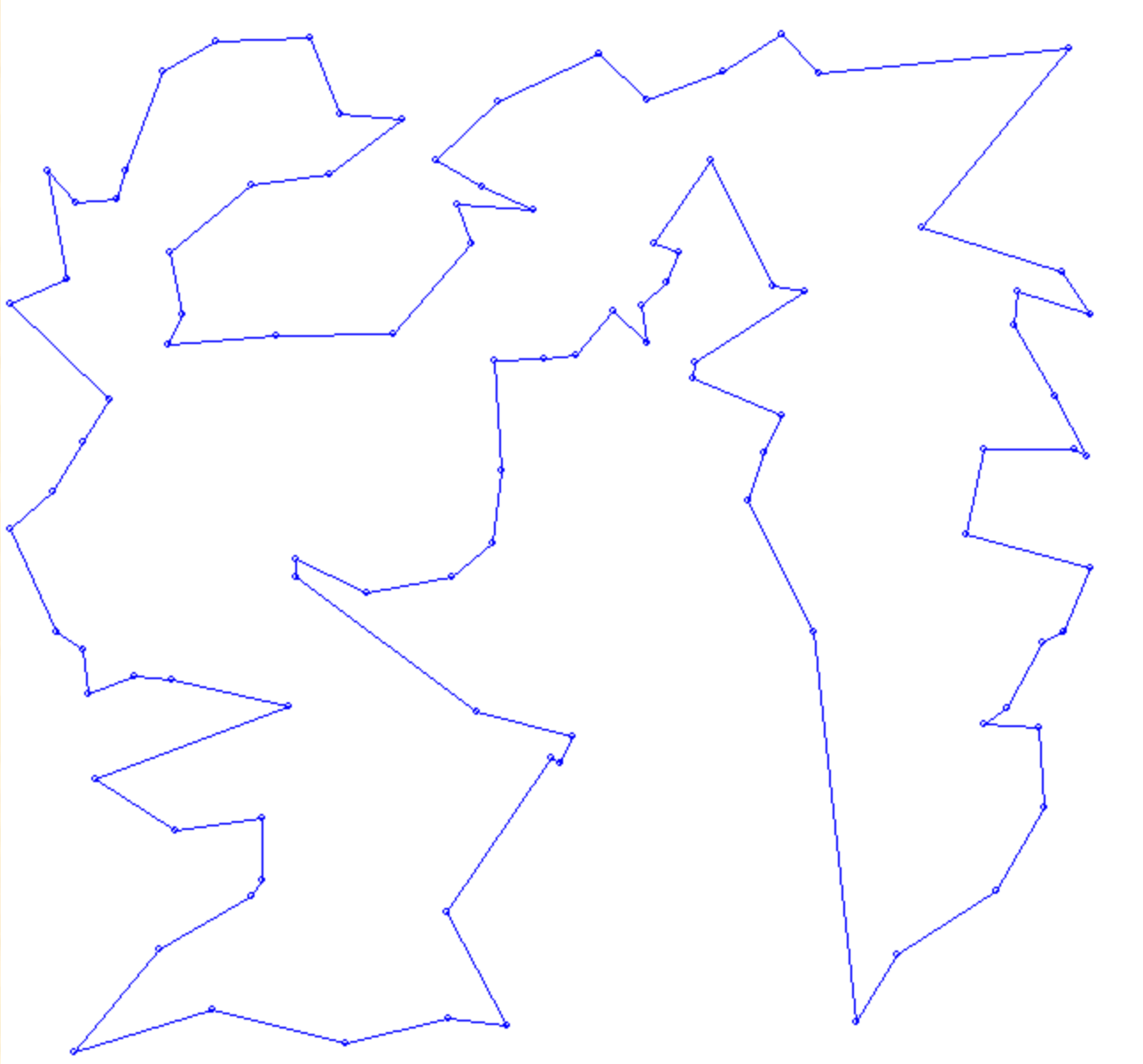
N=100

Problem komiwojażera - TSP



N=100

Problem komiwojażera - TSP



N=100

Problem komiwojażera - TSP

Forma reprezentacji:

Naturalną formą reprezentacji jest lista kolejno odwiedzanych miast:

12 – 32 – 3 – 44 …… - 100 – 55

Niekoniecznie jest to reprezentacja najlepsza!

Problem komiwojażera - TSP

Forma reprezentacji:

Załóżmy problem o 6-ciu miastach, dwa osobniki oraz krzyżowanie jednopunktowe, jak w klasycznym algorytmie genetycznym:

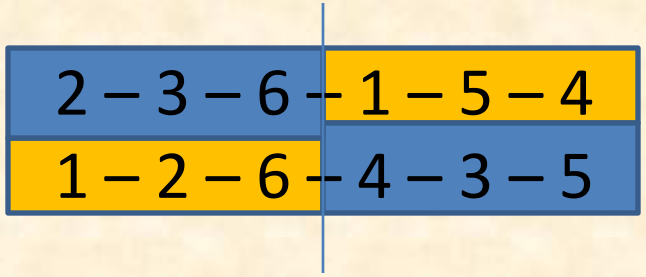
Rodzic 1: 2 – 3 – 6 – 1 – 5 – 4

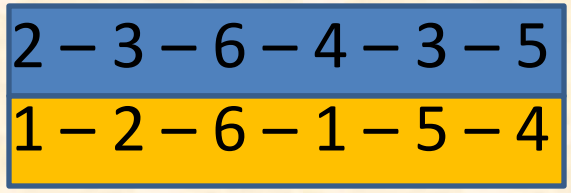
Rodzic 2: 1 – 2 – 6 – 4 – 3 – 5

Problem komiwojażera - TSP

Forma reprezentacji:

Załóżmy problem o 6-ciu miastach, dwa osobniki oraz krzyżowanie jednopunktowe, jak w klasycznym algorytmie genetycznym:

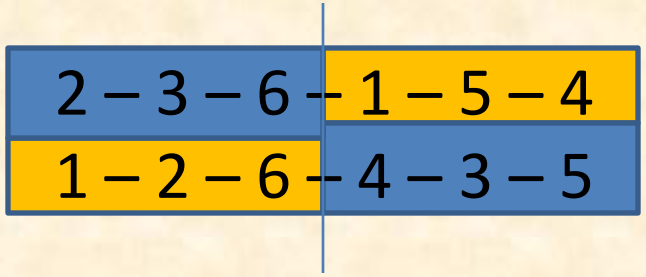


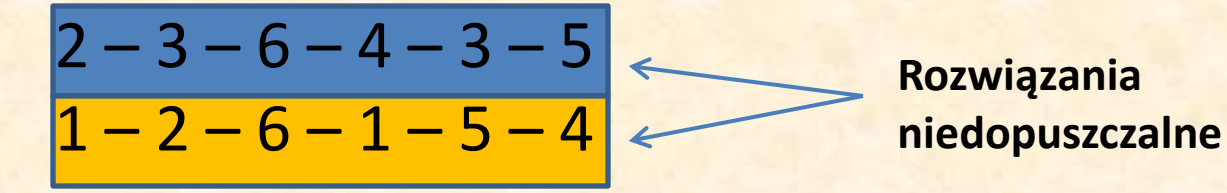


Problem komiwojażera - TSP

Forma reprezentacji:

Załóżmy problem o 6-ciu miastach, dwa osobniki oraz krzyżowanie jednopunktowe, jak w klasycznym algorytmie genetycznym:





Takiego problemu nie było przy problemie SAT

Problem komiwojażera - TSP

Możliwe rozwiązania:

- Naprawiać potomków (może być kosztowne)
- Stosować inną formę reprezentacji
- Stosować inne operatory krzyżowania (różnicowania)

Problem komiwojażera - TSP

Przestrzeń poszukiwań

Każda permutacja ***n*** liczb jest rozwiązaniem, ale różnych rozwiązań jest (dla symetrycznego ***TSP***):

$$|S| = n! / (2n) = (n-1)! / 2$$

dla n=10 $|S|$ = 181 000

dla n=20 $|S|$ = 10 000 000 000 000 000

Problem komiwojażera - TSP

Funkcja oceny:

W tym przypadku nie ma problemu z funkcją oceny – jest nią długość trasy.

Programowanie nieliniowe - NLP

Szukanie optimum funkcji z ograniczeniami

Przykład: znajdź maksimum funkcji:

$$G2(\mathbf{x}) = \left| \frac{\sum_{i=1}^{n} cos^4(x_i) - 2 * \prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i * x_i^2}} \right|$$

Przy ograniczeniach:

$$\prod_{i=1}^{n} x_i \geq 0.75$$

$$\sum_{i=1}^{n} x_i \leq 7.5 * n$$

dla $0 \leq x_i \leq 10, \; dla \; 1 \leq i \leq n$

Programowanie nieliniowe - NLP

Funkcja ta jest nieliniowa a jej globalne minimum nie jest znane – wiadomo, że leży w pobliżu środka układu współrzędnych.

Potencjalnych rozwiązań jest nieskończenie wiele – w praktyce ograniczone są dokładnością komputerów.

Co z rozwiązaniami niedopuszczalnymi?

Programowanie nieliniowe - NLP

Forma reprezentacji: binarna (klasyczny AG lub kodowanie rzeczywiste)

Programowanie nieliniowe - NLP

Forma reprezentacji - binarna (klasyczny AG lub kodowanie rzeczywiste).

Przestrzeń poszukiwań – ogromna i ograniczona.

Programowanie nieliniowe - NLP

Forma reprezentacji - binarna (klasyczny AG lub kodowanie rzeczywiste).

Przestrzeń poszukiwań – ogromna i ograniczona.

Funkcja oceny – można wykorzystać funkcję ***G2*** bezpośrednio, ale co z rozwiązaniami niedopuszczalnymi? **Jeśli będziemy je oceniać jako najgorsze, możemy stracić rozwiązania bliskie optymalnym, gdyż optymalne rozwiązanie leży gdzieś blisko obszaru niedopuszczalnego!**

Programowanie nieliniowe - NLP

Jeśli będziemy je oceniać jako najgorsze, możemy stracić rozwiązania bliskie optymalnym, gdyż optymalne rozwiązanie leży gdzieś blisko obszaru niedopuszczalnego!

Potencjalne rozwiązania:

- specjalizowane operatory różnicowania
- modyfikacje funkcji oceny – np. uwzględnienie stopnia naruszenia ograniczeń

Algorytmy Ewolucyjne

Problemy w trakcie projektowania algorytmów ewolucyjnych:

- Forma reprezentacji
- Duża przestrzeń rozwiązań
- Dobór funkcji oceny
- Rodzaj selekcji
- Odpowiednie operatory różnicowania

Algorytmy Ewolucyjne

Poszczególne składowe nie są niezależne od siebie, lecz mocno na siebie wpływają!

Forma reprezentacji wpływa na wielkość przestrzeni rozwiązań oraz na możliwe sposoby realizacji operatorów krzyżowania i mutacji.

Odpowiedni dobór funkcji oceny wpływa na skuteczność selekcji.

Funkcja oceny nie jest wyznaczona jednoznacznie!

Optymalność danego rozwiązania może być funkcją czasu! (losowość, konkurencja)

Algorytmy Ewolucyjne

Należy pamiętać, że powstały algorytm jest oceniany jako całość. Możliwości oceny wpływu np. samej selekcji czy też samego krzyżowania na jakość działania algorytmu są ograniczone.

Teoretyczne podstawy działania algorytmów genetycznych

Twierdzenie o schematach daje pewien teoretyczny wgląd na działanie algorytmu genetycznego i pozwala lepiej zrozumieć jego działanie.

Zakładamy:

- Kodowanie binarne
- Selekcje za pomocą koła ruletki
- Krzyżowanie jednopunktowe
- Mutację jednopunktową

Teoretyczne podstawy działania algorytmów genetycznych

Pojęcie schematu zostało wprowadzone w celu określenia zbioru chromosomów o pewnych wspólnych cechach, podobieństwach.

Zakładamy, że allele należą do zbioru {0, 1}.

Wtedy schemat jest zbiorem chromosomów zawierających zera i jedynki na wyszczególnionych pozycjach.

Teoretyczne podstawy działania algorytmów genetycznych

Schematy rozważamy z wykorzystaniem rozszerzonego alfabetu :

{0, 1, *}

* - oznacza dowolną wartość (0 lub 1)

Teoretyczne podstawy działania algorytmów genetycznych

Przykłady:

10*1 = {1001, 1011}

*01*10 = {001010, 001110, 101010, 101110}

Chromosom należy do danego schematu, jeżeli dla każdej pozycji (*locus*) $j=1, 2,\ldots, L$, gdzie L jest długością chromosomu, symbol występujący na j-tej pozycji chromosomu odpowiada symbolowi na j-tej pozycji schematu, przy czym zarówno 0 jaki i 1 odpowiadają symbolowi *.

Teoretyczne podstawy działania algorytmów genetycznych

Jeżeli w schemacie występuje *m* symboli *, to schemat ten zawiera 2^m chromosomów.

Przykład:

01 pasuje do schematów: **, *1, 0*, 01

Każdy chromosom od długości *L* należy do 2^L schematów.

Teoretyczne podstawy działania algorytmów genetycznych

Algorytm genetyczny opiera się na przetwarzaniu osobników najlepiej przystosowanych.

P(0) – populacja początkowa

P(t) – populacja w generacji ***t***

M(t) – pula rodzicielska (ang. *Mating pool*) wybrana z ***P(t)*** w generacji ***t***

p_k – prawdopodobieństwo krzyżowania

p_m – prawdopodobieństwo mutacji

Teoretyczne podstawy działania algorytmów genetycznych

Pożądane jest, by liczba osobników pasujących do schematu reprezentującego dobre rozwiązanie w populacji ***P(t)*** wzrastała wraz ze wzrostem generacji.

Mają na to wpływ trzy czynniki:

- Selekcja
- Krzyżowanie
- Mutacja

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ selekcji

S – dany schemat

c(S, t) – liczba osobników w populacji ***P(t)*** pasujących do schematu ***S***, zatem ***c(S, t)*** jest liczbą elementów zbioru

$$P(t) \cap S$$

Podczas selekcji chromosom ch_i zostaje wybrany do ***M(t)*** z prawdopodobieństwem

$$p_s(ch_i) = \frac{F(ch_i)}{\sum_{j=1}^{K} F(ch_j)}$$

gdzie ***K*** to rozmiar populacji.

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ selekcji

F(S, t) - średnia wartość funkcji przystosowania dla chromosomów w populacji ***P(t)*** pasujących do schematu ***S***. Jeżeli

$$P(t) \cap S = \{ch_i, \ldots, ch_{c(S,t)}\}$$

to

$$F(S,t) = \frac{\sum_{i=1}^{c(S,t)} F(ch_i)}{c(S,t)}$$

F(S, t) to inaczej przystosowanie schematu ***S*** w generacji ***t***.

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ selekcji

$\tilde{F}(t)$ Suma wartości funkcji przystosowania chromosomów w populacji ***P(t)*** o liczebności ***K***

$$\tilde{F}(t) = \sum_{i=1}^{K} F(ch_i^{(t)})$$

$\overline{F}(t)$ Średnia wartość funkcji przystosowania chromosomów w populacji ***P(t)***

$$\overline{F}(t) = \frac{1}{K}\tilde{F}$$

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ selekcji

Niech ***chr(t)*** będzie elementem puli rodzicielskiej ***M(t)***.

Dla każdego ***chr(t)*** i dla każdego ***i=1, 2, … , c(S, t)*** prawdopodobieństwo, że ***chr(t)=ch$_i$*** jest dane wzorem

$$F(ch_i) / \tilde{F}(t)$$

Zatem oczekiwana liczba chromosomów w populacji ***M(t)*** równych ***ch$_i$*** wynosi

$$K \frac{F(ch_i)}{\tilde{F}(t)} = \frac{F(ch_i)}{\overline{F}(t)}$$

Oczekiwana liczba chromosomów w zbiorze ***P(t)*** i reprezentujących ***S*** wybranych do puli rodzicielskiej ***M(t)*** jest równa

$$\sum_{i=1}^{c(S,t)} \frac{F(ch_i)}{\overline{F}(t)} = c(S,t) \frac{F(S,t)}{\overline{F}(t)}$$

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ selekcji

Niech ***chr(t)*** będzie elementem puli rodzicielskiej ***M(t)***.

Dla każdego ***chr(t)*** i dla każdego ***i=1, 2, … , c(S, t)*** prawdopodobieństwo, że ***chr(t)=ch$_i$*** jest dane wzorem

$$F(ch_i) / \tilde{F}(t)$$

Zatem oczekiwana liczba chromosomów w populacji ***M(t)*** równych ***ch$_i$*** wynosi

$$\sum_{i=1}^{c(S,t)} \frac{F(ch_i)}{\overline{F}(t)} = c(S,t) \frac{F(S,t)}{\overline{F}(t)}$$

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ selekcji

Jeśli przez ***b(S, t)*** oznaczymy liczbę chromosomów z puli ***M(t)*** pasujących do schematu ***S***, to można wyciągnąć następujący wniosek:

Wniosek 1 (wpływ selekcji):

Wartość oczekiwana ***b(S, t)***, czyli oczekiwana liczba chromosomów w puli rodzicielskiej ***M(t)*** pasujących do schematu ***S*** jest określona wzorem

$$E[b(S,t)] = c(S,t)\frac{F(S,t)}{\overline{F}(t)}$$

Stąd wynika, że jeżeli schemat ***S*** zawiera chromosomy o wartości funkcji przystosowania powyżej średniej, to oczekiwana liczba chromosomów pasujących do schematu ***S*** w puli rodzicielskiej ***M(t)*** jest większa niż liczba chromosomów pasujących do schematu ***S*** w populacji ***P(t)***

Zatem selekcja powoduje rozprzestrzenianie się schematów o przystosowaniu lepszym niż przeciętne a zanikanie schematów o gorszym przystosowaniu.

Teoretyczne podstawy działania algorytmów genetycznych

Rząd (ang. order) schematu ***S*** (liczność schematu ***S***) oznaczany jako ***o(S)***, jest to liczba ustalonych pozycji w schemacie, tzn. zer i jedynek w przypadku alfabetu {0, 1, *}

Przykład:

o(10*1) = 3

o(*01*10) = 4

o(**0*1*) = 2

o(*101**) = 3

Rząd schematu równa się długości ***L*** minus liczba symboli *.

Teoretyczne podstawy działania algorytmów genetycznych

Rozpiętość (ang. *defining length*) schematu ***S*** nazywana także długością, oznaczana jako ***d(S)***, jest to odległość między pierwszym i ostatnim ustalonym symbolem (jest to różnica między prawą i lewą skrajną pozycją o ustalonym symbolu).

Przykład:

d(10*1) = 3

d(*01*10) = 4

d(**0*1*) = 2

d(*101**) = 2

Rozpiętość to liczba z przedziału [0, ***L***-1]

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ krzyżowania

Niektóre schematy są bardziej wrażliwe na zniszczenie podczas krzyżowania niż inne. (Rozpatrujemy krzyżowanie jednopunktowe.)

Przykład

S1 = 1****0* - łatwo zniszczyć

S2 = **01*** - trudno zniszczyć

Istotna jest tutaj rozpiętość schematów.

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ krzyżowania

Rozważmy chromosom z puli ***M(t)*** należący do schematu ***S***.

Prawdopodobieństwo, że chromosom ten zostanie wybrany do krzyżowania wynosi p_k.

Jeżeli żaden z jego potomków nie będzie należał do schematu ***S*** to oznacza, że punkt krzyżowania musi znajdować się między pierwszym i ostatnim ustalonym symbolem w schemacie ***S***.

Prawdopodobieństwo tego równa się

$$d(S)/(L-1)$$

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ krzyżowania

Wniosek (wpływ krzyżowania)

Dla danego chromosomu w ***M(t)*** reprezentującego schemat ***S***, prawdopodobieństwo, że chromosom ten zostanie wybrany do krzyżowania i żaden z jego potomków nie będzie należał do schematu ***S***, jest ograniczone z góry przez

$$p_k \frac{d(S)}{L-1}$$

Jest to prawdopodobieństwo zniszczenia schematu *S*.

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ krzyżowania

<u>Wniosek (wpływ krzyżowania cd.)</u>

Dla danego chromosomu w ***M(t)*** reprezentującego schemat ***S***, prawdopodobieństwo, że chromosom ten albo nie zostanie wybrany do krzyżowania albo co najmniej jeden z jego potomków będzie należał do schematu ***S***, jest ograniczone z dołu przez

$$1 - p_k \frac{d(S)}{L-1}$$

Jest to prawdopodobieństwo przetrwania schematu *S*.

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ krzyżowania

Uwaga: Jeśli oba chromosomy rodzicielskie należą do schematu ***S***, wtedy obydwa chromosomy będące ich potomkami także należą do schematu ***S***.

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ mutacji

Operator mutacji zmienia losowo z prawdopodobieństwem p_m wartość na określonej pozycji z 0 na 1 i odwrotnie.

Jeśli schemat ma przetrwać mutację, to wszystkie ustalone pozycje w schemacie muszą pozostać niezmienione.

Dany chromosom z ***M(t)*** reprezentujący ***S***, pozostaje w schemacie ***S*** wtedy i tylko wtedy, gdy żaden z symboli w tym chromosomie, odpowiadających ustalonym symbolom w schemacie ***S***, nie ulega zmianie podczas mutacji.

Prawdopodobieństwo takiego zdarzenia wynosi

$$(1-p_m)^{o(S)}$$

Teoretyczne podstawy działania algorytmów genetycznych

Wpływ mutacji

Wniosek (wpływ mutacji)

Dla danego chromosomu w ***M(t)*** reprezentującego ***S***, prawdopodobieństwo, że chromosom ten będzie należał do schematu ***S*** po operacji mutacji, jest dane przez

$$(1 - p_m)^{o(S)}$$

Jest to prawdopodobieństwo przetrwania mutacji przez schemat *S*.

Teoretyczne podstawy działania algorytmów genetycznych

Efektem połączenia wpływu selekcji, krzyżowania i mutacji jest następujący schemat reprodukcji:

$$E[c(S,t+1)] \geq c(S,t)\frac{F(S,t)}{\overline{F}(t)}(1-p_k\frac{d(S)}{L-1})(1-p_m)^{o(S)}$$

Zależność ta pokazuje, jak liczba chromosomów pasujących do schematu ***S*** zmienia się z populacji na populację.

Im większa jest wartość każdego z czynników odpowiadających za selekcję, krzyżowanie i mutację, tym większa jest oczekiwana liczba dopasowań do schematu ***S*** w następnej populacji.

Teoretyczne podstawy działania algorytmów genetycznych

Jeśli prawdopodobieństwo mutacji jest bardzo małe (**$p_m << 1$**), to można przyjąć, że prawdopodobieństwo przetrwania mutacji przybliżamy jako:

$$(1-p_m)^{o(S)} = 1 - p_m o(S)$$

Otrzymujemy

$$E[c(S,t+1)] \geq c(S,t)\frac{F(S,t)}{\overline{F}(t)}(1-p_k\frac{d(S)}{L-1}-p_m o(s))$$

Teoretyczne podstawy działania algorytmów genetycznych

Dla dużych populacji można aproksymować

$$E[c(S,t+1)] \geq c(S,t)\frac{F(S,t)}{\overline{F}(t)}(1-p_k\frac{d(S)}{L-1}-p_m o(s))$$

jako

$$c(S,t+1) \geq c(S,t)\frac{F(S,t)}{\overline{F}(t)}(1-p_k\frac{d(S)}{L-1}-p_m o(s))$$

Teoretyczne podstawy działania algorytmów genetycznych

$$c(S,t+1) \geq c(S,t)\frac{F(S,t)}{\overline{F}(t)}(1-p_k\frac{d(S)}{L-1}-p_m o(s))$$

Wynika stąd, że oczekiwana liczba chromosomów pasujących do schematu ***S*** w następnej generacji jest funkcją aktualnej liczby chromosomów należących do tego schematu, względnego przystosowania schematu oraz rzędu i rozpiętości schematu.

Schematy o przystosowaniu powyżej średniej oraz małego rzędu i o małej rozpiętości charakteryzują się rosnącą liczbą swoich reprezentantów w kolejnych populacjach.

Wzrost ten jest wykładniczy.

Teoretyczne podstawy działania algorytmów genetycznych

$$c(S,t+1) \geq c(S,t)\frac{F(S,t)}{\overline{F}(t)}(1-p_k\frac{d(S)}{L-1}-p_m o(s))$$

Wzrost ten jest wykładniczy.
Dla dużych populacji stosujemy przybliżenie

$$c(S,t+1) = c(S,t)\frac{F(S,t)}{\overline{F}(t)}$$

Jeśli założymy, że schemat S ma przystosowanie o ***ε%*** powyżej średniej

$$F(S,t) = \overline{F}(t) + \varepsilon\overline{F}(t)$$

to zakładając, że nie zmienia się w czasie i startując od ***t=0***, otrzymamy

$$c(S,t) = c(S,0)(1+\varepsilon)^t$$

Teoretyczne podstawy działania algorytmów genetycznych

Do tej pory zakładaliśmy, że funkcja ***F*** przyjmuje tylko wartości dodatnie.

W przypadku występowania wartości ujemnych, wymagane są dodatkowe odwzorowania między optymalizowaną funkcją a funkcją przystosowania.

Teoretyczne podstawy działania algorytmów genetycznych

Podsumowanie

Twierdzenie o schematach

Schematy małego rzędu, o małej rozpiętości i o przystosowaniu powyżej średniej otrzymują rosnącą wykładniczo liczbę swoich reprezentantów w kolejnych generacjach algorytmu genetycznego.

Wynika z tego, że ważnym zadaniem jest kodowanie.

Powinno ono dawać dobre schematy małego rzędu, o małej rozpiętości i o przystosowaniu powyżej średniej.

Teoretyczne podstawy działania algorytmów genetycznych

Podsumowanie

Rezultatem twierdzenie o schematach jest tzw. hipoteza cegiełek (bloków budujących)

Algorytm genetyczny dąży do osiągnięcia rezultatu bliskiemu optimum poprzez zestawienie dobrych schematów (o przystosowaniu powyżej średniej), małego rzędu i o małej rozpiętości. Schematy te nazywane są cegiełkami lub blokami budującymi, które biorą udział w wymianie informacji podczas krzyżowania.

Teoretyczne podstawy działania algorytmów genetycznych

Podsumowanie

Rezultatem twierdzenie o schematach jest tzw. hipoteza cegiełek (bloków budujących)

Hipoteza ta nie została udowodniona, ale istnieją empiryczne przesłanki potwierdzające jej słuszność.

Przy założeniu słuszności tej hipotezy, problem kodowania wydaje się być krytyczny dla działania algorytmu genetycznego – kodowanie powinno spełniać koncepcje małych bloków budujących.

Siła algorytmów genetycznych tkwi w przetwarzaniu wielkiej liczby schematów.

Algorytm idealny

Nie ma idealnego algorytmu optymalizacji, który byłby najlepszy w każdym zadaniu.

Wolpert i Macready

No Free Lunch Theorems (NLF theorems)

Jeśli algorytm A jest lepszy od algorytmu B w poszukiwaniu optimum w pewnym zadaniu poszukiwania, wtedy algorytm B będzie lepszy od A w pewnym innym zadaniu.

Algorytm idealny

Nie ma idealnego algorytmu optymalizacji, który byłby najlepszy w każdym zadaniu.

Wolpert i Macready

No Free Lunch Theorems (NLF theorems)

Biorąc pod uwagę wszystkie możliwe problemy poszukiwania, średnia jakość wszystkich algorytmów poszukiwania jest taka sama (niezależnie od algorytmu).

Czy zatem nie ma sensu opracowywanie nowych algorytmów ?

Algorytm idealny

Twierdzenie to było udowodnione dla problemów z jedną funkcją celu.

Strategie ewolucyjne

Strategie ewolucyjne (SE) a algorytmy genetyczne (AG)

Podobieństwa

- Oba działają na populacjach rozwiązań
- Korzystają z zasady selekcji i przetwarzania osobników najlepiej przystosowanych

Strategie ewolucyjne

Strategie ewolucyjne (SE) a algorytmy genetyczne (AG)

Różnice

- Reprezentacja osobników
  - Klasyczne AG – kodowanie binarne
  - SE – wektory liczb zmiennoprzecinkowych

Strategie ewolucyjne

Strategie ewolucyjne (SE) a algorytmy genetyczne (AG)

Różnice

- Selekcja
  - Klasyczne AG – do nowej populacji wybierana jest pewna liczba osobników odpowiadająca liczebnością populacji rodzicielskiej; następuje to poprzez losowanie, gdzie prawdopodobieństwo wylosowania osobnika jest uzależnione (niekoniecznie proporcjonalne) od jego wartości funkcji przystosowania. Najgorsze chromosomy mogą być wylosowane.
  - SE – tworzona jest tymczasowa populacja, a jej wielkość różni się do rozmiaru populacji rodzicielskiej. Kolejna populacja powstaje przez wybór najlepszych osobników.

Strategie ewolucyjne

Strategie ewolucyjne (SE) a algorytmy genetyczne (AG)

Różnice

- Selekcja
  - Klasyczne AG – lepiej przystosowane osobniki mogą być wybrane kilkakrotnie
  - SE – osobniki wybierane są bez powtórzeń; selekcja jest deterministyczna

Strategie ewolucyjne

Strategie ewolucyjne (SE) a algorytmy genetyczne (AG)

Różnice

- Kolejność procedur selekcji i rekombinacji
  - Klasyczne AG – najpierw selekcja, potem rekombinacja
  - SE – najpierw rekombinacja, potem selekcja. Potomek jest wynikiem krzyżowania dwóch rodziców i mutacji. Niekiedy stosuje się jedynie mutację. Utworzona zostaje populacja tymczasowa, która podlega selekcji, która redukuje rozmiar tej populacji do rozmiaru populacji rodziców.

Strategie ewolucyjne

Strategie ewolucyjne (SE) a algorytmy genetyczne (AG)

Różnice

- Parametry operatorów krzyżowania i mutacji
  - Klasyczne AG – stałe przez cały przebieg algorytmu i wspólne dla wszystkich osobników
  - SE – ulegają ciągłej zmianie (samoadaptacja parametrów)

Strategie ewolucyjne

- **Strategia (1 + 1)**
- **Strategia (μ + λ)**
- **Strategia (μ, λ)**

Strategie ewolucyjne
Strategia (1 + 1)

Przetwarzany jest jeden chromosom bazowy ***x***, początkowo losowo ustalany.

W każdej generacji w wyniku mutacji powstaje nowy osobnik ***y***.

W wyniku porównania wartości funkcji przystosowania ***F(x)*** oraz ***F(y)*** wybierany jest lepszy z nich i staje się on nowym chromosomem bazowym ***x***.

W algorytmie tym nie występuje operator krzyżowania.

Strategie ewolucyjne
Strategia (1 + 1)

Chromosom ***y*** powstaje przez dodanie do każdego z genów chromosomu ***x*** pewnej liczby losowej, generowanej zgodnie z rozkładem normalnym

$$y_i = x_i + \sigma N_i(0,1)$$

gdzie y_i oznacza ***i***-ty gen chromosomu ***y***,

x_i oznacza ***i***-ty gen chromosomu ***x***,

σ to parametr określający zasięg mutacji,

$N_i(0,1)$ to liczba losowa generowana zgodnie z rozkładem normalnym dla ***i***-tego genu

Strategie ewolucyjne
Strategia (1 + 1)

$$y_i = x_i + \sigma N_i(0,1)$$

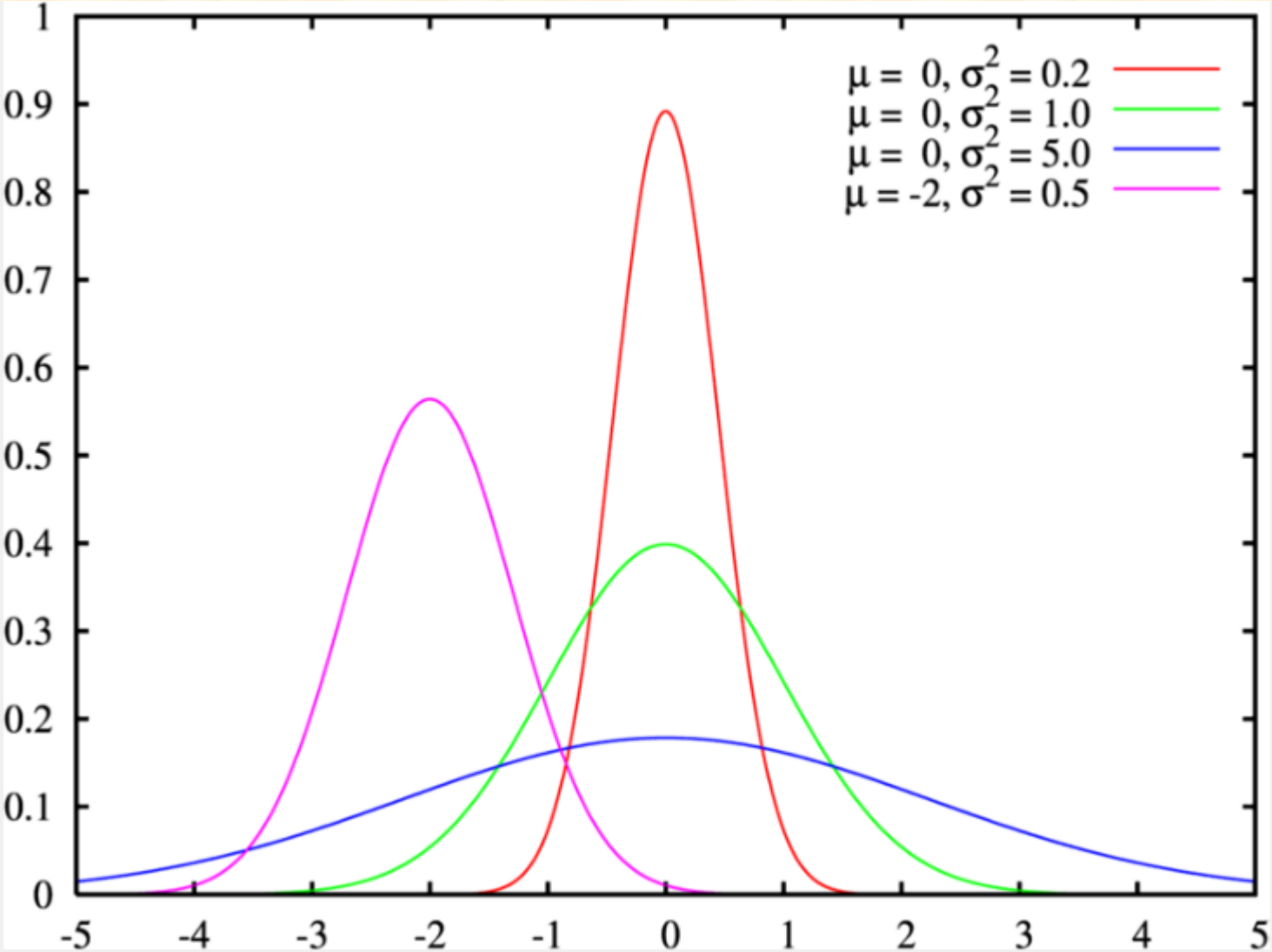
$$\phi_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

Rozkład normalny

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

Standardowy rozkład normalny

Strategie ewolucyjne
Strategia (1 + 1)



Strategie ewolucyjne
Strategia (1 + 1)

Adaptacji podlega zasięg mutacji tzn. parametr ***σ.***

Stosuje się w tym celu ***regułę 1/5 sukcesów***.

Najlepsze rezultaty w poszukiwaniu optymalnego rozwiązania uzyskuje się, gdy relacja między udanymi a wszystkimi mutacjami wynosi dokładnie 1/5.

*Gdy przez kolejnych **k** generacji stosunek udanych mutacji do wszystkich mutacji przewyższa wartość 1/5, wtedy zwiększamy wartość parametru **σ**.*

Gdy stosunek ten jest mniejszy od 1/5, wtedy zasięg mutacji jest zmniejszany.

Strategie ewolucyjne
Strategia (1 + 1)

Jeśli $\varphi(k)$ to współczynnik sukcesów operatora mutacji w poprzednich ***k*** generacjach, wtedy ***regułę 1/5 sukcesów*** można zapisać jako

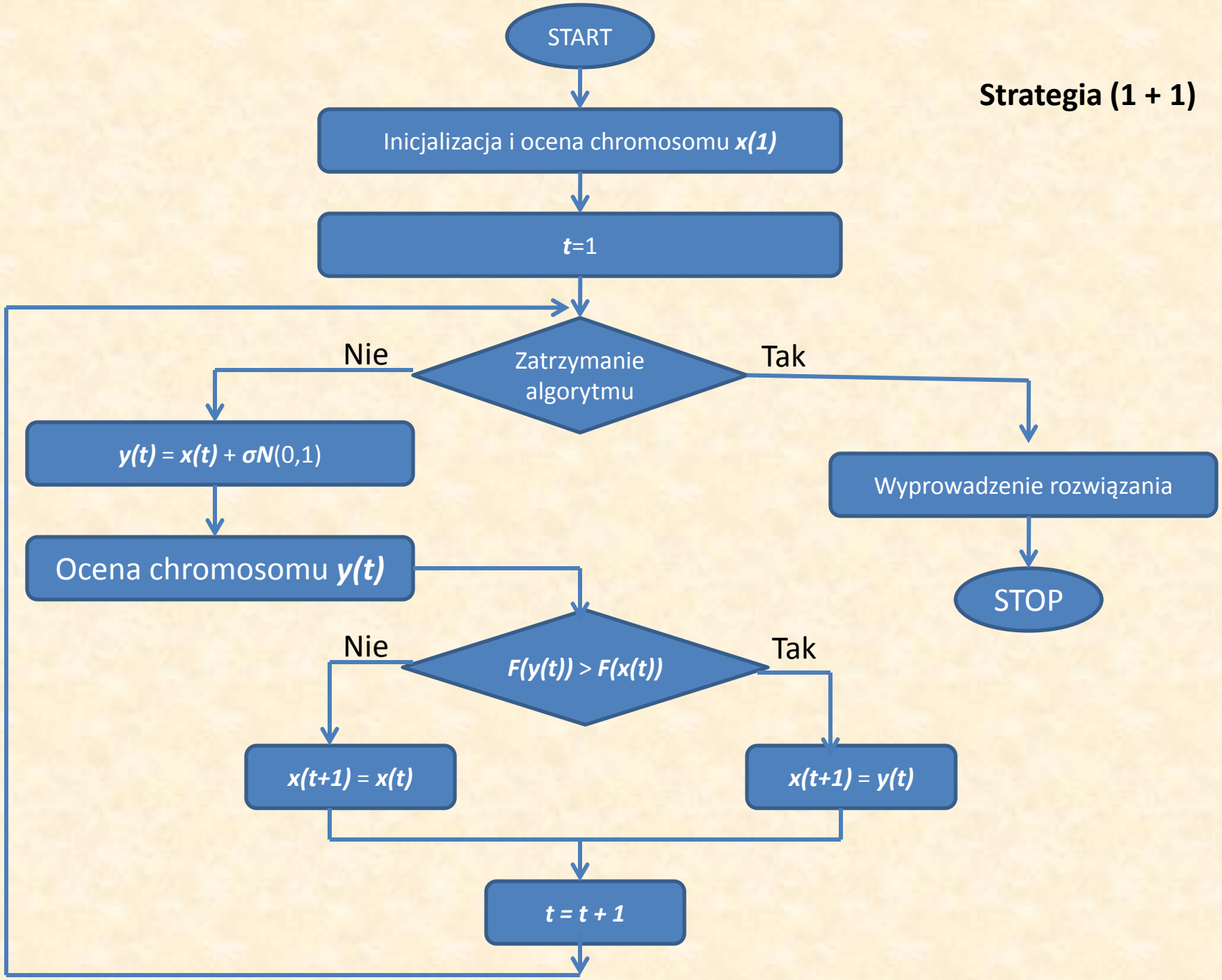
$$\sigma' = \begin{cases} c_1 \cdot \sigma & dla \quad \varphi(k) < 1/5 \\ c_2 \cdot \sigma & dla \quad \varphi(k) > 1/5 \\ \sigma & dla \quad \varphi(k) = 1/5 \end{cases}$$

gdzie współczynniki c_1 oraz c_2 regulują szybkość wzrostu lub zmniejszania się zasięgu mutacji σ.

Typowe wartości: $c_1 = 0.82$ oraz $c_2 = 1/c_1 = 1.2$.

Początkową wartość parametru σ można ustalić na $\sigma = 1$, natomiast $k=5$.

Strategia (1 + 1)



Strategie ewolucyjne
Strategia (μ + λ)

Strategia (***μ + λ***) jest rozwinięciem strategii (***1 + 1***).

Algorytm ten operuje na większej liczbie osobników i w związku z tym łatwiej utrzymuje różnorodność genotypów. Pozwala to unikać optimów lokalnych.

Algorytm zaczyna się od losowego wygenerowania początkowej populacji rodzicielskiej ***P***, zawierającej ***μ*** osobników.

Strategie ewolucyjne
Strategia (μ + λ)

Następnie tworzona jest, poprzez reprodukcję, populacja tymczasowa $\boldsymbol{T}$ zawierająca $\boldsymbol{\lambda}$ osobników, przy czym $\boldsymbol{\lambda} \geq \boldsymbol{\mu}$.

Strategie ewolucyjne
Strategia (μ + λ)

Następnie tworzona jest, poprzez reprodukcję, populacja tymczasowa ***T*** zawierająca $\boldsymbol{\lambda}$ osobników, przy czym $\boldsymbol{\lambda} \geq \boldsymbol{\mu}$.

Reprodukcja polega na wielokrotnym losowym wyborze $\boldsymbol{\lambda}$ osobników z populacji ***P*** (losowanie ze zwracaniem) i umieszczenie ich w populacji tymczasowej ***T***. (Brak nacisku selekcyjnego.)

Strategie ewolucyjne
Strategia (μ + λ)

Następnie tworzona jest, poprzez reprodukcję, populacja tymczasowa ***T*** zawierająca $\boldsymbol{\lambda}$ osobników, przy czym $\boldsymbol{\lambda} \geq \boldsymbol{\mu}$.

Reprodukcja polega na wielokrotnym losowym wyborze $\boldsymbol{\lambda}$ osobników z populacji ***P*** (losowanie ze zwracaniem) i umieszczenie ich w populacji tymczasowej ***T***.

Osobnicy z ***T*** podlegają krzyżowaniu i mutacji, w wyniku czego powstaje populacja ***O***, również o liczności $\boldsymbol{\lambda}$.

Strategie ewolucyjne
Strategia (μ + λ)

Następnie tworzona jest, poprzez reprodukcję, populacja tymczasowa ***T*** zawierająca $\boldsymbol{\lambda}$ osobników, przy czym $\boldsymbol{\lambda} \geq \boldsymbol{\mu}$.

Reprodukcja polega na wielokrotnym losowym wyborze $\boldsymbol{\lambda}$ osobników z populacji ***P*** (losowanie ze zwracaniem) i umieszczenie ich w populacji tymczasowej ***T***.

Osobnicy z ***T*** podlegają krzyżowaniu i mutacji, w wyniku czego powstaje populacja ***O***, również o liczności $\boldsymbol{\lambda}$.

Ostatnim krokiem jest wybór $\boldsymbol{\mu}$ najlepszych potomków z obydwu populacji ***P*** oraz ***O***, które będą stanowić nową populację rodzicielską.

Strategie ewolucyjne
Strategia (μ + λ)

Strategia ***(μ + λ)*** również wykorzystuje samoczynną adaptację zasięgu mutacji zgodnie z ***metodą 1/5 sukcesów***.

Każdy osobnik posiada dodatkowy chromosom ***σ***, zawierający wartości standardowych odchyleń wykorzystywanych podczas mutacji poszczególnych genów chromosomu ***x***.

Dodatkowo wprowadzono operator krzyżowania.

Strategie ewolucyjne
Strategia (μ + λ)

Ważne jest, że operacjom genetycznym ulegają obydwa chromosomy, zarówno wektor zmiennych niezależnych **x**, jak i wektor odchyleń standardowych $\boldsymbol{\sigma}$.

Strategie ewolucyjne
Strategia (μ + λ)

Krzyżowanie

Polega na wylosowaniu dwóch osobników i wymianie bądź uśrednianiu wartości ich genów.

Dwa nowe osobniki zastępują swoich rodziców.

Strategie ewolucyjne
Strategia (μ + λ)

Krzyżowanie (wymiana)

Rodzic 1 $(x^1, \sigma^1) = ([x_1^1, \ldots, x_n^1], [\sigma_1^1, \ldots, \sigma_n^1])$

Rodzic 2 $(x^2, \sigma^2) = ([x_1^2, \ldots, x_n^2], [\sigma_1^2, \ldots, \sigma_n^2])$

Potomek $(x', \sigma') = ([x_1^{q_1}, \ldots, x_n^{q_n}], [\sigma_1^{q_1}, \ldots, \sigma_n^{q_n}])$

gdzie $q_i = 1$ lub $q_i = 2$, tzn. każdy gen pochodzi z pierwszego lub drugiego wybranego rodzica).

Strategie ewolucyjne
Strategia (μ + λ)

Krzyżowanie (uśrednianie)

Rodzic 1 $(x^1, \sigma^1) = ([x_1^1, \ldots, x_n^1], [\sigma_1^1, \ldots, \sigma_n^1])$

Rodzic 2 $(x^2, \sigma^2) = ([x_1^2, \ldots, x_n^2], [\sigma_1^2, \ldots, \sigma_n^2])$

Potomek

$$(x', \sigma') = ([(x_1^1 + x_1^2)/2, \ldots, (x_n^1 + x_n^2)/2], [(\sigma_1^1 + \sigma_1^2)/2, \ldots, (\sigma_n^1 + \sigma_n^2)/2])$$

Strategie ewolucyjne
Strategia (μ + λ)

Krzyżowanie (uśrednianie ze współczynnikiem *a*)

Współczynnik ***a*** jest wylosowany z rozkładu jednostajnego ***U(0,1).***

Krzyżowanie przebiega zgodnie ze wzorami:

$$x'^{1}_{i} = a x^{1}_{i} + (1-a) x^{2}_{i}$$

$$x'^{2}_{i} = a x^{2}_{i} + (1-a) x^{1}_{i}$$

$$\sigma'^{1}_{i} = a \sigma^{1}_{i} + (1-a) \sigma^{2}_{i}$$

$$\sigma'^{2}_{i} = a \sigma^{2}_{i} + (1-a) \sigma^{1}_{i}$$

Strategie ewolucyjne
Strategia (μ + λ)

Mutacja

Wykonywana jest na pojedynczym osobniku.

Jako pierwszy poddawany jest mutacji chromosom $\boldsymbol{\sigma}=[\sigma_1, \sigma_2, \ldots, \sigma_n]$ zgodnie ze wzorem

$$\sigma'_i = \sigma_i \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1))$$

gdzie ***n*** to długość chromosomu,

N(0,1) to liczba losowa rozkładu normalnego losowana raz dla całego chromosomu ,

$\boldsymbol{N_i(0,1)}$ to liczba losowa z rozkładu normalnego losowana osobno dla każdego genu,

τ oraz ***τ'*** to parametry strategii ewolucyjnej mające wpływ na zbieżność algorytmu. Często

$$\tau' = \frac{C}{\sqrt{2n}} \qquad \tau = \frac{C}{\sqrt{2\sqrt{n}}}$$

C najczęściej ma wartość 1.

Strategie ewolucyjne
Strategia (μ + λ)

Mutacja

Nowe zakresy mutacji $\boldsymbol{\sigma}=[\sigma_1, \sigma_2, \ldots, \sigma_n]$ wpływają na zmianę wartości $\boldsymbol{x_i}$ zgodnie z zależnością:

$$x'_i = x_i + \sigma_i N_i(0,1))$$

$\boldsymbol{N_i(0,1)}$ to liczba losowa rozkładu normalnego losowana dla każdego genu w chromosomie.

Zmiana parametrów $\boldsymbol{\sigma}=[\sigma_1, \sigma_2, \ldots, \sigma_n]$ pozwala na samoadaptację procesu mutacji.

Strategie ewolucyjne
Strategia (μ + λ)

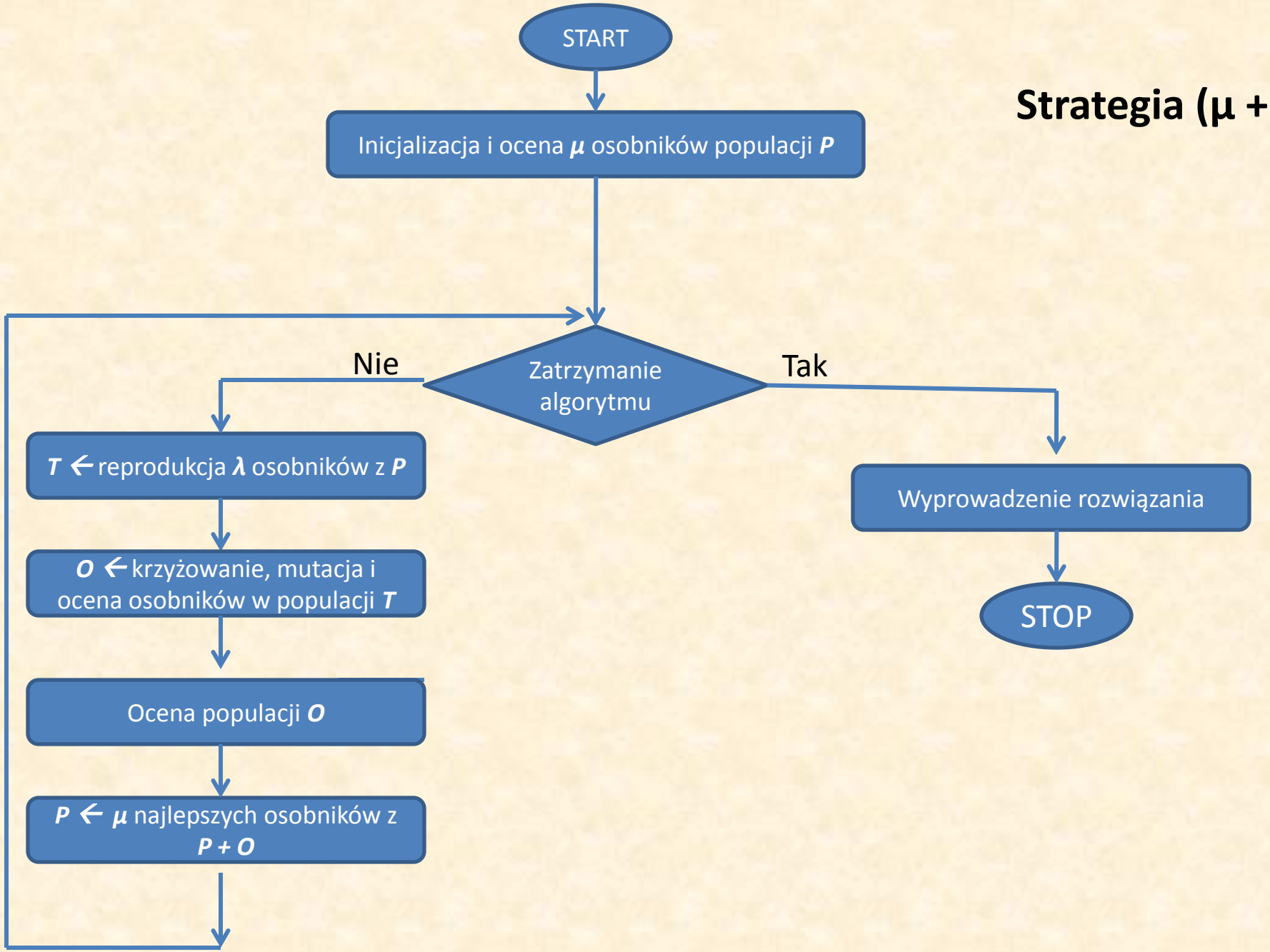
Mutacja

Zmiana parametrów $\boldsymbol{\sigma}=[\sigma_1, \sigma_2, \ldots, \sigma_n]$ pozwala na samoadaptację procesu mutacji.

Często można zaobserwować początkowy wzrost mutacji co można odczytać jako zwiększenie różnorodności populacji a tym samym poszerzenie zakresu poszukiwań.

Zaraz potem następuje gwałtowny spadek i zmniejszają się różnice w chromosomach wywoływane operatorem genetycznym. Osobniki zaczynają oscylować wokół ostatecznego rozwiązania.

Strategia (μ + λ)



Strategie ewolucyjne
Strategia (μ, λ)

Strategia ***(μ, λ)*** ma podobne działanie jak strategia ***(μ + λ)***.

Różnica polega na tym, że nową populację ***P*** zawierającą ***μ*** osobników wybiera się tylko spośród najlepszych ***λ*** osobników populacji ***O***.

Aby było to możliwe, musi być spełniony warunek

$$\mu > \lambda$$

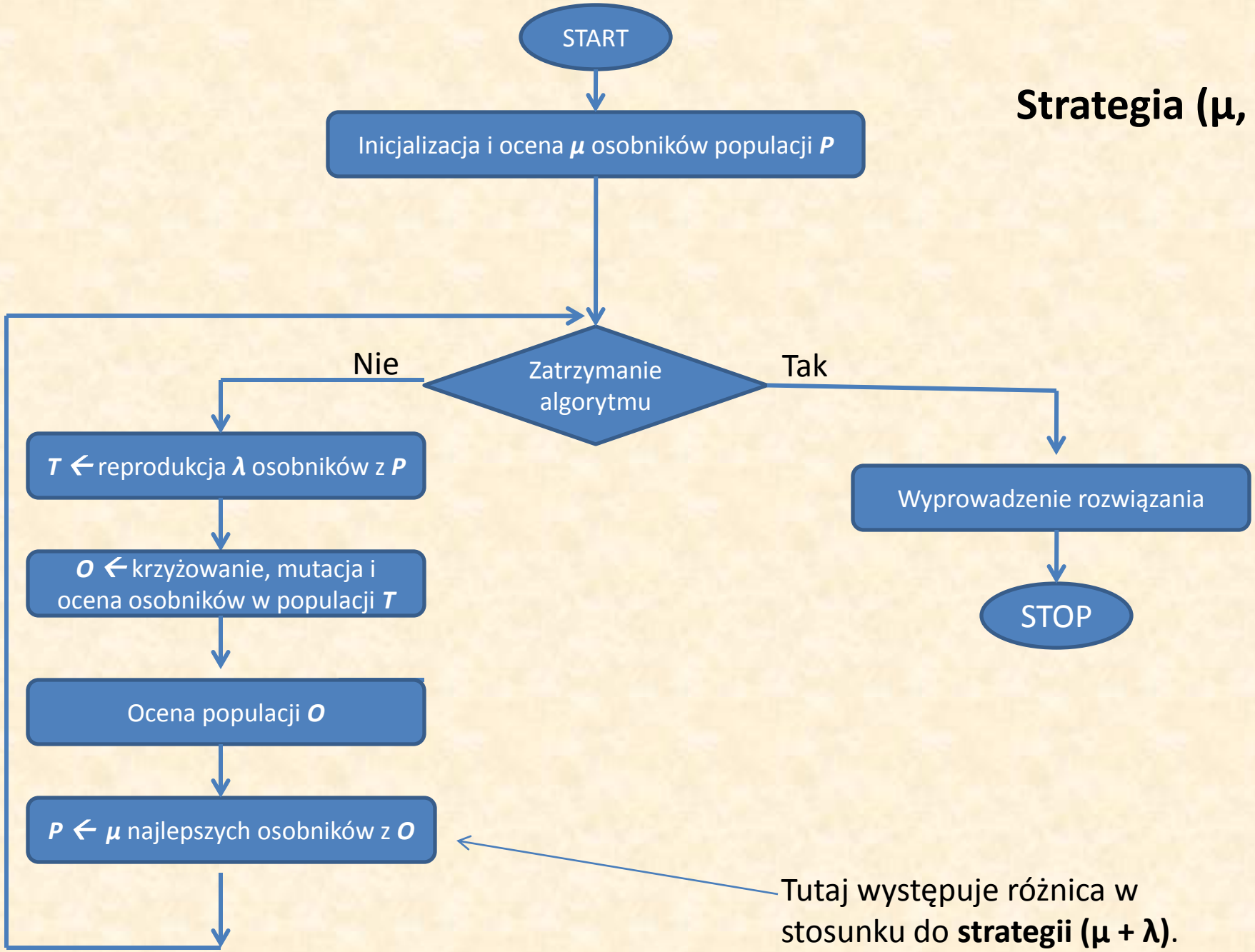
Strategie ewolucyjne
Strategia (μ, λ)

Strategia ***(μ, λ)*** ma tę zaletę nad strategią ***(μ + λ)***, że w przypadku tej drugiej populacja mogła być zdominowana przez jednego osobnika o dużej wartości funkcji przystosowania, ale zbyt dużych lub zbyt małych wartościach odchyleń standardowych, co utrudnia znalezienie lepszych rozwiązań.

Strategia ***(μ, λ)*** gwarantuje, że stare osobniki nie przechodzą do nowej puli rodzicielskiej.

Operatory genetyczne nie różnią się od tych w strategii ***(μ + λ).***

Strategia (μ, λ)



Tutaj występuje różnica w stosunku do **strategii (μ + λ)**.

Programowanie ewolucyjne (PE)

Początkowo rozwijane było w kontekście odkrywania gramatyki nieznanego języka. Gramatyka była modelowana za pomocą automatu skończonego, które podlegał ewolucji.

Później zastosowano PE do optymalizacji numerycznej.

Programowanie ewolucyjne (PE)

Programowanie ewolucyjne (PE) wykazuje podobieństwo do strategii ewolucyjnej **(μ + λ)**.

Różnica polega na tym, że podczas każdej generacji algorytmu PE nowa populacja ***O*** jest tworzona przez mutację każdego z osobników populacji rodzicielskiej ***P***.

W SE **(μ, λ)** każdy osobnik ma natomiast jednakową szansę na pojawienie się w populacji tymczasowej ***T***.

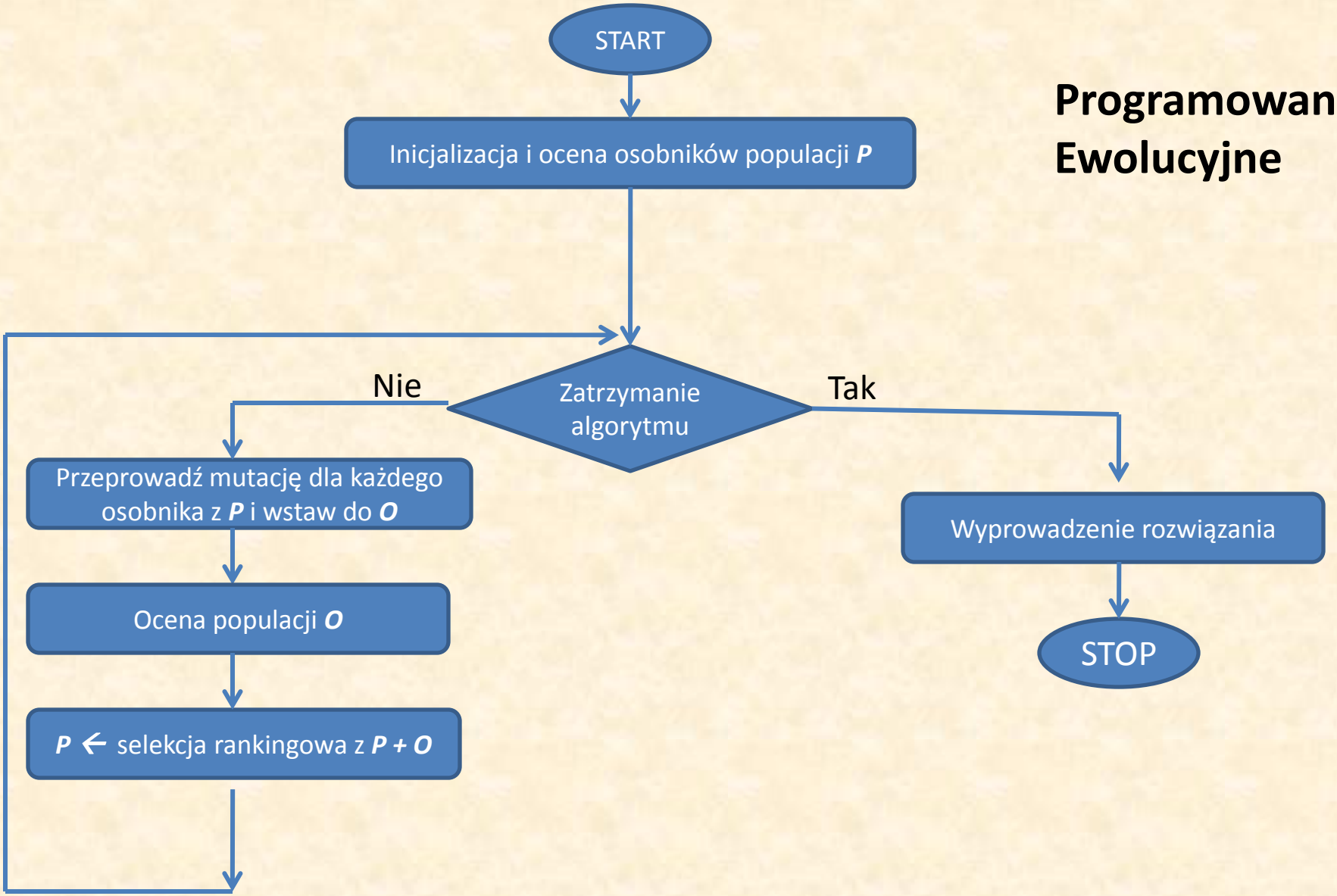
Programowanie ewolucyjne (PE)

W PE populacje ***P*** oraz ***O*** są tak samo liczne, tzn. $\mu = \lambda$.

Ostatecznie nowa populacja rodzicielska ***P*** jest tworzona za pomocą selekcji rankingowej, której podlegają zarówno ze starej populacji ***P*** jak i osobniki zmutowane z populacji ***O***.

Mutacja w PE polega na losowej perturbacji wartości poszczególnych genów.

Programowanie Ewolucyjne



Programowanie genetyczne (PG)

Jest to rozszerzenie klasycznego algorytmu genetycznego.

Wykorzystywane do automatycznego generowania programów komputerowych.

Stosuje język programowania LISP, w którym program jest reprezentowany tak samo jak dane, tzn. w postaci drzewa.

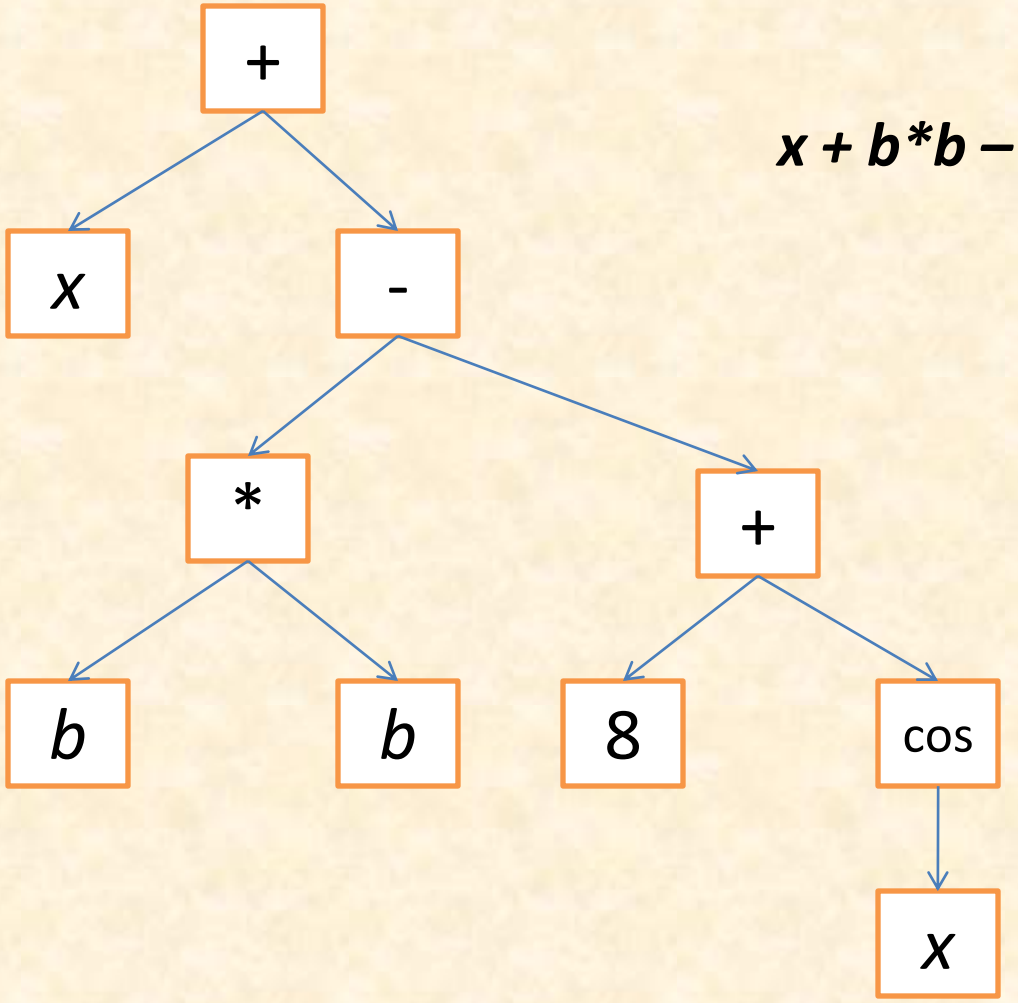
Programowanie genetyczne (PG)

Stosowane jest kodowanie drzewiaste.

Informacja zawarta jest w węzłach, a krawędzie określają wzajemne stosunki.

Wymaga to zastosowania specjalnych operatorów różnicowania.

Programowanie genetyczne (PG)



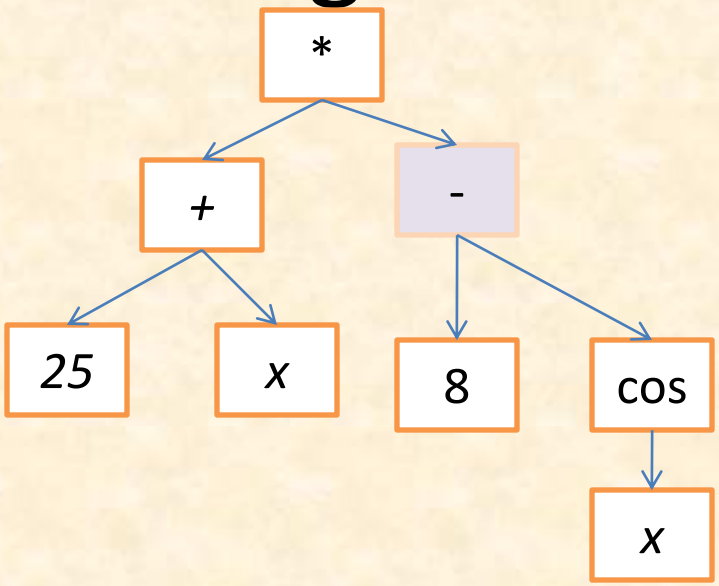
x + b*b – (8 + cos(x))

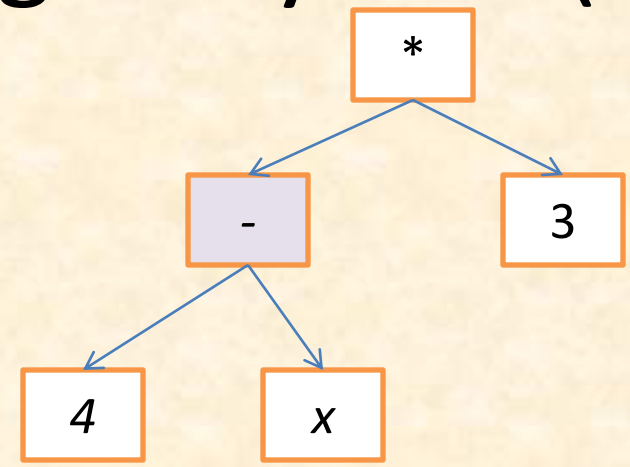
Programowanie genetyczne (PG)

Krzyżowanie

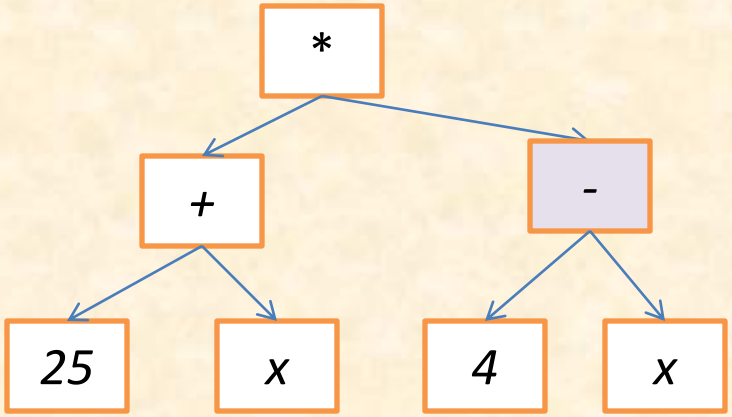
Może polegać na losowym wyborze węzła w każdym rodzicu i wymianie odpowiednich poddrzew związanych z tymi węzłami.

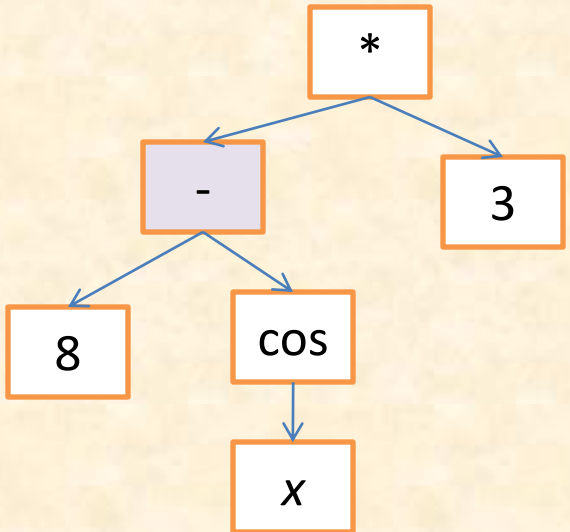
Programowanie genetyczne (PG)





krzyżowanie





Programowanie genetyczne (PG)

Mutacja

Może polegać na losowym wyborze węzła w danym osobniku a następnie:

- Zmianie zawartości węzła
- Jeśli jest to węzeł terminalny – wygenerowaniu dla niego losowego poddrzewa
- Usunięciu węzła razem ze związanym z nim poddrzewem
- Zastąpieniu poddrzewa związanego z wylosowanym węzłem przez losowe poddrzewo

Eksploracja i eksploatacja

Poprzez promowanie lepszych osobników, algorytmy ewolucyjne mają tendencję do podążania w kierunku lepszych rozwiązań.

Zdolność poprawiania średniej wartości przystosowania w populacji nazywa się **naciskiem selektywnym**.

Duży nacisk selektywny występuje wtedy, gdy większa jest wartość oczekiwana liczby kopii lepszego osobnika niż wartość oczekiwana liczby kopii gorszego osobnika.

Eksploracja i eksploatacja

Eksploracja – przeszukiwanie całej przestrzeni rozwiązań w celu przybliżenia się do (zlokalizowania) globalnego optimum.

Eksploatacja – poruszanie się po fragmencie przestrzeni w pobliżu domniemanego optimum.

Eksploracja i eksploatacja

Eksploracja zwiększamy poprzez zmniejszenie nacisku selektywnego – wtedy słabsze osobniki mają większą szanse na przeżycie i odkrywanie nowych obszarów w przestrzeni poszukiwań, gdzie może potencjalnie znajdować się globalne optimum.

Eksploatacja zwiększamy poprzez zwiększenie nacisku selektywnego – wtedy aktualnie najlepsze osobniki zaczynają dominować. Przedwczesny zbyt duży nacisk na eksploatację może powodować utknięcie algorytmu w optimach lokalnych.

Eksploracja i eksploatacja

Wpływ na eksplorację i eksploatację mają również parametry algorytmu ewolucyjnego, takie jak:

- Prawdopodobieństwo krzyżowania
- Prawdopodobieństwo mutacji

Zwiększenie ich powoduje zwiększenie różnorodności populacji, a przez to zwiększenie eksploracji.

Zmniejszenie ich powoduje zmniejszenie różnorodności i w wyniku zwiększenie eksploatacji.

Eksploracja i eksploatacja

W dobrze działającym algorytmie ewolucyjnym eksploatacja i eksploracja są „zrównoważone”.

Metody selekcji

Rodzaj selekcji mocno wpływa na proporcje pomiędzy eksploracją a eksploatacją.

Metody selekcji

Selekcja za pomocą koła ruletki (proporcjonalna)

- Procedura losowa
- Prawdopodobieństwo wyboru jest proporcjonalne do jakości przystosowania osobnika

$$p_s(ch_i) = \frac{F(ch_i)}{\sum_{j=1}^{K} F(ch_j)}$$

Metody selekcji

Selekcja za pomocą koła ruletki (proporcjonalna)

- Dany osobnik może zostać wybrany więcej niż jeden raz
- Z tej metody można korzystać jeśli wartości funkcji przystosowania są dodatnie
- Nadaje się jedynie do maksymalizacji funkcji w przypadku minimalizacji potrzebne są dodatkowe zabiegi)

Metody selekcji

Selekcja za pomocą koła ruletki (proporcjonalna)

- Wady:
  - Osobniki o małej wartości funkcji przystosowania są wcześnie eliminowane z populacji, co może prowadzić do przedwczesnej zbieżności algorytmu do optimum lokalnego
  - Można temu zapobiegać stosując skalowanie funkcji przystosowania

Metody selekcji

Selekcja rankingowa (rangowa)

Osobniki ustawiane są kolejno zgodnie z wartością funkcji przystosowania – od najlepszego do najgorszego. Każdy osobnik ma numer określający jego pozycję na liście, czyli swoją rangę.

Liczba kopii każdego osobnika wprowadzana do populacji ***M(t)*** jest zdefiniowana przez wcześniej ustaloną funkcję, która zależy od rangi osobnika.

Metody selekcji

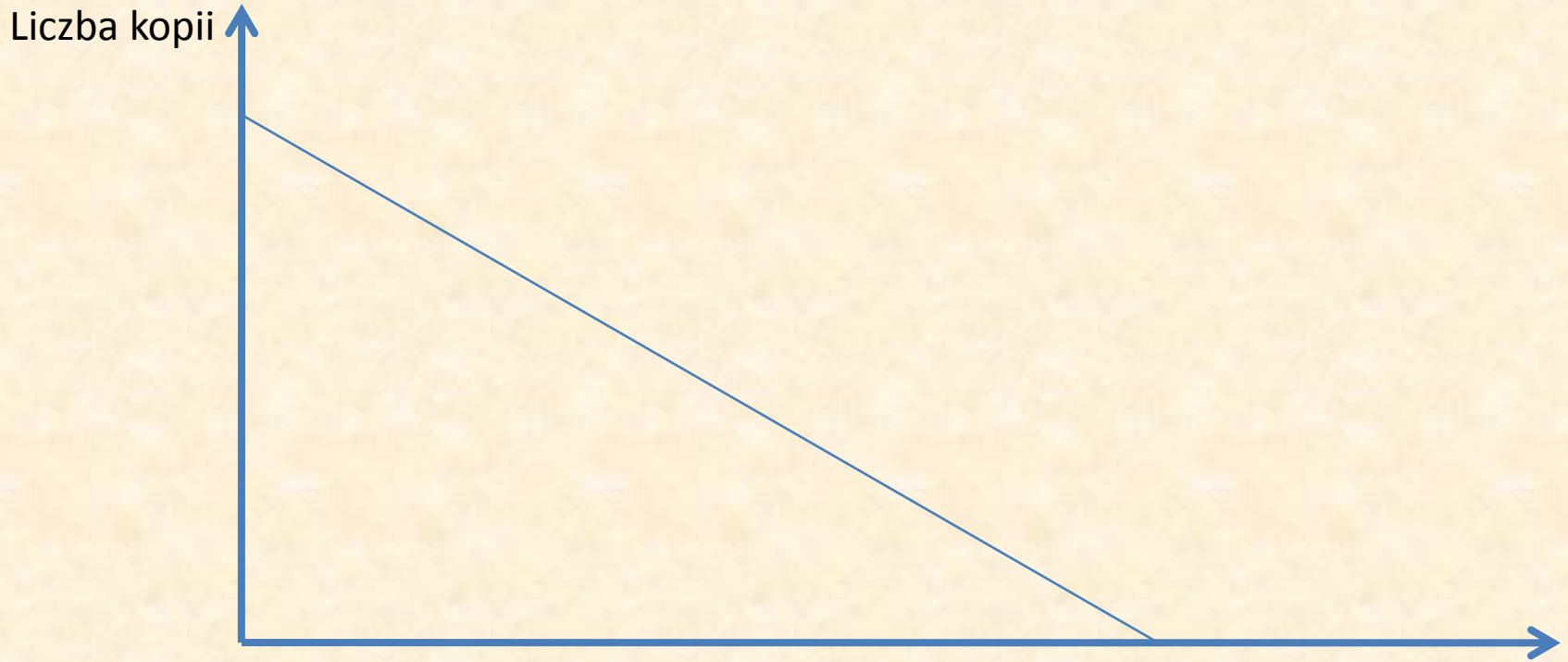
Selekcja rankingowa (rangowa)

Zalety

- Liczba kopii możliwych do osiągnięcia dla gorszych osobników nie zależy od tego, o ile są gorsze od osobnika najlepszego – co było problemem w selekcji proporcjonalnej.
- Można bez problemu wykorzystać zarówno do minimalizacji jak i maksymalizacji funkcji.

Metody selekcji

Selekcja rankingowa (rangowa)



Przykład funkcji użytej w selekcji rankingowej

Metody selekcji

Selekcja turniejowa

- Osobniki w populacji zostają podzielone na podgrupy (losowo, z powtórzeniami), najczęściej 2 lub 3 osobniki
- Z każdej z nich wybierany jest osobnik najlepszy (deterministycznie lub z uwzględnieniem elementu losowego)

Metody selekcji

Selekcja turniejowa

- Nadaje się zarówno do minimalizacji jak i maksymalizacji funkcji
- Praktyka pokazuje, że działa lepiej niż selekcja za pomocą koła ruletki

Metody selekcji

Inne metody selekcji są najczęściej modyfikacją selekcji proporcjonalnej, rankingowej i turniejowej.

Metody selekcji

Selekcja progowa

- Szczególny przypadek selekcji rankingowej
- Funkcja określająca prawdopodobieństwo przejścia osobnika do puli rodzicielskiej ma postać progu
- Wartość progu wpływa bezpośrednio na nacisk selektywny

Metody selekcji

Selekcja progowa

$$p_s(ch_i) = \begin{cases} \dfrac{1}{\rho \cdot rozm\_pop} & dla \quad 0 \le r(ch_i) \le \rho \cdot rozm\_pop \\ 0 & w \quad przeciwnym \quad wypadku \end{cases}$$

ρ - to parametr sterującym naciskiem selektywnym, im ta wartość jest mniejsza, tym mniej osobników będzie reprodukować

r(ch$_i$) – ranga chromosomu ***ch$_i$***

Metody selekcji

Selekcja stłoczenia (ang. *crowding selection*)

- Nowo utworzone osobniki zastępują najbardziej podobne osobniki rodzicielskie, niezależnie od wartości ich funkcji przystosowania
- Celem jest zachowanie jak największej różnorodności osobników w populacji
- Parametr ***cf*** (ang. *crowding factor*) określa liczbę rodziców podobnych do nowego osobnika, spośród których zostanie wylosowany osobnik do usunięcia

Metody selekcji

Strategia elitarna (ang. *elitist strategy*)

- Polega na ochronie najlepszych chromosomów w kolejnych populacjach
- Najlepszy osobnik (lub najlepsze osobniki) zawsze przechodzą do następnej populacji

Metody selekcji

Częściowa wymiana poulacji (algorytm z ustalonym stanem, ang. *steady-state*)

- Część populacji przechodzi bez zmian do następnego pokolenia – tzn. nie podlega krzyżowaniu ani mutacji
- W praktyce często tylko jeden lub dwa osobniki podlegają wymianie, zamiast stosowania krzyżowania i mutacji w obrębie całej populacji.

Skalowanie funkcji przystosowania

Powody

- Aby zapobiec przedwczesnej zbieżności algorytmu (ochrona przed dominacją najlepszego osobnika)
- W końcowej fazie, gdy populacja zachowuje znaczną różnorodność, ale średnia wartość przystosowania niewiele odbiega od maksymalnej
- Skalowanie zapobiega sytuacji, w której osobniki najlepsze i najgorsze otrzymują prawie identyczne funkcje oceny

Skalowanie funkcji przystosowania

Skalowanie to odpowiednie przekształcenie funkcji przystosowania

Najczęściej wykorzystywane:

- Skalowanie liniowe
- Obcinanie typu sigma
- Skalowanie potęgą

Skalowanie funkcji przystosowania

Skalowanie liniowe

$$F' = a \cdot F + b$$

F – oryginalna funkcja przystosowania

F' – funkcja przystosowanie po skalowaniu

Parametry ***a*** i ***b*** dobiera się tak, by średnia wartość funkcji po skalowaniu była równa średniej wartości przed skalowaniem, a maksymalna wartość funkcji po skalowaniu była wielokrotnością średniej wartości funkcji przystosowania.

Współczynnik wielokrotności przyjmuje się między **1.2** a **2**

Skalowanie funkcji przystosowania

Obcinanie typu sigma

$$F' = F + (\overline{F} - c \cdot \sigma)$$

F – oryginalna funkcja przystosowania

F' – funkcja przystosowanie po skalowaniu

c – mała liczba naturalna (zazwyczaj od 1 do 5)

σ – odchylenie standardowe w populacji

$\overline{F}$ - średnia wartość funkcji przystosowania

Ujemne wartości F_i ustawiane są na 0.

Skalowanie funkcji przystosowania

Skalowanie potęgą

$$F' = F^k$$

k – liczba bliska 1, np. ***k=1.005***

Ujemne wartości F_i ustawiane są na 0.

Skalowanie funkcji przystosowania

Skalowanie potęgą
np. ***k=2.5***

$F(ch_1)=13.2$
$F(ch_2)=13.1$
$F(ch_3)=12.8$

Po normalizacji (dzielenie każdej wartości przez sumę wszystkich):

$F(ch_1)=0.33759591$
$F(ch_2)=0.33503836$
$F(ch_3)=0.32736573$

$F(ch_1)/F(ch_2)=1.00763358779$
$F(ch_1)/F(ch_3)=1.03125$

Skalowanie funkcji przystosowania

Skalowanie potęgą
np. ***k=2.5***

$F(ch_1)/F(ch_2)=1.00763358779$
$F(ch_1)/F(ch_3)=1.03125$

Skalowanie: $F' = F^{2.5}$ i ponowna normalizacja:

$F(ch_1)=0.34398263$
$F(ch_2)=0.33750478$
$F(ch_3)=0.31851259$

$F(ch_1)/F(ch_2)=1.01919336771$
$F(ch_1)/F(ch_3)=1.07996555452$

Metody kodowania

Kodowanie binarne

Wykorzystuje się zapis dwójkowy liczb dziesiętnych – każdy bit odpowiada kolejnej potędze liczby 2.

$$[10011]_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19_{10}$$

Metody kodowania

Kodowanie binarne

W przypadku kodowania liczb rzeczywistych z przedziału **$[a_i, b_i]$** za pomocą **n_i** bitów chromosomu **x**, gdzie bity kodujące ***i***-tą zmienną mają numery od **s(i)** do **$s(i)+n_i-1$**

$$x_i = a_i + \frac{b_i - a_i}{2^{n_i} - 1} \sum_{j=0}^{n_i - 1} 2^j \, x_{s(i)+j}$$

Metody kodowania

Kodowanie binarne

Alfabet binarny

- Jest prosty i można stosować proste operatory genetyczne
- Nie jest naturalny dla wielu problemów
- Powoduje powstanie bardzo dużych przestrzeni rozwiązań (np. wiele zmiennych przy dużej wymaganej dokładności powoduje powstanie bardzo długich chromosomów)

Metody kodowania

Binarne – kodowanie Graya

W kodowaniu tym ciągi binarne odpowiadające dwóm kolejnym liczbom całkowitym różnią się tylko jednym bitem.

Może się to okazać korzystne ze względu na mutację.

Metody kodowania

Binarne – kodowanie Graya

Lp.	Kod binarny	Kod Gray'a
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Metody kodowania

Naturalny kod dwójkowy → kodowanie Gray'a

1. Zapisujemy liczbę w kodzie dwójkowym i uzupełniamy zerami do wymaganej liczby bitów:

Wyraz 7

Kod dwójkowy 0111

Metody kodowania

Naturalny kod dwójkowy → kodowanie Gray'a

2. Pod spodem wypisujemy ten sam numer przesunięty w prawo o 1 bit. Najmniej znaczący bit odrzucamy. Na początku dopisujemy bit o wartości 0.

0111 kod dwójkowy
0011 przesunięcie

Metody kodowania

Naturalny kod dwójkowy → kodowanie Gray'a

3. Nad odpowiadającymi sobie bitami wykonujemy operację logiczną XOR. Wynik jest wyrazem w kodzie Gray'a.

```
    0111    kod dwójkowy
    0011    przesunięcie
XOR ------
    0100    kod Graya
```

Metody kodowania

kodowanie Gray’a → Naturalny kod dwójkowy

1. Przepisujemy najstarszy bit z kodu Gray'a do najstarszego bitu słowa dwójkowego. Bity te w obu kodach mają tę samą wartość.

1110 kod Gray’a

1 kod dwójkowy

Metody kodowania

kodowanie Gray’a → Naturalny kod dwójkowy

2. Otrzymany bit umieszczamy na pozycji przesuniętej w prawo o jedno miejsce i wykonujemy operację XOR z bitem kodu Gray'a. W ten sposób otrzymujemy kolejne bity kodu dwójkowego.

1110 kod Gray’a

1

10 kod dwójkowy

Metody kodowania

kodowanie Gray'a → Naturalny kod dwójkowy

3. Identyczną operację wykonujemy z poprzednio otrzymanym bitem otrzymując kolejny bit kodu dwójkowego.

1110 kod Gray'a

0

101 kod dwójkowy

Metody kodowania

kodowanie Gray'a → Naturalny kod dwójkowy

3. Identyczną operację wykonujemy z poprzednio otrzymanym bitem otrzymując kolejny bit kodu dwójkowego.

1110 kod Gray'a

1

1011 kod dwójkowy

Metody kodowania

Kodowanie logarytmiczne

Stosowane w celu zmniejszenia długości chromosomów w algorytmie genetycznym.

Wykorzystuje się je w problemach o wielu parametrach i dużych przestrzeniach poszukiwań.

Metody kodowania

Kodowanie logarytmiczne

- Pierwszy bit **α** jest bitem znaku funkcji wykładniczej
- Drugi bit **β** jest bitem znaku wykładnika funkcji wykładniczej
- Pozostałe bity **bin** reprezentują wykładnik funkcji wykładniczej

$$[\alpha\beta bin]_2 = (-1)^{\alpha} e^{(-1)^{\beta}[bin]_{10}}$$

Metody kodowania

Kodowanie logarytmiczne

Przykład

$$[01110]_2 = (-1)^0 e^{(-1)^1 [110]_{10}} = 0.002478752$$

Za pomocą 5 bitów można zakodować liczby z przedziału **$[-e^7, e^7]$** co jest dużo większym zakresem niż w przypadku zwykłego kodowania binarnego **[0, 31]**.

Metody kodowania

Kodowanie za pomocą wektora liczb rzeczywistych

Wymaga specjalnych operatorów krzyżowania i mutacji (porównaj ze strategiami ewolucyjnymi).

Rodzaje krzyżowań

Krzyżowanie jednopunktowe

Występuje w klasycznym algorytmie genetycznym.

Rodzice:

ch_1=110001**0100**

ch_2=**01110**10110

Potomkowie:

ch'_1=1100010110

ch'_2=**0111010100**

Rodzaje krzyżowań

Krzyżowanie dwupunktowe (ang. *two-point crossover*)

Potomkowie dziedziczą fragmenty chromosomów wyznaczone przez dwa wylosowane punkty przecięcia.

Rodzice:

ch_1=110**0010**100

ch_2=**011**1010**110**

Potomkowie:

ch'_1=1101010100

ch'_2=**0110010110**

Rodzaje krzyżowań

Krzyżowanie wielopunktowe (ang. *multiple-point crossover*)

Jest uogólnieniem krzyżowań jednopunktowego i dwupunktowego i charakteryzuje się odpowiednią liczbą wylosowanych punktów krzyżowania.

Rodzaje krzyżowań

Krzyżowanie równomierne (jednolite, jednostajne) (ang. uniform crossover)

Odbywa się zgodnie z wylosowanym wzorcem wskazującym, które geny są dziedziczone od pierwszego rodzica, a które od drugiego.

Krzyżowanie to można stosować do różnych rodzajów kodowania (nie tylko binarnego).

Rodzice:

ch_1=1101001110

ch_2=1011111100

wylosowany wzorzec: 0101101110

Potomkowie:

ch'_1=1001101100

ch'_2=1111011110

Jedynki we wzorcu wskazują geny podlegające wymianie.

Rodzaje mutacji

Mutacja wprowadza pewne urozmaicenie w populacji, przez co algorytm może uciec z pułapek optimów lokalnych.

W klasycznym algorytmie genetycznym jego rola jest raczej mała

W innych rodzajach algorytmów ewolucyjnych mutacja jest operacją dominującą – porównaj ze strategiami ewolucyjnymi.

Rodzaje mutacji

Mutacja chromosomu binarnego

Operacji mutacji w chromosomie podlegają te geny, dla których wylosowana liczba z zakresu [0, 1] jest mniejsza od prawdopodobieństw a mutacji p_m. Mutacja polega na mutacji wartości bitu lub wylosowaniu nowej wartości ze zbioru {0, 1}.

Inwersja

Inwersja

Inwersja działa na pojedynczym chromosomie. Zmienia ona kolejność alleli między dwoma losowo wybranymi pozycjami (locus) chromosomu.

Operator ten został zdefiniowany poprzez wzorowanie się na biologicznym procesie **inwersji chromosomowej**.

Inwersja nie jest zbyt często stosowana w algorytmach ewolucyjnych.

Przykład:

010**011**101010

010**1110**01010

Rodzaje mutacji

Mutacja chromosomu liczb rzeczywistych

W przypadku liczb rzeczywistych nie można przeprowadzić prostej negacji.

Mutację dla genu w postaci liczby rzeczywistej można przeprowadzić na parę sposobów.

Rodzaje mutacji

Mutacja chromosomu liczb rzeczywistych

Zakładamy, że wartość genu x_i należy do przedziału $[a_i, b_i]$.

Dla każdego genu losujemy liczbę z zakresu **[0, 1]** i jeśli jest ona mniejsza niż prawdopodobieństwo mutacji p_m, to mutacja przebiega zgodnie ze wzorem:

$$y_i = a_i + (b_i - a_i)U_i(0, 1)$$

gdzie y_i to nowa wartość genu x_i, a $U_i(0, 1)$ to liczba losowa wygenerowana z rozkładu równomiernego z przedziału **[0,1]**.

Wartość y_i nie zależy od x_i – im większe p_m, tym bardziej algorytm zbliża się do przeszukiwania losowego.

Rodzaje mutacji

Mutacja chromosomu liczb rzeczywistych

Częściej stosowana jest mutacja polegająca na dodaniu do aktualnej wartości genu x_i wartości z_i pewnej zmiennej losowej (niekoniecznie pochodzącej z rozkładu równomiernego; częściej stosowany jest rozkład normalny lub rozkład Cauchy'ego).

Dla każdego genu losujemy liczbę z zakresu **[0, 1]** i jeśli jest ona mniejsza niż prawdopodobieństwo mutacji p_m, to mutacja przebiega zgodnie ze wzorem:

$y_i = x_i + z_i$

Rodzaje mutacji

Mutacja chromosomu liczb rzeczywistych

Inne metody mutacji dla chromosomów liczb rzeczywistych zdefiniowane zostały przy omawianiu strategii ewolucyjnych.

Parametry Algorytmu Ewolucyjnego

Parametry Algorytmu Ewolucyjnego

- Algorytm ewolucyjny ma często więcej parametrów sterujących niż inne metody heurystyczne.
- Jest to zarówno zaleta AE (elastyczność, możliwość dostosowania do problemu) jak i ich wada (ciężko dobrać odpowiedni zestaw parametrów)

Parametry Algorytmu Ewolucyjnego

- Co jest zmieniane
  - Reprezentacja osobników
  - Funkcja oceny
  - Operatory różnicowania
  - Selekcja
  - Populacja (rozmiar)

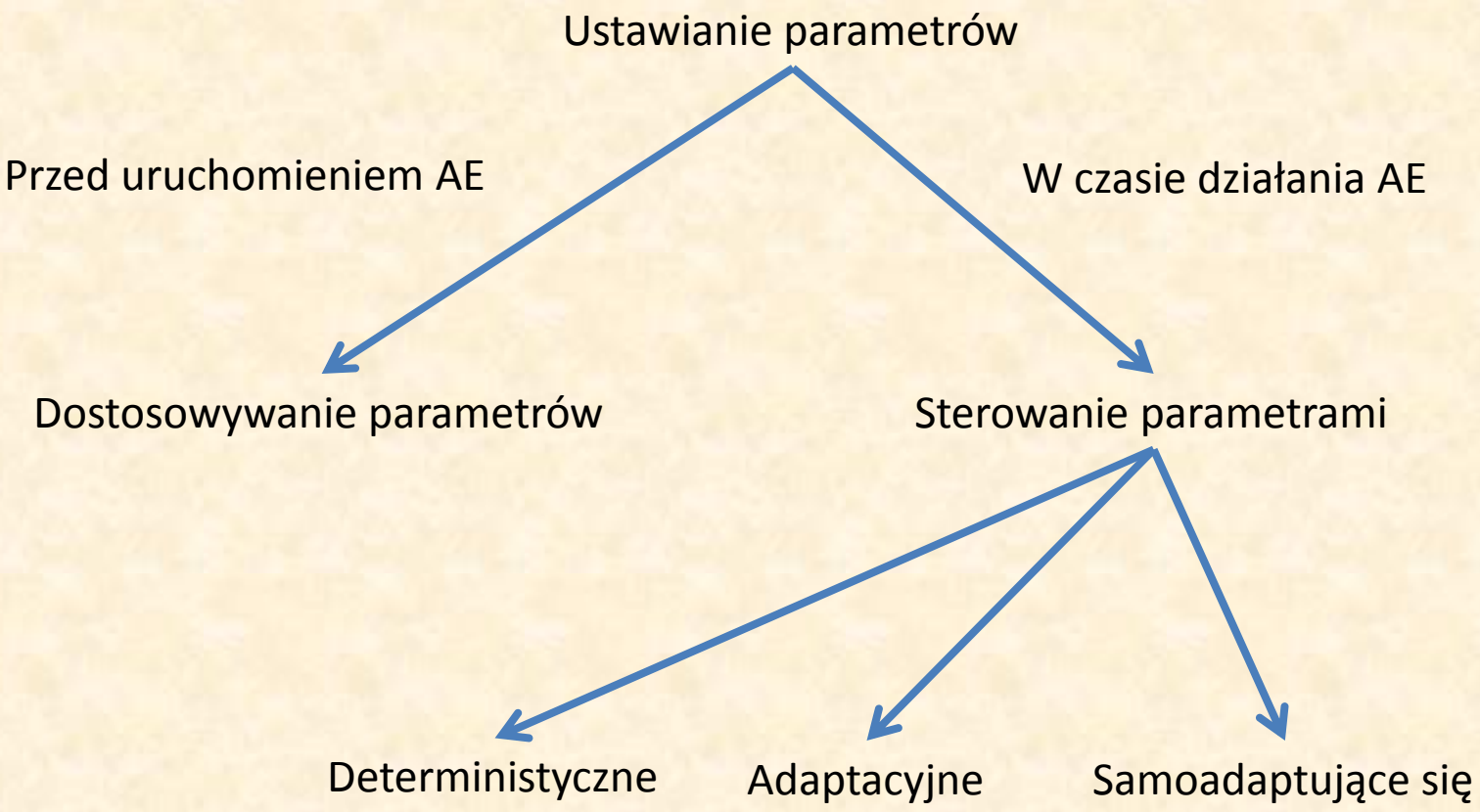
Parametry Algorytmu Ewolucyjnego

- Jak jest zmieniane
  - Deterministycznie
  - Adaptacyjnie
  - Samoadaptacyjnie

Parametry Algorytmu Ewolucyjnego

- Zakres wpływu zmian parametru
  - Pojedyncza zmienna
  - Osobnik
  - Populacja
  - Cała funkcja celu

Parametry Algorytmu Ewolucyjnego



Parametry Algorytmu Ewolucyjnego

Analiza matematyczna i teoria

Bardzo słabo rozwinięta.

Dobre wyniki w bardzo wąskich dziedzinach, np. badania Rechenberga, który wskazał optymalne wartości parametrów mutacji dla pewnej klasy funkcji podlegających optymalizacji w przypadku populacji składającej się z jednego osobnika.

Parametry Algorytmu Ewolucyjnego

Ręczny dobór parametrów

- Bazujący na doświadczeniu (swoim i innych)
- Czasochłonny
- Zniechęcający
- Ewolucja jest procesem dynamicznym, parametry mogą mieć różne wartości optymalne na różnych etapach algorytmu

Parametry Algorytmu Ewolucyjnego

Dynamiczny (zależny od czasu) dobór parametrów

Np. prawdopodobieństwo mutacji

$p_m(t)$ zamiast p_m

- Jaka powinna być zależność od czasu?
- Często lepszy niż stała wartość parametru – często nieoptymalna postać ***p(t)*** daje lepsze wyniki niż nieoptymalna wartość stała ***p***
- Nie bierze pod uwagę aktualnego stanu ewolucji

Parametry Algorytmu Ewolucyjnego

Przykład – prawdopodobieństwo mutacji wykorzystującej rozkład normalny

$$x'_i = x_i + N(0, \sigma(t))$$

$$\sigma(t) = 1 - 0.9 \cdot \frac{t}{T}$$

t – numer aktualnej generacji

T - maksymalna liczba generacji

Parametry Algorytmu Ewolucyjnego

Stosowanie heurystyk

Np. reguła 1/5 prób

Prawdopodobieństwo mutacji powinno być zwiększone, jeśli w wyniku mutacji dostajemy poprawę częściej niż w co piątej mutacji, oraz zmniejszone jeśli poprawa występuje rzadziej niż co piąty raz.

(Porównaj ze strategiami ewolucyjnymi)

Parametry Algorytmu Ewolucyjnego

Stosowanie podejścia adaptacyjnego i samoadaptacyjnego (ewolucja ewolucji)

- Przy zmianach wartości parametrów uwzględniać aktualny stan populacji (postęp poszukiwań, różnorodność populacji, itp.)
- Parametr uczynić składową chromosomu, która również podlega ewolucji (porównaj ze strategią ewolucyjną)
- Użyć algorytmu ewolucyjnego gdzie osobniki kodują zestawy parametrów innego algorytmu ewolucyjnego i są oceniane na podstawie wyników tego algorytmu korzystającego z danego zestawu parametrów

Parametry Algorytmu Ewolucyjnego

Przykład: Zmiana reprezentacji

W trakcie działania algorytmu ciąg bitów chromosomu zmienia interpretację poprzez zawężanie przedziałów zmiennych wokół kolejnych najlepszych rozwiązań.

Parametry Algorytmu Ewolucyjnego

Przykład: Zmiana funkcji celu

Zmiana funkcji kary w trakcie działania algorytmu ewolucyjnego do problemu z ograniczeniami (patrz kolejne slajdy).

Parametry Algorytmu Ewolucyjnego

Przykład: Zmiany w krzyżowaniu

$$p_c = \frac{Nagroda(Krzyz)/N(Krzyz)}{Nagroda(Krzyz)/N(Krzyz) + Nagroda(Mut)/N(Mut)}$$

Nagroda(Krzyz) – liczba krzyżowań, które doprowadziły do poprawy

N(Krzyz) – liczba wszystkich krzyżowań

Mut – analogicznie dla mutacji

Parametry Algorytmu Ewolucyjnego

Przykład: Zmiany w krzyżowaniu

- Możliwe jest użycie więcej niż jednego rodzaju krzyżowania.
- Osobnik zawiera dodatkowy bit (dodatkowe bity), które wskazują na rodzaj krzyżowania, które powinno być użyte.
- Potomkowie dziedziczą odpowiednie bity od rodziców.
- Jeśli pewna operacja krzyżowania daje lepsze nowe osobniki, ten rodzaj krzyżowania rozprzestrzenia się w populacji.

Parametry Algorytmu Ewolucyjnego

Przykład: Zmiany w selekcji

Proporcjonalne szeregowanie – każdy osobnik ma przypisane własne prawdopodobieństwo wyboru, które jest proporcjonalne do jego numeru na liście rankingowej.

$$p(i) = \frac{2 - b + 2i(b-1)/(rozm\_pop - 1)}{rozm\_pop}$$

b – wartość oczekiwana liczby potomków pochodzących od najlepszego osobnika.

Zmieniając wartość ***b*** np. w zakresie ***[1, 2]***, można zmieniać nacisk selekcyjny.

Problemy z ograniczeniami

Problemy z ograniczeniami

$$G1(x) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

$$2x_1 + 2x_2 + x_{10} + x_{11} \leq 10$$

$$2x_1 + 2x_3 + x_{10} + x_{12} \leq 10$$

$$2x_2 + 2x_3 + x_{11} + x_{12} \leq 10$$

$$-8x_1 + x_{10} \leq 0$$

$$-8x_2 + x_{11} \leq 0$$

$$-8x_3 + x_{12} \leq 0$$

$$-2x_4 - x_5 + x_{10} \leq 0$$

$$-2x_6 - x_7 + x_{11} \leq 0$$

$$-2x_8 - x_9 + x_{12} \leq 0$$

Problemy z ograniczeniami

Większość rzeczywistych problemów nakłada ograniczenia na potencjalne rozwiązania – część z nich jest rozwiązaniami niedopuszczalnymi.

W algorytmie ewolucyjnym w populacji możemy otrzymywać zarówno rozwiązania dopuszczalne jak i niedopuszczalne.

Końcowe rozwiązanie musi być dopuszczalne, inaczej nie jest w ogóle rozwiązaniem.

Problemy z ograniczeniami

Podczas ewolucji, podczas poszukiwania rozwiązań dopuszczalnych przydatne jednak może być operowanie na rozwiązaniach niedopuszczalnych.

Problemy z ograniczeniami

W metodach ewolucyjnych funkcja oceny wartościuje poszczególne rozwiązania i jest głównym powiązaniem między problemem a algorytmem.

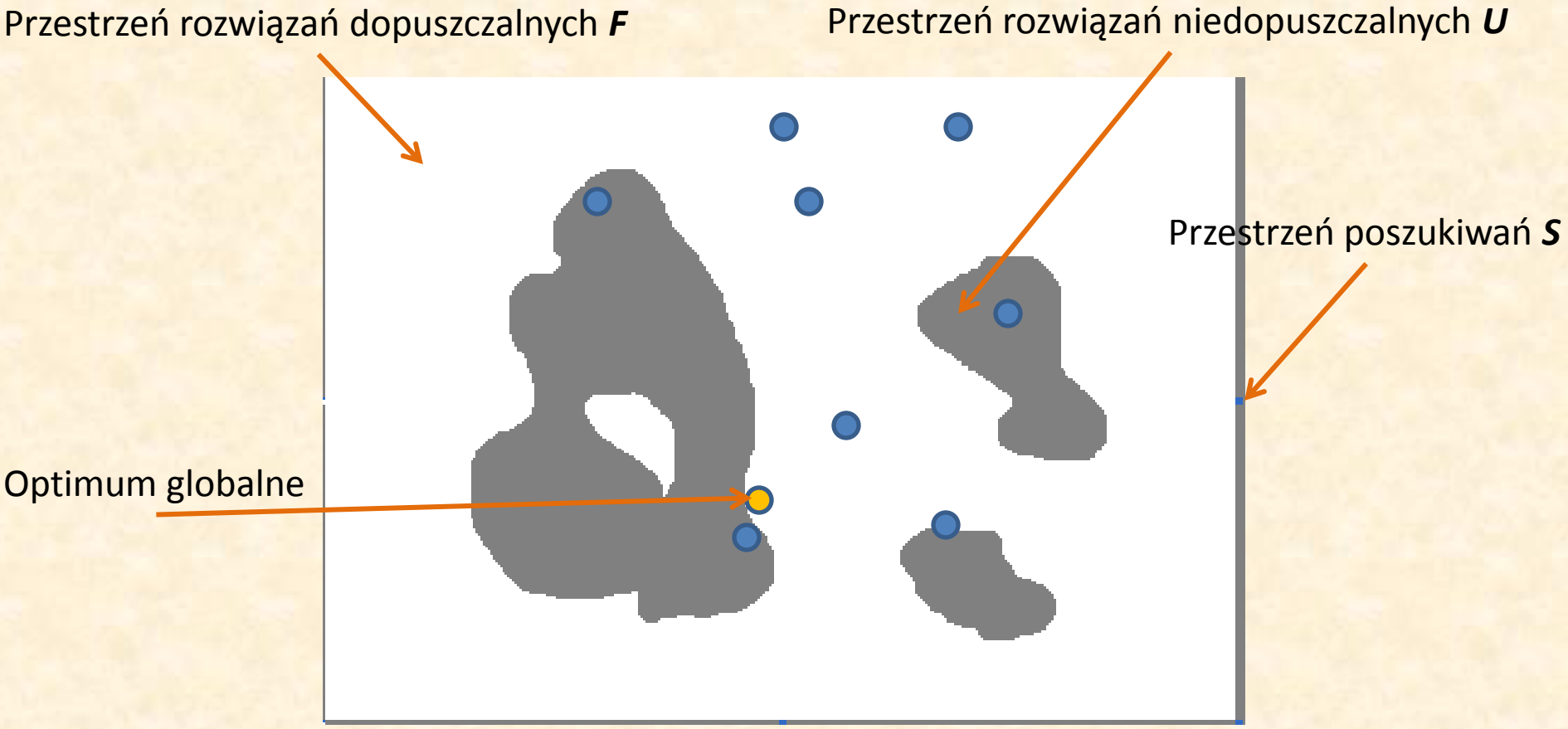
Ważą sprawą jest, by odpowiednio oceniać osobniki, również te niedopuszczalne.

Problemy z ograniczeniami

Trudno jest zaprojektować takie operatory różnicowania, które zawsze dają rozwiązania dopuszczalne a jednocześnie o dobrej jakości.

W niektórych problemach ograniczenia są tak mocno, że ciężko jest nawet wygenerować początkową, losową populację rozwiązań dopuszczalnych. Często chodzi o wygenerowanie jakiegokolwiek rozwiązania dopuszczalnego (ang. ***constraint satisfaction problems***)

Problemy z ograniczeniami



F – (ang. *feasible*) obszar dopuszczalny
U – (ang. *unfeasible*) obszar niedopuszczalny

Problemy z ograniczeniami

Najczęściej należy określić dwie funkcje oceny:

$\boldsymbol{eval_f}$ – funkcja oceny dla rozwiązań dopuszczalnych

$\boldsymbol{eval_u}$ – funkcja oceny dla rozwiązań niedopuszczalnych

Problemy z ograniczeniami

Trudności:

- Jak porównywać dwa osobniki dopuszczalne?

Problemy z ograniczeniami

Trudności:

- Jak porównywać dwa osobniki dopuszczalne?
- Jak porównywać dwa osobniki niedopuszczalne?

Problemy z ograniczeniami

Trudności:

- Jak porównywać dwa osobniki dopuszczalne?
- Jak porównywać dwa osobniki niedopuszczalne?
- W jak sposób są powiązane funkcje oceny $\boldsymbol{eval}_f$ oraz $\boldsymbol{eval}_u$? Czy zawsze jakikolwiek osobnik dopuszczalny jest lepszy od każdego osobnika niedopuszczalnego?

Problemy z ograniczeniami

Trudności:

- Czy osobniki niedopuszczalne powinny być od razu eliminowane z populacji?

Problemy z ograniczeniami

Trudności:

- Czy osobniki niedopuszczalne powinny być od razu eliminowane z populacji?
- Czy powinniśmy poprawiać osobniki niedopuszczalne przenosząc je do najbliższego punktu w przestrzeni dopuszczalnej? Jaki będzie koszt takiej operacji?

Problemy z ograniczeniami

Trudności:

- Jeśli poprawiamy osobniki niedopuszczalne, to poprawiony osobnik powinien zastępować niedopuszczalnego, czy tylko być użyty do oceny?
- Czy i jak karać osobniki niedopuszczalne?
- Czy startować od populacji tylko rozwiązań dopuszczalnych i stosować operatory gwarantujące dopuszczalność potomstwa?

Problemy z ograniczeniami

Trudności:

- Czy zmieniać topologię przestrzeni poszukiwań za pomocą dekoderów, które tłumaczą rozwiązania niedopuszczalne na dopuszczalne?
- Czy przetwarzać osobno osobniki i ograniczenia?
- Czy poszukiwać granicy między obszarem dopuszczalnym a niedopuszczalnym?

Problemy z ograniczeniami

Trudności:

- Czy traktować każde ograniczenie jako nowe kryterium, którego naruszenie trzeba minimalizować a cały problem z ograniczeniami traktować jako problem optymalizacji wielokryterialnej?

Problemy z ograniczeniami

Co można / trzeba wykonać

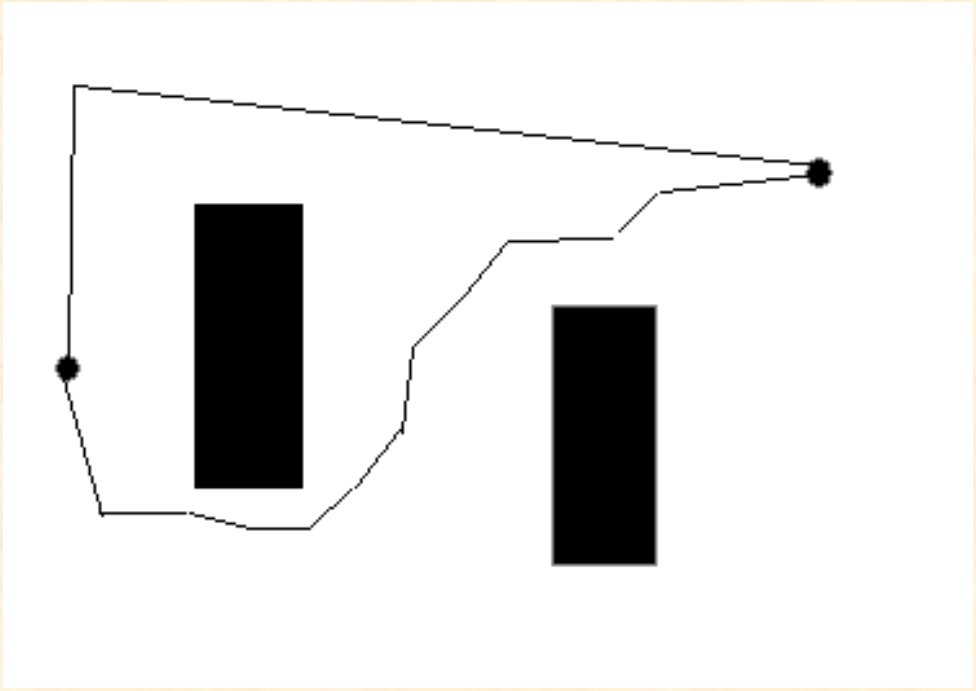
- Określenie funkcji $\boldsymbol{eval_f}$
- Określenie funkcji $\boldsymbol{eval_u}$
- Określenie zależności między $\boldsymbol{eval_f}$ a $\boldsymbol{eval_u}$
- Odrzucanie rozwiązań niedopuszczalnych
- Poprawianie rozwiązań niedopuszczalnych
- Zastępowanie osobników ich poprawionymi wersjami
- Nakładanie kar na osobniki niedopuszczalne
- Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania
- Stosowanie dekoderów
- Oddzielenie osobników i ograniczeń
- Badanie granicy między dopuszczalną a niedopuszczalną częścią przestrzeni poszukiwań
- Znajdowanie rozwiązań dopuszczalnych

Problemy z ograniczeniami

Określenie funkcji $\boldsymbol{eval}_f$

W większości przypadków określenie funkcji celu wynika z rozwiązywanego problemu. Czasami jednak nie jest ona oczywista.

Problemy z ograniczeniami
Określenie funkcji $\boldsymbol{eval_f}$



Która ścieżka jest lepsza? Krótsza czy gładsza?

Problemy z ograniczeniami

Określenie funkcji $eval_f$

Przykład

Problem SAT

$$F(\bar{x}) = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (x_2 \vee x_3)$$

p = [0, 0, 0]

q = [1, 0, 0]

F(p) = F(q) = 0

Bezpośrednia wartość funkcji F nie daje nam rozróżnienia między **p** i **q**.

Jednym z podejść jest zdefiniowanie $eval_f$ jako stosunku liczby iloczynów logicznych, które dają prawdę, do wszystkich iloczynów logicznych.

$eval_f$(**p**) = 0.666

$eval_f$(**q**) = 0.333

Problemy z ograniczeniami

Określenie funkcji $\boldsymbol{eval_f}$

Czasami nie jest konieczne określania $\boldsymbol{eval_f}$ jako odwzorowania na liczby rzeczywiste – wystarczająca może się okazać możliwość porównywania osobników. Można wtedy stosować selekcje turniejową oraz rankingową. Nie można wykorzystać selekcji proporcjonalnej, gdyż nie ma liczbowej oceny osobników.

Przykład:

W zadaniu szukania optymalnego kontrolera danego procesu, porównanie dwóch osobników może polegać na sprawdzeniu, który z nich pierwszy popełni błąd wykonując zadanie sterowania.

Problemy z ograniczeniami

Określenie funkcji $\boldsymbol{eval_u}$

Funkcja $\boldsymbol{eval_u}$ może powstać jako niezależna funkcja oceny bądź też poprzez modyfikację funkcji $\boldsymbol{eval_f}$:

$$eval_u(x) = eval_f(x) \pm Q(x)$$

Gdzie ***Q(x)*** oznacza

- karę związaną z przekroczeniem ograniczeń
- koszt naprawy danego osobnika

Problemy z ograniczeniami

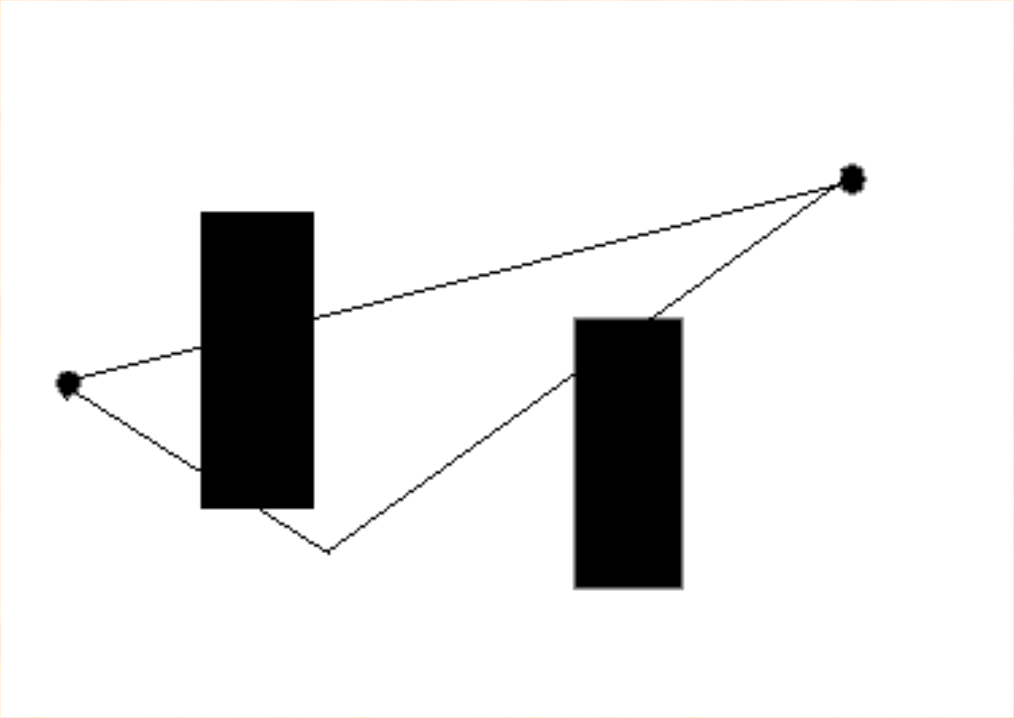
Określenie funkcji $\boldsymbol{eval}_u$

Osobnik może być karany za:

- Samo przekroczenie ograniczeń
- Stopień przekroczenia ograniczeń

Nie jest łatwo dobrać odpowiednio wielkość kary. Prowadzone są badania nad **somodostosowującymi się karami**, które są ustalane w drodze ewolucji.

Problemy z ograniczeniami
Określenie funkcji $\boldsymbol{eval}_u$



Które z rozwiązań niedopuszczalnych jest lepsze?

Określenie funkcji $eval_u$

Problem plecakowy

Plecak ma ograniczoną pojemność (maksymalną wagę W przedmiotów, jakie można do niego zapakować). Z danego zbioru przedmiotów (każdy o wadze w_i oraz wartości p_i) należy wybrać zestaw o maksymalnej wartości, który można zapakować do plecaka.

$$\max \sum_i p_i$$

$$\sum_i w_i \leq W$$

Określenie funkcji $eval_u$

Problem plecakowy

Załóżmy, że plecak ma pojemność 99.

Dwa rozwiązania niedopuszczalne o tej samej wartości ***V***:

Pierwsze: waga = 20 + 20 + 20 + 20 + 20 = 100

Drugie: waga = 20 + 20 + 20 + 39 + 6 = 105

Rozwiązanie drugie bardziej narusza ograniczenie, ale jeśli przedmiot o wadze 6 jest mało wartościowy, to jego usunięcia da nam rozwiązanie dopuszczalne lepsze niż jakiekolwiek ulepszone rozwiązanie otrzymane przez poprawę pierwszego.

Zatem branie pod uwagę jedynie samego stopnia naruszania ograniczeń może nie być dobrym pomysłem. Należy również uwzględnić łatwość poprawiania osobnika oraz jakość poprawionej wersji.

Problemy z ograniczeniami
Określenie zależności między $eval_f$ a $eval_u$

Jeśli istnieją dwie funkcje $eval_f$ oraz $eval_u$ należy określić globalną funkcję oceny

$$eval(p) = \begin{cases} q_1 \cdot eval_f(p) & dla \quad p \in F \\ q_2 \cdot eval_u(p) & dla \quad p \in U \end{cases}$$

Wagi q_1 oraz q_2 służą do wyskalowania znaczenia obu funkcji oceny. Wagi te mogą być dobierane dynamicznie w trakcie ewolucji, zwiększając presję na rozwiązania niedopuszczalne.

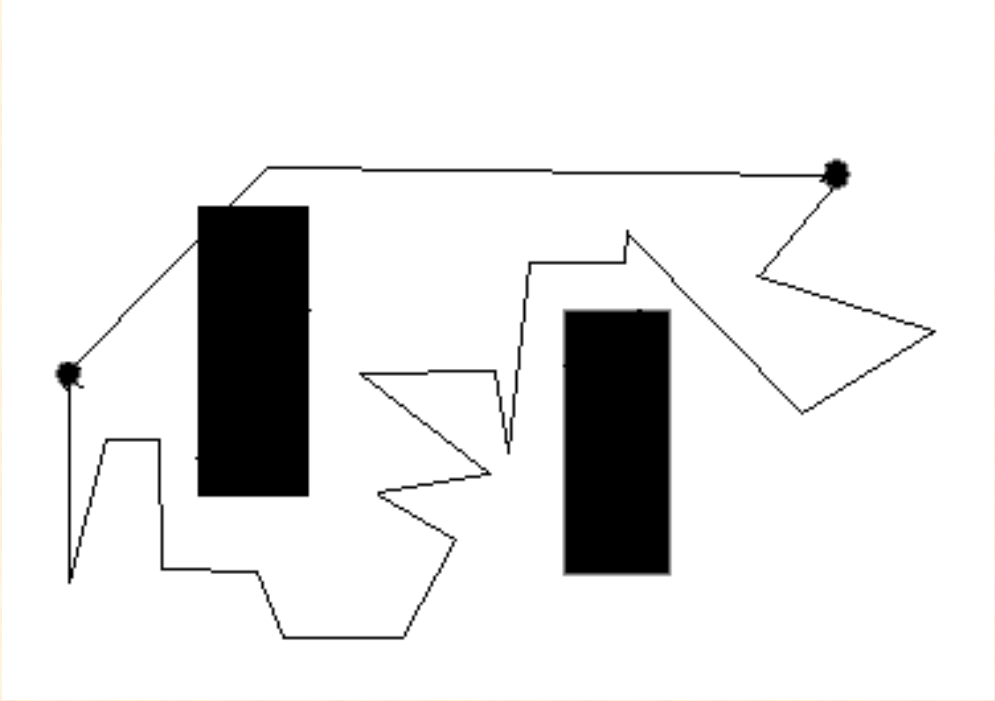
Problemy z ograniczeniami
Określenie zależności między $eval_f$ a $eval_u$

Można również podjąć odpowiednie zabiegi, by najgorszy osobnik dopuszczalny był zawsze lepszy niż jakikolwiek osobnik niedopuszczalny. Można to osiągnąć poprzez dodanie / odjęcie do/od funkcji oceny osobników niedopuszczalnych odpowiedniej wartości.

W przeciwnym wypadku istnieje możliwość, że osobniki niedopuszczalne zaczną dominować w populacji.

Z drugiej strony – osobnik niedopuszczalny może być korzystny jeśli znajduje się blisko optimum.

Problemy z ograniczeniami
Określenie zależności między $eval_f$ a $eval_u$



Które z rozwiązań jest lepsze? Krótsze niedopuszczalne (prawie optymalne), czy dopuszczalne ale zagmatwane?

Odrzucanie rozwiązań niedopuszczalnych

„Kara śmierci”

Może działać dość dobrze, jeśli część przestrzeni dopuszczalna stanowi istotną część całej przestrzeni poszukiwań oraz jest ona wypukła.

Poprawianie osobników niedopuszczalnych

- Popularne zwłaszcza w problemach optymalizacji kombinatorycznej (TSP, problem plecakowy, problem pokrycia zbiorami), gdzie operacje „naprawy” jest dość łatwo zdefiniować
- Osobnik oryginalny może być zastąpiony przez swoją poprawioną wersję bądź nie, może to nastąpić z pewnym prawdopodobieństwem.
- Jest to swoistego rodzaju połączenie ewolucji i uczenia (tzw. ***efekt Baldwina***) – ewolucja i uczenie oddziałują na siebie, wartość uzyskana w wyniku lokalnego poszukiwania (uczenia się) jest przekazywana potomkom

Poprawianie osobników niedopuszczalnych

Trudności:

- Czasami naprawianie osobników jest tak samo trudne jak pierwotny problem
- Dla każdego zadania należy zdefiniować procedury naprawiania

Poprawianie osobników niedopuszczalnych

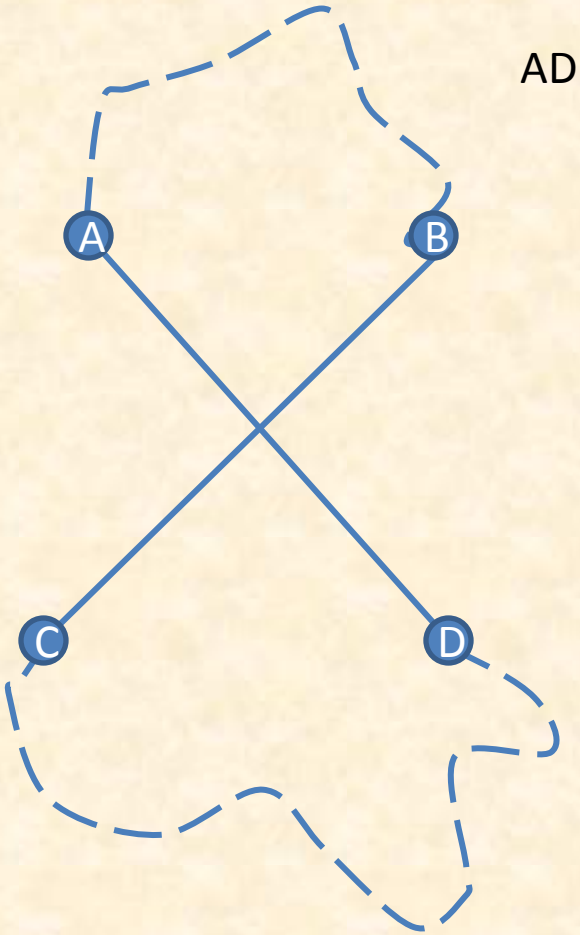
Zastępowanie osobników ich poprawionymi wersjami wiąże się z pojęciem ewolucji lamarkowskiej (ang. ***Lamarckian evolution***), w której zakłada się, że osobnik w czasie swojego życia doskonali się i ulepszenia przekazuje w postaci zmienionego kodu genetycznego.

Od strony biologicznej jest to teoria błędna, jednak może służyć do budowy algorytmów ewolucyjnych.

Algorytm ewolucyjny z przeszukiwaniem lokalnym

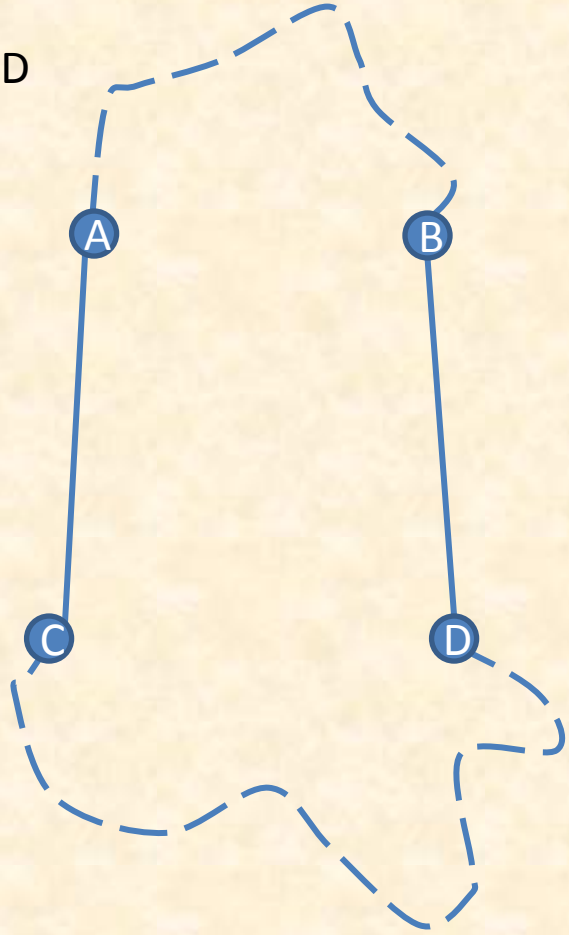
Przykład TSP

Algorytmy 2-optymalne wyszukują krzyżujące się krawędzie, jednak nie gwarantują globalnego optimum.



Zawsze

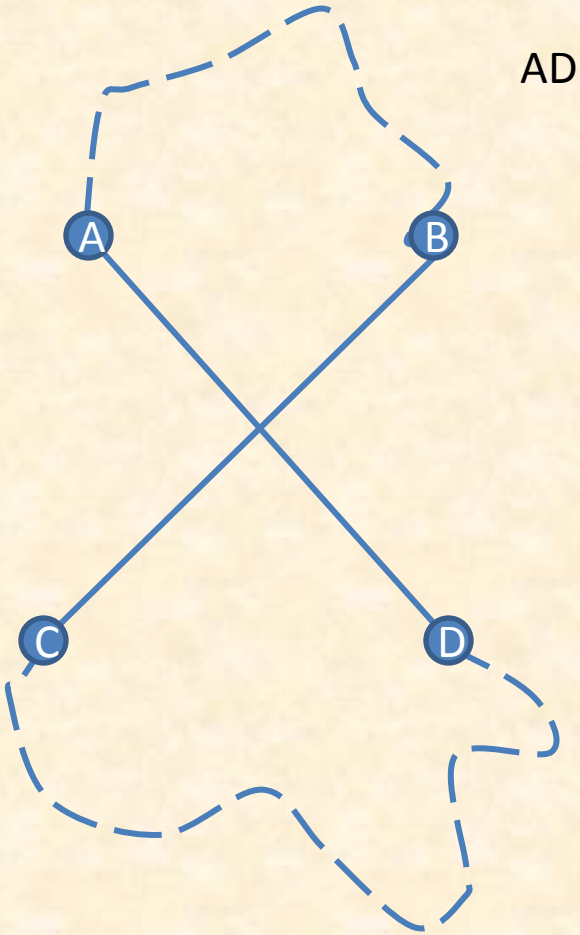
AD + BC > AC + BD



Algorytm ewolucyjny z przeszukiwaniem lokalnym

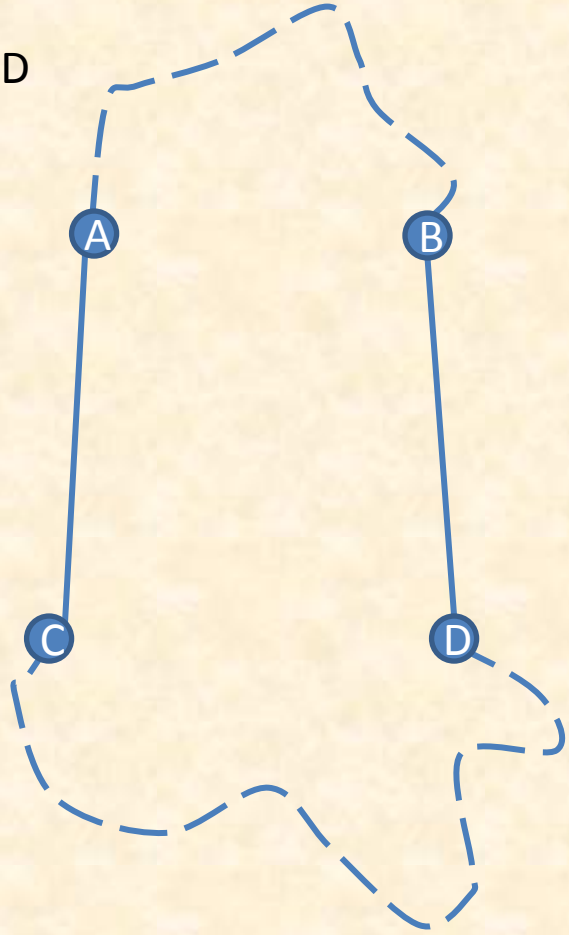
Przykład TSP

Algorytmy 2-optymalne mogą być użyte do poprawy osobników w algorytmie genetycznym.



Zawsze

AD + BC > AC + BD



Algorytm ewolucyjny z przeszukiwaniem lokalnym

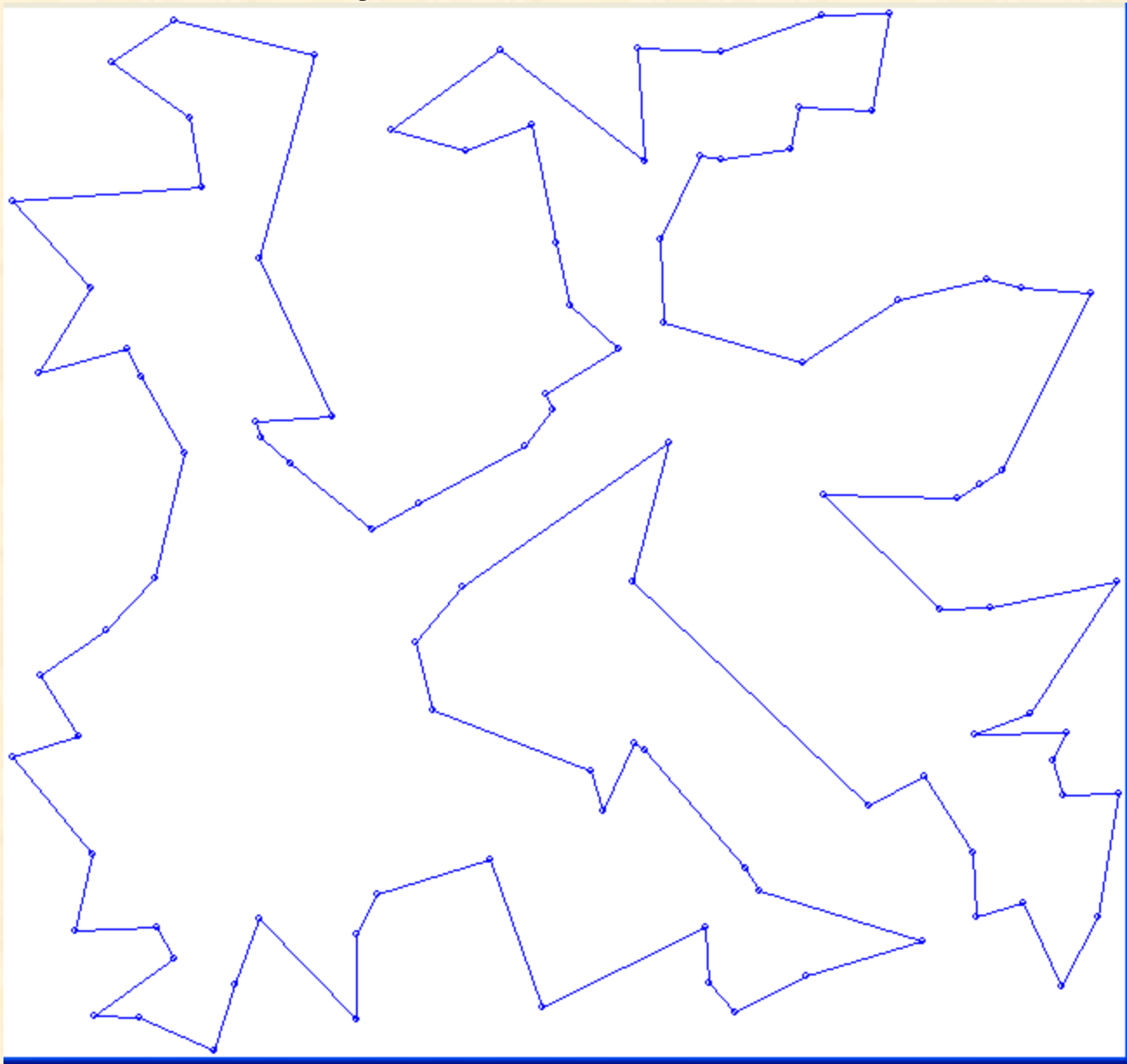
Przykład TSP

Uwaga: Tutaj osobniki kodujące trasy z krzyżującymi się krawędziami są osobnikami dopuszczalnymi.

Algorytm ewolucyjny z przeszukiwaniem lokalnym

Algorytm 2-opt

8,58611



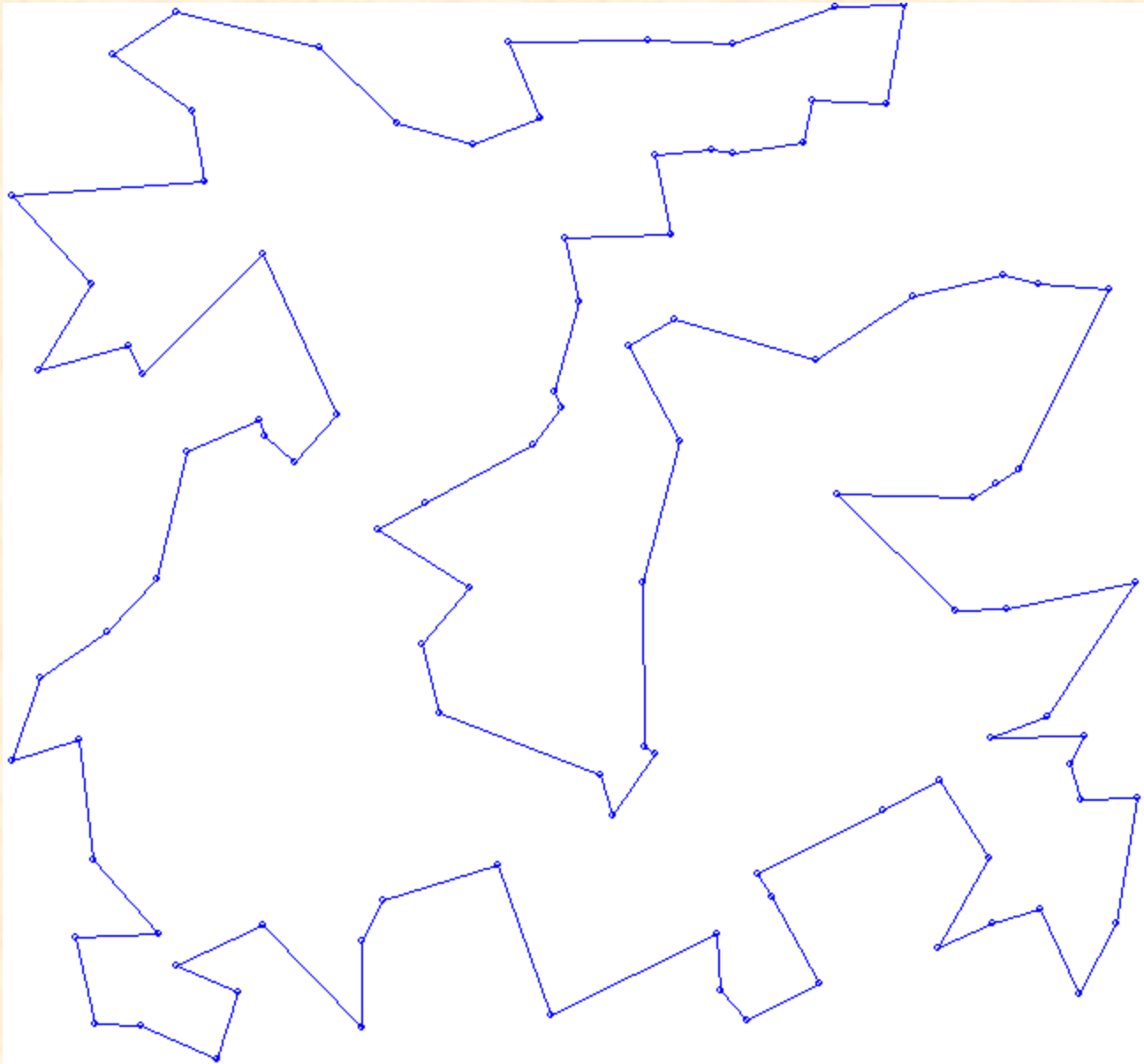
Liczba miast = 100

Algorytm ewolucyjny z przeszukiwaniem lokalnym

Przykład TSP

8,03553

Alg. Genetyczny
50 osobników
30000 iteracji



Liczba miast = 100

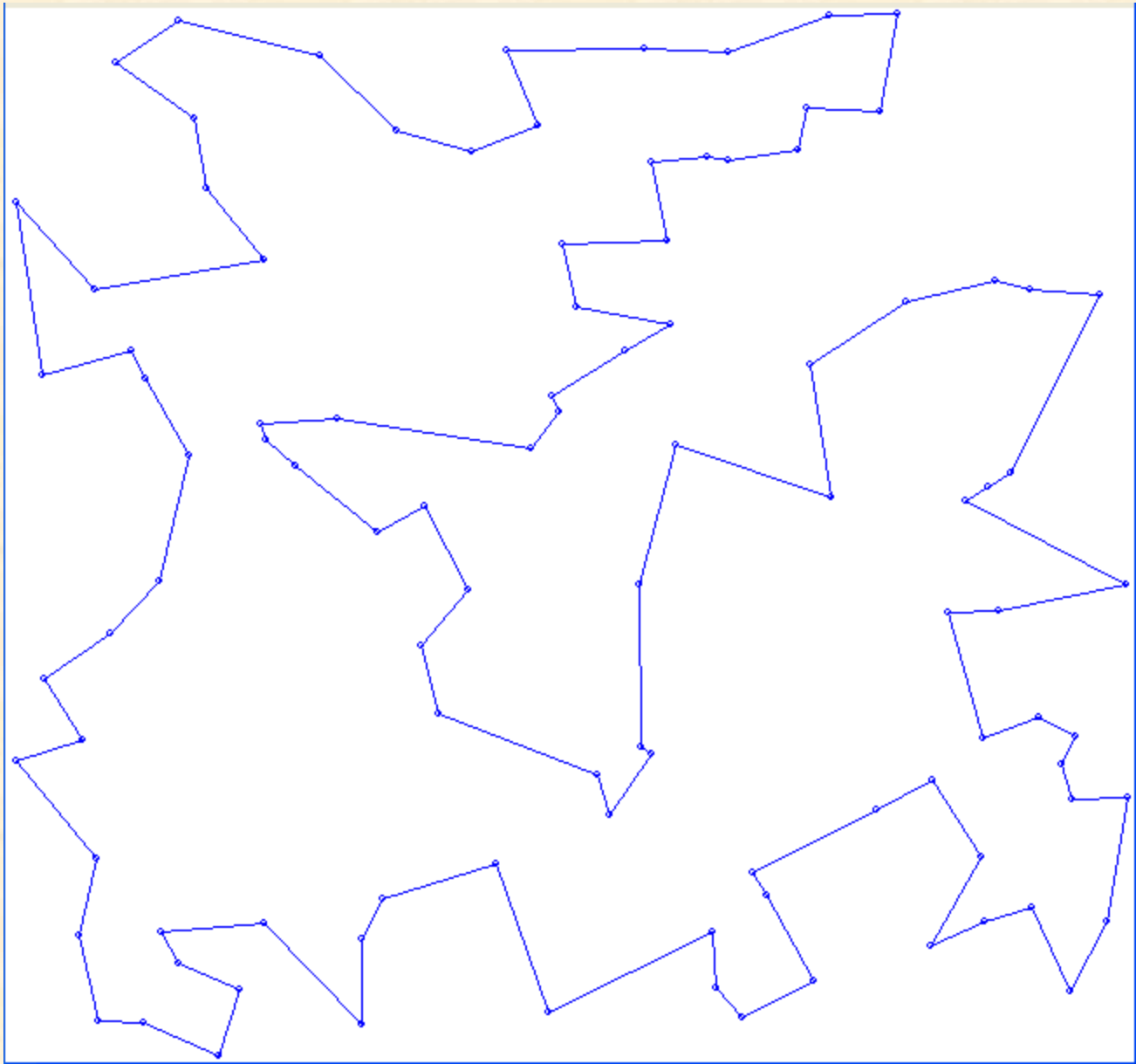
Algorytm ewolucyjny z przeszukiwaniem lokalnym

Przykład TSP

7.91715

Alg. Gen.
+
Opt-2

10 iteracji



Liczba miast = 100

Metody oparte na funkcjach kary

- Problem z ograniczeniami jest zastąpiony problemem bez ograniczeń, ale ze zmodyfikowaną funkcją celu
- Zmniejszana jest wartość rozwiązań niedopuszczalnych

$$eval(x) = \begin{cases} f(x) & dla \quad x \in F \\ f(x) + kara(x) & dla \quad x \in U \end{cases}$$

Metody oparte na funkcjach kary

- Kara jest równa 0 dla rozwiązań dopuszczalnych
- Dla rozwiązań niedopuszczalnych kara jest zazwyczaj oparta na odległości wyznaczającej, jak daleko rozwiązanie się znajduje od obszaru ***F***, lub ile kosztuje jego naprawienie, czyli przekształcenie w rozwiązanie z ***F***

$$eval(x) = \begin{cases} f(x) & dla \quad x \in F \\ f(x) + kara(x) & dla \quad x \in U \end{cases}$$

Metody oparte na funkcjach kary

- Wiele metod korzysta z funkcji f_j dla $1 \le j \le m$, gdzie m to liczba wszystkich ograniczeń, q to liczba ograniczeń nierównościowych

$$f_j(x) = \begin{cases} \max\{0, g_j(x)\} & dla \quad 1 \le j \le q \\ |h_j(x)| & dla \quad q+1 \le j \le m \end{cases}$$

Metody kar statycznych

- Dla każdego ograniczenia tworzona jest pewna liczba stopni jego naruszenia, które określają odpowiedni współczynnik kary $\boldsymbol{R_{ij}}$
- Wyższe stopnie wymagają wyższych współczynników kary
- Zaczynamy od losowej populacji
- Każdy osobnik jest oceniany za pomocą funkcji

$$eval(x) = f(x) + \sum_{j=1}^{m} R_{ij} \cdot f_j^2(x)$$

Metody kar statycznych

Cechy:

- Duża liczba parametrów
- Może dawać dobre wyniki jeśli współczynniki kary oraz przedziały są dostosowane do problemu

$$eval(x) = f(x) + \sum_{j=1}^{m} R_{ij} \cdot f_j^2(x)$$

Metody kar dynamicznych

Osobniki są oceniane za pomocą funkcji

$$eval(x) = f(x) + (C \cdot t)^a \sum_{j=1}^{m} f_j^b(x)$$

Gdzie ***C***, ***a*** oraz ***b*** są stałymi, np.

C = 0.5

a = b = 2

Czynnik ***(C * t)^a*** rośnie w miarę kolejnych iteracji, więc kary są coraz większe. Często może on rosnąć zbyt szybko, co jest wadą tej metody.

Metoda kar wyżarzanych

- Podziel ograniczenia na równania liniowe, nierówności liniowe, równania nieliniowe, nierówności nieliniowe
- Wybierz losowo jeden punkt początkowy – populacja początkowa składa się z kopii tego osobnika. Ten punkt spełnia ograniczenia liniowe.
- Ustaw temperaturę początkową na $t=t_0$
- Przeprowadź ewolucję oceniając osobniki za pomocą funkcji

$$eval(x,t) = f(x) + \frac{1}{2 \cdot t} \sum_{j=1}^{m} f_j^2(x)$$

- Jeśli $t < t_f$ to zatrzymaj, w przeciwnym razie
  - Zmniejsz temperaturę
  - Użyj najlepszego dostępnego rozwiązania jako punktu początkowego w następnej iteracji
  - Powtórz główny krok algorytmu

Metoda kar wyżarzanych (GENOCOP II)

- Algorytm za pomocą specjalnych operatorów utrzymuje dopuszczalność ze względu na ograniczenia liniowe
- Nacisk na rozwiązania niedopuszczalne się zwiększa w miarę jak zmniejsza się temperatura ***t***
- t_0 – temperatur początkowa, $t_0 = 1$
- t_f – temperatura zamarzania (ang. *freezing*), $t_f = 0.000001$
- Schemat schładzania: $t_{i+1} = 0.1 * t_i$

Metoda kar dostosowujących się

- Ocena osobników zgodnie ze wzorem

$$eval(x) = f(x) + \lambda(t) \sum_{j=1}^{m} f_j^2(x)$$

Metoda kar dostosowujących się

- Wykorzystywana jest informacja zwrotna z ewolucji do korygowania kar

$$\lambda(t+1) = \begin{cases} \frac{1}{\beta_1} \cdot \lambda(t) & if \quad b^i \in F \quad dla \quad t-k+1 \le i \le t \\ \beta_2 \cdot \lambda(t) & if \quad b^i \in U \quad dla \quad t-k+1 \le i \le t \\ \lambda(t) & w \quad przeciwnym \quad wypadku \end{cases}$$

b_i oznacza najlepszego osobnika w ***i-tej*** iteracji zgodniej z funkcją ***eval***

β_1 i $\beta_2 > 0$

$\beta_1 \neq \beta_2$

Metoda kar dostosowujących się

- Jeśli wszystkie najlepsze osobniki w ostatnich ***k*** pokoleniach były dopuszczalne to zmniejszamy wartość kary
- Jeśli wszystkie najlepsze osobniki w ostatnich ***k*** pokoleniach były niedopuszczalne to zwiększamy wartość kary
- Jeśli w ostatnich ***k*** pokoleniach były najlepsze osobniki dopuszczalne i niedopuszczalne to nie zmieniamy wartości kary

Metoda kar z segregacją

- Zbyt małe kary prowadzą do rozwiązań niedopuszczalnych
- Zbyt duże kary ograniczają przeszukiwanie

Metoda kar z segregacją

- Metoda z segregacją polega na zastosowaniu dwóch funkcji kar ze statycznymi współczynnikami kar p_1 oraz p_2
- Wartość p_1 ustawiana jest celowo zbyt wysoko a p_2 celowo zbyt nisko
- Każdy potomek jest oceniany za pomocą obu funkcji

$$f_i(x) = f(x) + p_i(x) \qquad i = 1,2$$

Metoda kar z segregacją

- Tworzone są dwie listy rankingowe, zwierające rodziców i potomków
- Rodzice następnego pokolenia są wybierania na przemian z pierwszej listy i z drugiej, kolejno zgodnie z kolejnością występowania
- W takim schemacie selekcji będą utrzymywane dwie podpopulacje – jedna z osobnikami leżącymi częściej w obszarze rozwiązań niedopuszczalnych, a druga z osobnikami częściej w obszarze dopuszczalnym

Zapewnianie przewagi osobników dopuszczalnych

- Każdy osobnik oceniany zgodnie ze wzorem (***r*** to stała)

$$eval(x) = f(x) + r \cdot \sum_{j=1}^{m} f_j(x) + \theta(t, x)$$

- Funkcja **θ(t,x)** zależy od iteracji i ma wpływ na ocenę osobników niedopuszczalnych
- Dla każdego osobnika dopuszczalnego ***x*** i każdego niedopuszczalnego ***y*** mamy

eval(x) < eval(y)

(dla minimalizacji), tzn. <u>każde dopuszczalne jest lepsze niż jakiekolwiek niedopuszczalne</u>

Zapewnianie przewagi osobników dopuszczalnych

Osobniki niedopuszczalne nie mogą być lepsze niż najgorsze dopuszczalne

$$\theta(t,x) = \begin{cases} 0 & if \quad x \in F \\ \max\{0, \max_{x \in F}\{f(x)\} - \min_{x \in U}\{f(x) + r\sum_{j=1}^{m} f_j(x)\}\} & if \quad x \in U \end{cases}$$

Zapewnianie przewagi osobników dopuszczalnych

Inna metoda:

$$eval(x) = \begin{cases} f(x) & if \quad x \in F \\ f_{\max} + \sum_{j=1}^{m} f_j(x) & if \quad x \in U \end{cases}$$

gdzie f_{max} jest wartością dla najgorszego aktualnego rozwiązania dopuszczalnego w populacji

- Wartość funkcji nie jest brana pod uwagę podczas oceny osobnika niedopuszczalnego
- Algorytm początkowo skupia się na znalezieniu rozwiązań dopuszczalnych a następnie na znajdowaniu lepszych rozwiązań dopuszczalnych.
- Rozwiązania dopuszczalne są zawsze oceniane lepiej niż niedopuszczalne

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

GENOCOP (**GE**netic algorithm for **N**umerical **O**ptimization of **CO**nstrained **P**roblems)

Specjalne operatory różnicowania przekształcają osobniki dopuszczalne w inne osobniki dopuszczalne.

Założenia: tylko liniowe ograniczenia oraz dopuszczalna populacja początkowa.

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

GENOCOP

- Za pomocą równań liniowych niektóre zmienne są eliminowane – są zastępowane liniowymi kombinacjami pozostałych.
- Jeśli jakaś zmienna x_i ma zostać zmieniona, to algorytm określa jej aktualną dziedzinę wynikającą z liniowych ograniczeń.

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

GENOCOP

- Ponieważ są tylko ograniczenia liniowe, przestrzeń dopuszczalna jest wypukła
- W wypukłych przestrzeniach krzyżowanie arytmetyczne osobników dopuszczalnych ***x*** i ***y*** daje zawsze rozwiązanie dopuszczalne

$$z = a*x + (1-a)*y \quad \text{dla} \quad 0 \leq a \leq 1$$

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

GENOCOP

$$G1(x) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

$$2x_1 + 2x_2 + x_{10} + x_{11} \le 10$$

$$2x_1 + 2x_3 + x_{10} + x_{12} \le 10$$

$$2x_2 + 2x_3 + x_{11} + x_{12} \le 10$$

$$-8x_1 + x_{10} \le 0$$

$$-8x_2 + x_{11} \le 0$$

$$-8x_3 + x_{12} \le 0$$

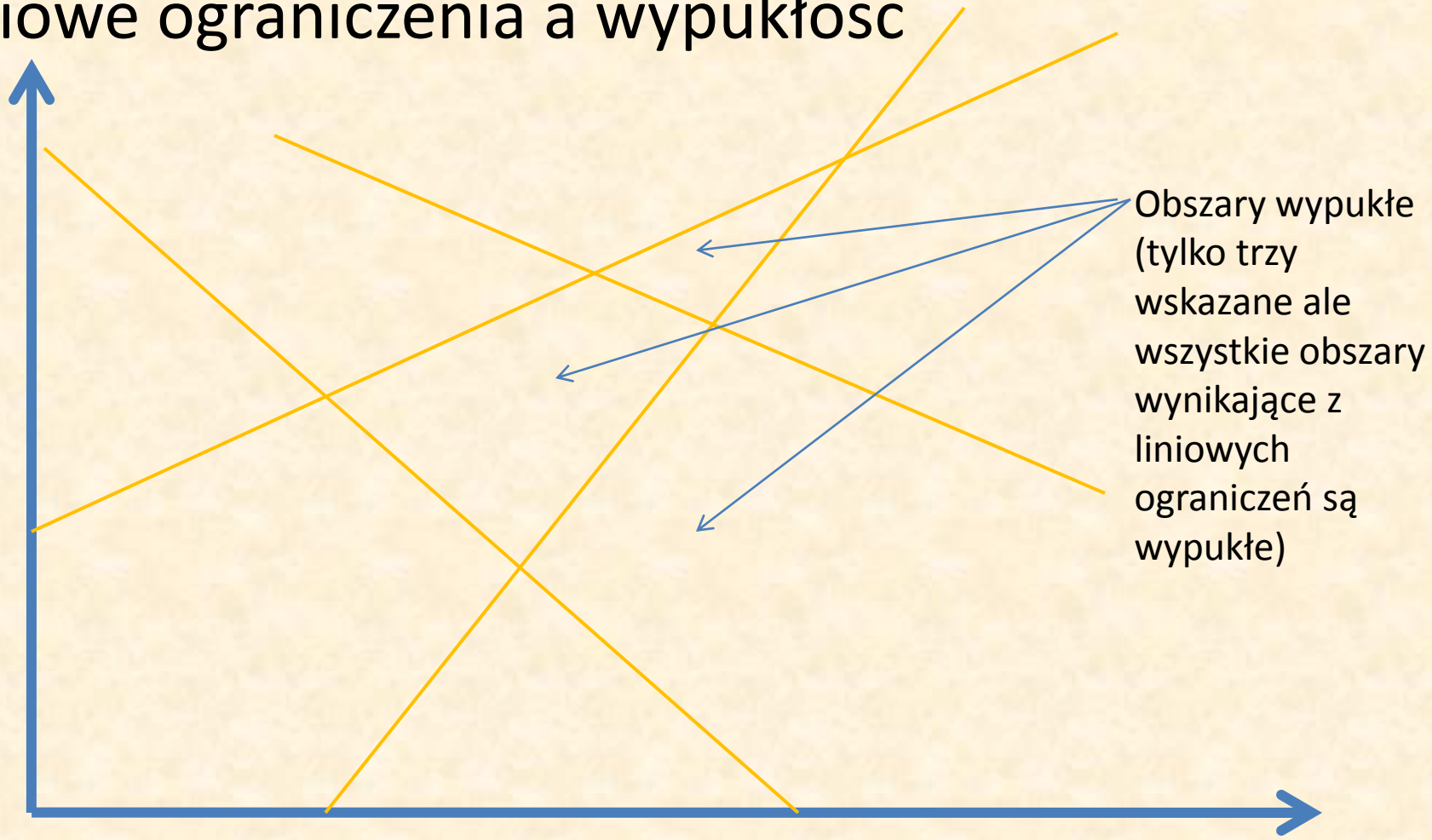
$$-2x_4 - x_5 + x_{10} \le 0$$

$$-2x_6 - x_7 + x_{11} \le 0$$

$$-2x_8 - x_9 + x_{12} \le 0$$

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

Liniowe ograniczenia a wypukłość



Obszary wypukłe (tylko trzy wskazane ale wszystkie obszary wynikające z liniowych ograniczeń są wypukłe)

Programowanie nieliniowe - NLP

Szukanie optimum funkcji z ograniczeniami

Przykład: znajdź maksimum funkcji:

$$G2(\mathbf{x}) = \left| \frac{\sum_{i=1}^{n} cos^4(x_i) - 2 * \prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i * x_i^2}} \right|$$

Przy ograniczeniach:

$$\prod_{i=1}^{n} x_i \geq 0.75$$

$$\sum_{i=1}^{n} x_i \leq 7.5 * n$$

dla $0 \leq x_i \leq 10, \; dla \; 1 \leq i \leq n$

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

Przykład

- Funkcja ta jest nieliniowa, jej maksimum globalne nie jest znane, lecz wiadomo, że leży w pobliżu początku układu współrzędnych
- Ograniczenie liniowe nie jest aktywne w pobliżu początku układu współrzędnych – będziemy je pomijać.

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

Pomysł: przeszukiwać granicę między obszarem dopuszczalnym a niedopuszczalnym.

Przeszukiwanie na granicach między tymi obszarami może być istotne w problemach z nieliniowymi ograniczeniami równościowymi lub z nieliniowymi ograniczeniami aktywnymi w docelowym optimum.

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

Granica jest określona za pomocą równania

$$\prod x_i = 0.75$$

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

Inicjowanie

Wybieramy losowo dodatnią liczbę x_i i jako wartość x_{i+1} przyjmujemy jej odwrotność. Ostatnia zmienna ma wartość 0.75 (dla ***n*** nieparzystego) lub jakąś wielokrotność 0.75 (dla ***n*** parzystego). Powoduje to, że punkt leży na powierzchni granicznej.

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

Krzyżowanie geometryczne

Dla ***a*** losowego z przedziału ***[0, 1]***

$$(x_i)(y_i) \rightarrow (x_i^a y_i^{1-a})$$

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

Mutacja

Wybieramy losowo dwie zmienne i mnożymy jedną z nich przez losowy współczynnik $q > 0$, drugą mnożymy przez $1/q$.

Ograniczamy q tak, by były spełnione ograniczenia dotyczące zmiennych.

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

Przykład 2

$$G3(x) = (\sqrt{n})^n \cdot \prod_{i=1}^{n} x_i$$

$$\sum_{i=1}^{n} x_i^2 = 1 \qquad 0 \le x_i \le 1$$

$$1 \le i \le n$$

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

Inicjowanie

Wybieramy losowo ***n*** zmiennych y_i, obliczamy

$$s = \sum_{i=1}^{n} y_i^2$$

i inicjujemy osobnika x_i zgodnie ze wzorem

$$x_i = y_i / s \qquad i \in [1, n]$$

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

Krzyżowanie sferyczne

Tworzymy potomka z_i z rodziców x_i oraz y_i dla a losowo wybranego z przedziału $[0, 1]$:

$$z_i = \sqrt{a \cdot x_i^2 + (1-a) \cdot y_i^2} \qquad i \in [1, n]$$

Utrzymywanie populacji dopuszczalnej za pomocą specjalnych sposobów reprezentacji i operatorów różnicowania

Mutacja

Wybieramy dwie współrzędne ***i ≠ j*** oraz losową liczbę ***p*** z przedziału ***[0, 1]***

$$x_i \rightarrow p \cdot x_i$$

$$x_j \rightarrow q \cdot x_j$$

$$q = \sqrt{\left(\frac{x_i}{x_j}\right)^2 (1 - p^2) + 1}$$

Stosowanie dekoderów

W tego rodzaju podejściach, struktura danych reprezentująca osobnika nie przedstawia rozwiązania bezpośrednio, ale podaje wskazówki jak zbudować rozwiązanie dopuszczalne.

Stosowanie dekoderów

Przykład – problem plecakowy

- Sortujemy przedmioty malejąco względem p_i/w_i (stosunek wartości do wagi)
- Osobniki kodujemy jako ciągi binarne o długości równej liczbie przedmiotów
- Rozwiązanie konstruujemy zgodnie z zasadą *„weź kolejny przedmiot z posortowanej listy, jeśli wskazuje na to bit 1 i jeśli to możliwe”*

Stosowanie dekoderów

Przykład – problem plecakowy

- Możemy stosować tradycyjne metody krzyżowania i mutacji – potomstwo zawsze koduje rozwiązanie dopuszczalne

Stosowanie dekoderów

W przypadku stosowania dekoderów ważne jest by:

- Dla każdego rozwiązania dopuszczalnego istniała możliwość jego zakodowania
- Dla każdego zakodowanego rozwiązania jego wersja rozkodowana była dopuszczalna
- Wszystkie rozwiązania z ***F*** były kodowane przez tę samą liczbę kodowań (choć korzystne może być, jeśli to rozwiązania bliskie optymalnemu są kodowane większą liczbą ciągów kodowych)
- Kodowanie było operacją szybką
- Kodowanie miało <u>własność lokalności</u>, tzn. małe zmiany w wersji zakodowanej powodowało małe zmiany we właściwym rozwiązaniu

Oddzielenie osobników i ograniczeń

Podejście to obejmuje metody optymalizacji wielokryterialnej.

Oryginalny problem z jedną funkcją celu i ograniczeniami staje się problemem optymalizacji z wieloma kryteriami bez ograniczeń.

Nowe kryteria wymagają minimalizacji stopnia naruszania ograniczeń pierwotnego problemu.

Podejście z pamięcią behawioralną

- Polega na obsłudze ograniczeń w określonej kolejności.
- W danym momencie optymalizowane jest tylko jedno ograniczenie

Podejście z pamięcią behawioralną

- Zacznij od losowej populacji
- ***j=1*** (***j*** to licznik ograniczeń)
- Przeprowadź ewolucję, przy czym osobniki oceniaj jedynie za pomocą funkcji

eval(x) = f_j(x)

Ewolucja kończy się, gdy zadany procent populacji stanie się dopuszczalny ze względu na przetwarzane ograniczenie (tzw. próg przełączenia)

- Ustaw ***j = j + 1***
- Bieżąca populacja staje się punktem wyjścia do następnej fazy ewolucji. Usuwane są osobniki, które nie spełniają ograniczeń

  1 .. j-1.

  Kryterium zatrzymania jest ponownie osiągnięcie progu przełączenia ze względu na ***j-te*** ograniczenie.
- Jeśli ***j = m*** (wszystkie ograniczenia zostały użyte) wykonaj optymalizację funkcji oceny, jednocześnie odrzucając osobniki niedopuszczalne.

Podejście z pamięcią behawioralną

- W końcowej fazie stosowana jest kara śmierci, ale nie na losowej populacji

GENOCOP III

- Algorytm utrzymuje dwie oddzielne populacje
- $\boldsymbol{P_s}$ – populacja punktów poszukiwania (ang. *search points*) spełniających ograniczenia liniowe.
- Specjalne operatory gwarantują, że potomkowie również spełniają ograniczenia liniowe.
- Oceniane przez porównanie z punktami z drugiej populacji i poprawianie za pomocą specjalnych operatorów.

GENOCOP III

- Algorytm utrzymuje dwie oddzielne populacje
- $\boldsymbol{P_r}$ – populacja punktów odniesienia(ang. *reference points*) będących w pełni dopuszczalnymi.
- Oceniane bezpośrednio przez funkcję oceny

KONIEC