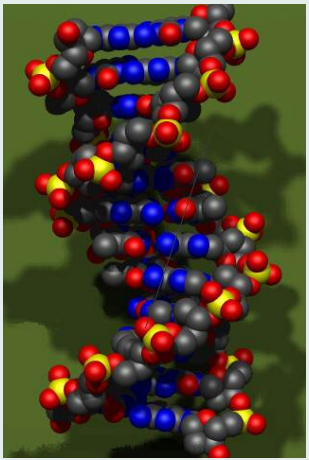
Wykład 5: Algorytmy genetyczne

Nguyen Hung Son
son@mimuw.edu.pl



1. Wprowadzenie

- Algorytmy genetyczne stanowią pewną gałąź działu "obliczeń ewolucyjnych", Jest to dynamicznie rozwijająca dziedzina w Sztucznej Inteligencji.
- AG są inspirowane przez Darwinowską teorią ewolucji. Rozwiązywanie problemu z użyciem AG polega na ewolucji!

1.1. Historia AG:

- początek: prace biologów Barricelli (1957), Fraser (1960). Symulowali oni procesy genetyczne przy pomocy komputerów.
- Idea "programowania ewolucujnego" została wprowadzona w 1960 przez I. Rechenberg w pracy "Evolution strategies" (*Evolutionsstrategie*).
- Holland w pracach nad systemami adaptacyjnymi(1962): pierwsze zastosowania algorytmów genetycznych w zagadnieniach sztucznej adaptacji. To zostało opisane w jego książce "Adaption in Natural and Artificial Systems" publikowanej w 1975.
- Bagley (1967) opracował algorytm genetyczny grający w prostą grę logiczną.
- W 1970 zastosowano AG do systemu rozpoznającego postacie ludzkie (przykład problemu nierozwiązywalnych tradycyjnymi metodami).
- W 1971 pojawiła się pierwsza praca badająca skuteczność AG do optymalizacji funkcji (Hollstein).
- J. Koza (1992): "Genetic programming"
- Do dziś, AG znalazły setki nowych zastosowań.

2. Podstawy biologiczne

2.1. Chromosomy

- Żywe organizmy składają się z komórek (cells).
- W każdej komórce znajduje się taki sam zbiór chromosomów.
- Chromosomy są łańcuchami DNA i pełnią rolę modelu całego organizmu.
- Chromosomy składają się z genów (bloków DNA). Każdy gen koduje poszczególne białko (protein) (ściślej mówiąc: każdy gen odpowiada za pewną cechę organizmu, np. kolor oczu). Możliwe ustawienia dla niego nazywamy *allelami*.
- Kompletny zbiór materiału genetycznego (wszystkich chromosomów) jest nazwany "genome".
- Zbiór genów w genome jest zwany "genotyp". Genotyp stanowi podstawą do formułowania fenotypu organizmu, jego fizycznych i mentalnych charakterystyk, np. kolor oczu czy inteligencja, ...

2.2. Reprodukcja

- "Rekombinacja" (crossover) jest pierwszym procesem podczas reprodukcji. Geny rodziców tworzą nowe geny potomka.
- Nowo tworzone geny potomka mogą ulegać "mutacji". Mutacja oznacza, że elementy DNA są lekko zmodyfikowane (głównie przez błędy popełnione podczas ich kopiowania od rodziców)
- Stopień dostosowania organizmu do środowiska (fitness) jest mierzony poprzez sukces dokonany przez ten organizm w swoim życiu

3. Podstawy działania AG

- Jest wyraźna różnica w AG między fenotypem (dorosły osobnik, rozwiązania) a genotypem (tzn. chromosom, reprezentacja);
- Skoro AG działa wyłącznie na genotypach, jakiś proces dekodowania genotypu na fenotyp jest potrzebny (uogólnione rośnięcie);
- Chromosomami są ciągi znaków (np. "0" i "1");
- Dorosłym osobnikiem może być dowolna, którą można kodować (dekodować) za pomocą chromosomów. Przykładem fenotypów są wektory parametrów lub list wyborów

Genetyka	**Algorytmy genetyczne**
chromosom	ciąg kodowy
gen	cecha, znak (w ciągu kodowym)
allele (odmiany genu)	warianty cechy
genotyp	struktura (zbiór ciągów kodowych)
fenotyp	zbiór parametrów, rozwiązanie, punkt

3.1. Zarys podstawowych AG

1. **Start:** Utwórz losową populację zawierającą n chromosomów;
2. **Fitness:** Oceń fitness $f(x)$ dla każdego chromosomu x w populacji;
3. **Nowa populacja:** Utwórz nową populację przez powtórzenie następujących kroków dopóki nowa populacja będzie pełna:
4. **Selekcja:** Wybierz dwóch chromosomów z populacji w sposób losowy ale zgodny z fitness chromosomów;
5. **Krzyżowanie:** Stwórz 2 nowe chromosomy z wybranych chromosomów (rodzicielskich) przez krzyżowanie z p-wem p_c (gdy nie zachodzi krzyżowanie, potomkami są po prostu kopie rodzicielskich chromosomów)
6. **Mutacja:** Geny w chromosomie ulegną mutacji z p-wem p_m;
7. **Akceptacja:** Umieść nowe chromosomy w nowej populacji;
8. **Zamiana:** Użyj nowej populacji do dalszych działań algorytmu;
9. **Test:** Jeśli warunek STOPu zachodzi, zatrzymaj algorytm i zwracaj najlepsze rozwiązanie;
10. **Powrót:** Idź do kroku 2

3.2. Przykład

- **Problem:** Znaleźć liczbę w zbiorze $\{0, 1, ..., 255\}$, której zapis binarny zawiera największą liczbę sekwencji "01".
- Znane rozwiązanie: 85 (01010101)
- Ustawienie dla AG:

  Chromosomy: 8 bitów (bajt).

  Dekodowanie: konwersja z binarnego zapisu do dziesiętnego.

  Fitness: Liczba sekwencji "01".

  Populacja: 10 osobników.

  Selekcja: Wybierz 5 najlepszych osobników do kojarzenia.

  Klonowanie: Każdy z nich jest kopiowany.

  Rekombinacja: (kojarzenia) Obcinanie 2 ciągów w losowej pozycji i wymiana ogonów.

  Mutacja: nie.

Inicjalizacja:

#	Chromosomy	Rozwiązanie	Fitness
a	10010111	151	2
b	01010100	84	3
c	11101010	234	2
d	00101001	41	3
e	11011011	219	2
f	10111110	190	1
g	01001110	78	2
h	10010010	146	2
i	00100011	35	2
j	11110100	244	1
Średnia			2.0

Selekcja:

#	Chromosomy	Rozwiązanie	Fitness
b	01010100	84	3
d	00101001	41	3
a	10010111	151	2
c	11101010	234	2
e	11011011	219	2
g	01001110	——	2
h	10010010	——	2
i	00100011	——	2
f	10111110	——	1
j	11110100	——	1

Nowa populacja po klonowaniu:

#	Chromosomy	Rozwiązanie	Fitness
b	01010100	84	3
d	00101001	41	3
a	10010111	151	2
c	11101010	234	2
e	11011011	219	2
Średnia			2.4

Kojarzenie:

#	Chromosomy	Rozwiązanie	Fitness
b	010 10100	84	3
a	100 10111	151	2
b+a	01010111	87	3
d	001010 01	41	3
b	010101 00	84	3
d+b	00101000	40	2
a	10 010111	151	2
c	11 101010	234	2
a+c	10101010	170	3
b	0101010 0	84	3
e	1101101 1	219	2
b+e	01010101	85	4
e	1101 1011	219	2
d	0010 1001	41	3
e+d	11011001	217	2

Nowa kompletna populacja po sortowaniu:

#	Chromosomy	Rozwiązanie	Fitness
b+e	01010101	85	4
b	01010100	84	3
b+a	01010111	87	3
d	00101001	41	3
a+c	10101010	170	3
a	10010111	151	2
c	11101010	234	2
e	11011011	219	2
d+b	00101000	40	2
e+d	11011001	217	2
Średnia			2.6

3.3. Kodowanie rozwiązań w AG

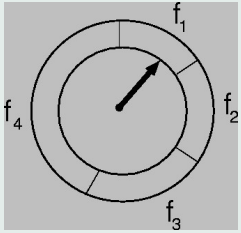
- Parametry całkowite:
- Parametry rzeczywiste:
- Wektor parametrów:

3.4. Selekcja

- Zasada: osobniki lepiej przystosowane do środowiska (większy fitness) ma większą szansę przeżyć (naturalną) selekcję;
- Selekcja proporcjonalna do fitness:
  - Niech $f_i > 0$: fitness i-tego osobnika, $\overline{f} = \frac{1}{N}\sum_j f_j$: średnie przystosowanie populacji, wówczas osobnik zostanie wybrany przez selekcję z p-wem:

$$p_i = \frac{f_i}{\sum_j f_j} = \frac{f_i}{N\overline{f}}$$

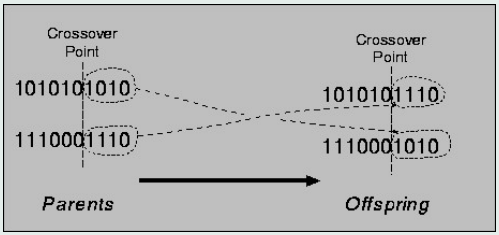
  - Ta metoda może być implementowana przez RULETKI, na przykład:



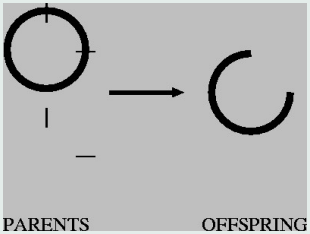
  - W praktyce, implementujemy przez dystrybuantę:
- Selekcja rankingowa, turniejowa, elitarna,...

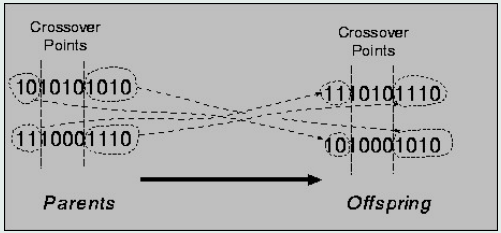
3.5. Krzyżowanie

- Krzyżowanie jedno-punktowe;

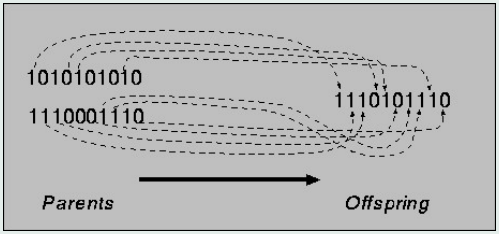


- Krzyżowanie dwu-punktowe;





- Krzyżowanie wielo-punktowe;



4. Podstawowe Twierdzenia w GA

4.1. Schematy

- Schematem jest dowolne słowo nad alfabetem $\{0, 1, *\}$. Przykład:

$$010 * 1, *110*, * * * * *, 10101, ...$$

- Znak * jest symbolem uniwersalnym oznaczającym dowolny znak. Schemat jest wzorcem opisującym wiele ciągów bitowych. E.g. *10*1 reprezentuje 01001, 01011, 11001, 11011
- $o(H)$ – rząd schematu H = liczba pozycji ustalonych (nie "*") w schemacie H.
- $\delta(H)$ – rozpiętość schematu H = Odległość między skrajnymi pozycjami ustalonymi (nie-gwiazdkowymi)w schemacie H

4.2. Przykłady schematów

alfabet $\Sigma = \{0, 1\}$

długość ciągu kodowego $l = 6$;

schemat	$H_1 = 110 * 11$	$H_2 = *0 * *1*$	$H_3 = 1 * * * **$
rząd	$o(H_1) = 5$	$o(H_2) = 2$	$o(H_3) = 1$
rozpietość	$\delta(H_1) = 5$	$\delta(H_2) = 3$	$\delta(H_3) = 0$

2^l ciągów kodowych;

3^l możliwych schematów;

$2^{l-o(H)}$ osobników (chromosomów) representujących schemat H;

2^l schematów reprezentowanych przez jednego osobnika;

4.3. Twierdzenie o schematach

Przypomnijmy podstawową wersję algorytmu genetycznego:

```
ALGORYTM_GENETYCZNY() //zwykly
{
    int t = 0;
    P(t) = populacja_poczatkowa;
    Ocen_chromosomy(P(t));
    while (! warunek_konca())
    {
        t = t+1;
        C(t) = selekcja (P(t-1));
        CR(t) = przetworzenie (C(t), P_c, P_m);
        P(t) = CR(T);
        Ocen_chromosomy(P(t));
    }
}
```

Używamy następujących oznaczeń dla zbadanego schematu H:

$m(H,t)$	: oczekiwana obecność schematu H w chwili t
$m(H,t+1)$	: oczekiwana obecność schematu H w chwili $t+1$
$f(H)$	: średnie przystosowanie reprezentantów schematu H w chwili t
$\bar{f} = \frac{\sum f_i}{N}$	: średnie przystosowanie całej populacji
p_c	: prawdopodobieństwo krzyżowania
$\delta(H)$	: rozpiętość schematu H
l :	: długość chromosomów
$o(H)$	: rząd schematu H
p_m	: prawdopodobieństwo mutacji

wówczas mamy:

Twierdzenie *(o schematach)*

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{\bar{f}}\left[1 - p_c\frac{\delta(H)}{l-1} - o(H)p_m\right]$$

4.4. Wniosek z twierdzenia o schematach

- dla schematów niskiego rzędu i o małej rozpiętości destruktywne efekty krzyżowania i mutacji można pominąć, a wtedy:

$$m(H, t+1) = m(H, t)\frac{f(H)}{\bar{f}(t)} = m(H, t)(1+\varepsilon)$$

przy założeniu, że $f(H, t) = \bar{f}(1+\varepsilon)$, czyli schemat H ma dopasowanie o ε większe niż średnia w populacji.

- zakładając, że ε nie zmienia się w czasie, wówczas długoterminowy efekt przetwarzania schematu H będzie następujący:

$$m(H, t+s) = m(H, t)(1+\varepsilon)^t$$

czyli oceniane "powyżej średniej" schematy "niskiego rzędu" i o "małej rozpiętości" uzyskują **wykładniczo rosnącą** liczbę reprezentantów w kolejnych pokoleniach

4.5. Hipoteza cegiełek

(ang. Building Block Hypothesis)

algorytm genetyczny poszukuje lepszych rozwiązań przez zestawianie wysokiej jakości schematów niskiego rzędu i o małej rozpiętości (tzw. cegiełek)

- oceniając poszczególne ciągi kodowe GA pośrednio ocenia jakość repre- zentowanych przez nie schematów;
- krzyżowanie pozwala na zestawianie ze sobą wysokiej jakości schematów niskiego rzędu, budując tak samo dobre lub lepsze schematy wyższego rzędu;
- jednocześnie przetwarzanych jest od 2^l do $N\dot{2}^l$ schematów, ale część z nich jest niszczona
- Holland oszacował, że "użytecznie" przetwarzanych jest tylko ok. N^3 schematów (ukryta równoległość, ang. implicit parallelism)

```
ALGORYTM_GENETYCZNY() //modyfikacja Hollanda
{
    int t = 0;
    P(t) = populacja_poczatkowa;
    Ocen_chromosomy(P(t));
    while (! warunek_konca())
    {
        t = t+1;
        while(kompletna(P(t))
        {
            chr1 =  wybor(P(t-1));
            if(RANDOM(0,1) < P_c) {
                chr2 = losowanie(P(t));
                    // niezaleznie od fitness
                l = wybor_punktu(1,l-1);
                krzyzowanie(chr1,chr2);
                    // stworzone sa p_chr1 i p_chr2
                dolacz(p_chr1,P(t));
                dolacz(p_chr2,P(t));
            }else
                dolacz(chr1,P(t));
        }
        Ocen_chromosomy(P(t));
    }
}
```

Twierdzenie *(o schematach w wersji Hollanda)*

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{\bar{f}}\left[1 - p_c\frac{\delta(H)}{l-1}\left(1 - \frac{m(H,t)}{l}\right)\right]$$

```
ALGORYTM_GENETYCZNY() //modyfikacja D.Schaffera
{
    int t = 0;
    P(t) = populacja_poczatkowa;
    Ocen_chromosomy(P(t));
    while (! warunek_konca())
    {
        t = t+1;
        while(kompletna(P(t))
        {
            chr1 =  wybor(P(t-1));
            if(RANDOM(0,1) < P_c) {
                chr2 = wybor(P(t));
                    // tu jest roznica
                l = wybor_punktu(1,l-1);
                krzyzowanie(chr1,chr2);
                    // stworzone sa p_chr1 i p_chr2
                dolacz(p_chr1,P(t));
                dolacz(p_chr2,P(t));
            }else
                dolacz(chr1,P(t));
        }
        Ocen_chromosomy(P(t));
    }
}
```

Twierdzenie *(o schematach w wersji D.Schaffera)*

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{\bar{f}} \cdot \cdot \left[1 - p_c \frac{\delta(H)}{l-1}\left(1 - \frac{f(H)}{\bar{f}} \cdot \frac{m(H,t)}{l}\right)\right]$$

Twierdzenie *(Banacha o punkcie stałym) Niech $\langle S, \delta \rangle$ będzie przestrzenią metryczną zupełną i niech $f : S \to S$ będzie odwzorowaniem zwężającym. Wtedy f ma jednoznaczny punkt stały $x^* \in S$ taki, że*

$$\forall_{x_0 \in S} \lim_{i \to \infty} f^i(x_0) = x^*$$

W przypadku algorytmów genetycznych

S : przestrzeń wszystkich możliwych populacji.

$\delta : S \to S$: $\delta(P_1, P_2) = \begin{cases} 0 & \text{if } P_1 = P_2, \\ |M - f(P_1)| + |M - f(P_2)| & \text{wpp.} \end{cases}$

$f : S \to S$:

```
ALGORYTM_GENETYCZNY() z odw. zwężającym {
   int t = 0;
   P(t) = populacja_poczatkowa;
   Ocen_chromosomy(P(t));
   while (! warunek_konca())
   {
       t = t+1;
       C(t) = selekcja (P(t-1));
       CR(t) = przetworzenie (C(t), P_c, P_m);
       P(t) = CR(T);
       Ocen_chromosomy(P(t));
       if (f(P(t-1)) >= f(P(t)))
           t = t-1;
   }
}
```

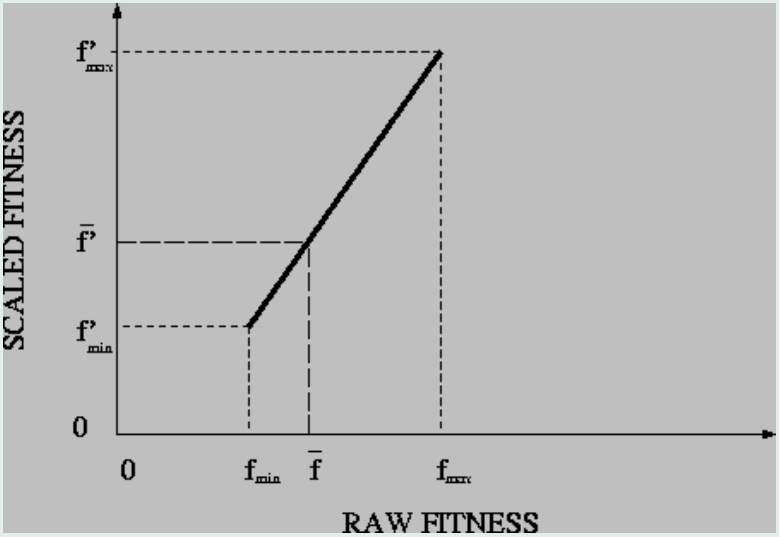
5. AG w optymalizacji

5.1. Przekształcenie funkcji celu na funkcję fitness

W wielu problemach optymalizacji typu:

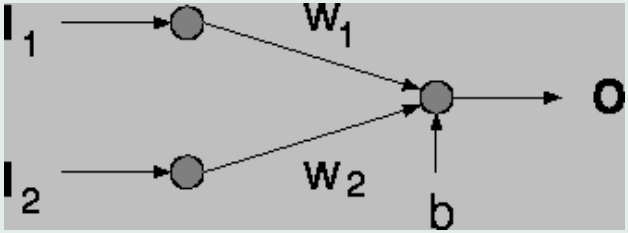
"znaleźć $s^* \in \mathcal{S}$ taki, że $Q(s^*) = \max_{s \in \mathcal{S}} Q(s)$

- Jeśli $\forall_s Q(S) > 0$, możemy definiować $f_{raw}(s) = Q(s)$
- Możemy (1) przesuwać, (2) odwrócić (3) skalować zakres funkcji f_{raw}



5.2. AG w optymalizacji bezwarunkowej

PRZYKŁAD 1: Znaleźć wagi dla sztucznego neuronu



$$O(I_1, I_2) = sgn(w_1 I_1 + w_2 I_2 + b)$$

I_1	I_2	I_1 OR I_2
0	0	0
0	1	1
1	0	1
1	1	1

$$U(w_1, w_2, b) = \Sigma |T_i - O_i|$$

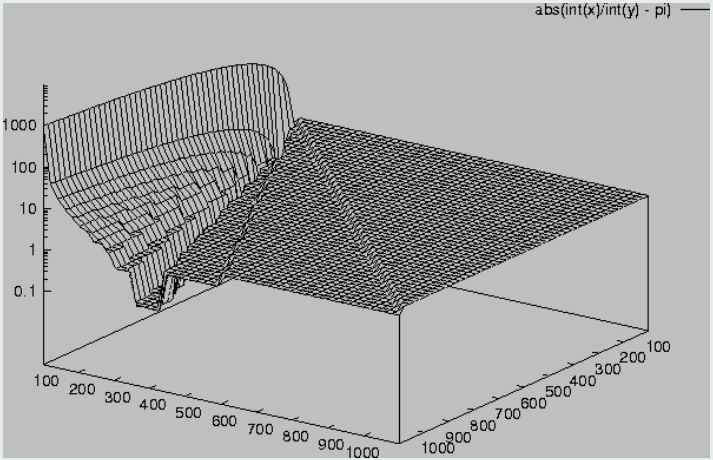
Parametry dla AG:

- 1-punktowe krzyżowanie z $p_c = 0.7$
- bez mutacji
- Selekcja turniejowa (N=2)
- Chromosomy = ciągi $(b_1 b_2 b_3)$ gdzie $b_i \in \{-1, 0, 1\}$
- Fitness $= 4 - U(w1, w_2, b)$
- Liczba indywiduów = 4
- Liczba generacji = 3

Kolejny Przykład: Znaleźć najlepszą wymierną aproksymację liczby π za pomocą liczb naturalnych od 0 do 1023. Innymi słowami, minimalizować funkcję

$$U(x_1, x_2) = \left| \frac{x_1}{x_2} - \pi \right|$$

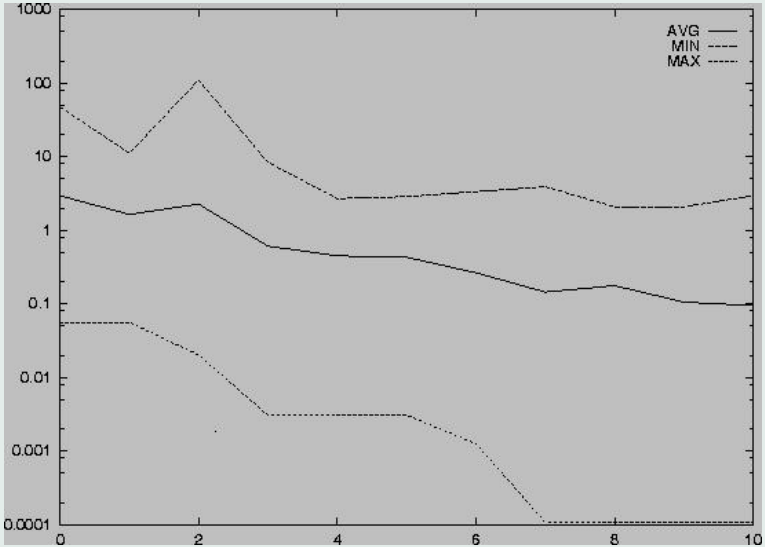
gdzie $x_1, x_2 \in \{0, ..., 1023\}$. Wykres tej funkcji wygląda następująco



Wiadomo, że najlepszym ułamkiem jest $355/113$ ($U < 0.00001$)
$22/7$ też jest dobry.

Parametry dla AG:

- 1-punktowe krzyżowanie z $p_c = 0.7$
- mutacja $p_m = 0.01$
- Selekcja turniejowa
- Chromosomy = ciągi 20-bitowe
- Fitness = $1000 - U(x_1, x_2)$
- Liczba indywiduów = 100
- Liczba generacji = 50



Najlepsze znalezione liczby po 50 generacji to: $x_1 = 776, x_2 = 247$, tzn. U=0.000108.

5.3. AG w optymalizacji warunkowej

Minimalizować(maks.) funkcję $U(x)$ z uwzględnieniem warunków:

1. $g_i(x) \geq 0$;
2. $h_j(x) = 0$;

Najczęściej stosuje się metodę funkcji kar do transformacji problemu:

- Zamiast minimalizacji funkcji $U(x)$, minimalizujemy funkcję

$$U(x) + r \sum_i \Phi[g_i(x)]$$

gdzie r jest wsp. kary, a Φ jest funkcją kary, np.

$$\Phi[g_i(x)] = \begin{cases} g_i^2(x) & \text{jeśli } g_i(x) < 0 \\ 0 & \text{w p.p.} \end{cases}$$

- Podobnie możemy postępować z równościami
- Zalety: prostota
- Wady: Nie które warunki mogą być naruszone poprzez "dobre rozwiązania w pobliżu granicy przestrzeni przeszukiwań.

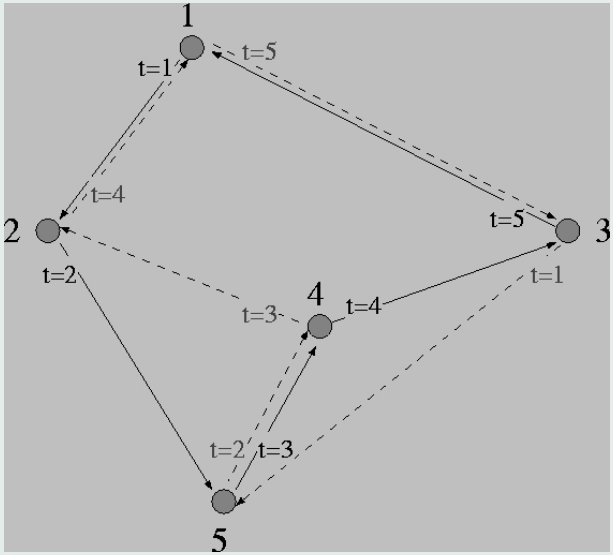
5.4. AG w optymalizacji kombinatorycznej

- Rozpatrzmy problem komiwojażera:
- Aby używać standardowego AG, musimy rozwiązać następujący problem:
  - Znaleźć taką binarną reprezentację trasy, aby móc ją łatwo przekształcić na chromosomy;
  - Funkcja fitness, która uwzględnia warunki zadania;
- Np. macierz permutacji:

$$V_{ct} = \begin{cases} 1 & \text{jeśli } c \text{ jest } i\text{-tym miastem na trasie} \\ 0 & \text{w p.p.} \end{cases}$$

zawierająca dokładnie jedną "1" w każdej kolumnie i w każdym wierszu.

- Np. trasy 12543 i 35421 dla nast. problemu :



może być reprezentowana przez macierz:

miasto	czas 1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	0	0	1
4	0	0	0	1	0
5	0	0	1	0	0

i

miasto	czas 1	2	3	4	5
1	0	0	0	0	1
2	0	0	0	1	0
3	1	0	0	0	0
4	0	0	1	0	0
5	0	1	0	0	0

- Można je zamienić na chromosomy:

  $\chi_1 : 1000001000000010001000100$ i

  $\chi_2 : 000010001010000010001000.$

- AG może stworzyć chromosomy niebędące reprezentantami poprawnych rozwiązań. To się zdarzy:
  - przy inicjalizacji losowych chromosomów; lub
  - przy operatorach genetycznych (mutacja i krzyżowanie).
- Same kłopoty z reprezentacją macierzową. Lepiej szukać innej!

5.5. Inna reprezentacja

- Reprezentacja permutacyjna, np. 12543
- Potrzebne są specjalistyczne operacje genetyczne:

PMX (partially mapped crossover): Dwu-punktowe krzyżowanie definiuje zamianę.

- Np., dla
  $v_1 = \mathbf{12}543$
  $v_2 = \mathbf{35}421$
  dwu-punktowe krzyżowanie definiuje przedział dopasowania
- liczby w przedziale dopasowania tworzą funkcję zamiany, np.
  $1 \Leftrightarrow 3$, $2 \Leftrightarrow 5$.
- Nowy chromosom otrzymuje się przez stosowanie funkcji zamiany do pierwszego chromosomu, np. $v_1 \rightarrow v_3 = 35241$

OX (ordered crossover): OX bierze fragment trasy pierwszego rodzica i zachowuje, w miarę możliwości, porządek w trasie drugiego rodzica:

- Dwu-punktowe krzyżowanie
  $v_1 = 1\mathbf{25}43$
  $v_2 = 3\mathbf{54}21$
- Tworzymy nową trasę przez kopiowanie środkowego fragmentu pierwszej trasy: $v_4 = *25**$;
- zaczynając od drugiego punktu cięcia ustawimy kolejność miast według drugiej trasy: [21345]
- uzupełniamy trasę począwszy od drugiego punktu cięcia: $v_4 = 42513$;

CX (cycle crossover):

- Wybierzny pierwsze miasto z pierwszej trasy
  $v_1 = \mathbf{1}2543$
  $v_2 = 35421$
  $v_5 = 1****$
- kolejne miasto do rozpatrywania jest definiowane przez drugą trasę:
  $v_1 = 1254\mathbf{3}$
  $v_2 = \mathbf{3}5421$
  $v_5 = 1***3$
  powtarzamy to dopóki spotkania cyklu.
- Pozostałe miasta są przepisane z drugiej trasy $v_5 = 15423$

ERX (Edge recombination crossover): potomkowie są tworzone wyłącznie z krawędzi występujących u rodziców:

- Tworzymy macierz połączeń z tras rodziców:

$v_1 = 1\mathbf{2}\mathbf{5}43$
$v_2 = 3\mathbf{5}\mathbf{4}21$

otrzymujemy

miasto	łączone z
1	2 3
2	1 5 4
3	4 1 5
4	5 3 2
5	2 4 3

- Wybierzmy miasto z najmniejszą ilością połączeń: $v_6 = 1****$ i usuwamy informacje z nim związane;

miasto	łączone z
1	2 3
2	1 5 4
3	4 1 5
4	5 3 2
5	2 4 3

powinniśmy na końcu dostać $v_6 = 13542$

6. Strategie ewolucyjne

Główne zastosowanie: optymalizacja funkcji rzeczywistych wielowymiarowych.

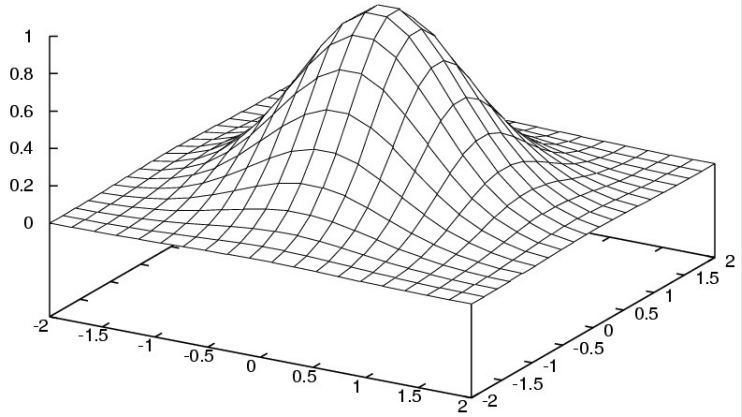
Zadanie: znaleźć maksimum funkcji

$$f : R^n \rightarrow R$$

na danym przedziale

$$D = [a_1, b_1] \times ...[a_n, b_n]$$

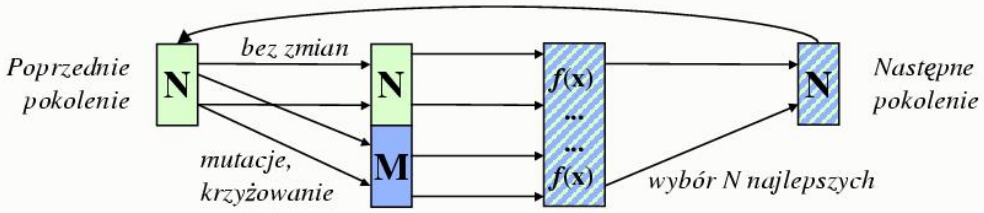
$$f(x, y) = e^{-x^2-y^2}$$



Rozwiązanie: przez $\mathbf{x} = (x_1, ...x_n)$ oznaczmy przykładowe rozwiązanie (punkt z przestrzeni stanów, $x_i \in [a_i, b_i]$). Każdemu wektorowi $\mathbf{x}$ przyporządkujemy pomocniczy wektor wartości rzeczywistych dodatnich: $\sigma = (\sigma_1, ...\sigma_n)$. Para $(\mathbf{x}, \sigma)$ będzie stanowiła kod genetyczny osobnika.

SCHEMAT DZIAŁANIA

1. Tworzymy losową populację złożoną z N osobników postaci $(\mathbf{x},\sigma)$.
2. Dopisujemy do niej M osobników potomnych, tworzonych na podstawie losowo wybranych osobników z poprzedniego pokolenia za pomocą mutacji i krzyżowania.
3. Dla każdego osobnika w populacji pośredniej liczymy wartość optymalizowanej funkcji.
4. Spośród N+M osobników wybieramy N najlepszych względem $f(\boldsymbol{x})$. Te osobniki przeżyją i utworzą następne pokolenie.
5. Powtarzamy od punktu 2. z aktualną populacją N-osobnikową.



OPERATORY

Mutacji podlegają zwykle wszystkie osobniki dodawane do populacji pośredniej. Mutacja polega na zmianie parametrów (x,σ) zgodnie ze wzorami:

$$\sigma_i = \sigma_i \cdot e^{N(0,\Delta)}$$

$$x_i = x_i + N(0,\sigma_i)$$

gdzie N(0,σ) oznacza liczbę losową wygenerowaną zgodnie z rozkładem normalnym o wart. oczekiwanej **0** i odchyleniu standardowym σ.

Krzyżowaniu podlega część (np. 25%) osobników dodawanych do populacji pośredniej. Osobnik potomny składany jest z genów dwóch losowo wybranych osobników rodzicielskich.

$(x_1, x_2, x_3, x_4, x_5, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$

$(y_1, y_2, y_3, y_4, y_5, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5)$

$\rightarrow (x_1, y_2, y_3, x_4, y_5, \sigma_1, \tau_2, \tau_3, \sigma_4, \tau_5)$

Uwaga: wartości x_i są dziedziczone zawsze razem z odpowiadającymi im wartościami pomocniczymi σ_i.