Algorytm mrówkowy

Metody optymalizacji

Rafał Adamczak
Katedra Informatyki Stosowanej

www.fizyka.umk.pl/~raad/optymalizacja.pdf

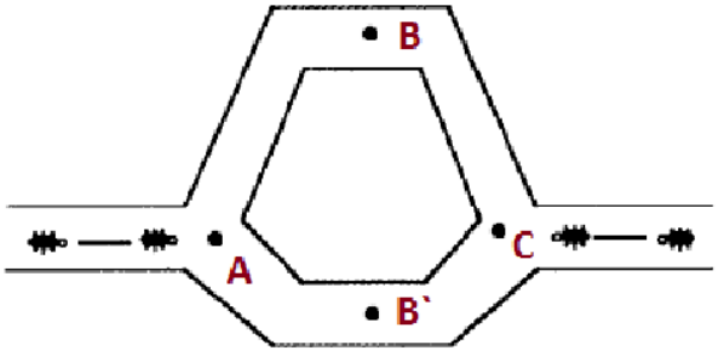
Systemy mrówkowe,podobnie jak algorytmy genetyczne, są systemami opartymi na populacji

Występuje populacja mrówek, w której każda z mrówek, która znajdzie rozwiązanie, informuje o tym pozostałe mrówki

- Mrówki są właściwie ślepe, a mimo to są w stanie znaleźć drogę do pożywienia i z powrotem do mrowiska. Jak to możliwe?
- Obserwacje zachowania mrówek stały się inspiracją do powstania nowego typu algorytmów zwanych mrówkowymi (albo systemami mrówkowymi)
- Algorytm mrówkowy powstał w 1996, został wymyślony przez Dorigo, w chwili obecnej jest wciąż intensywnie rozwijany

- Czas t w algorytmie mrówkowym jest wielkością dyskretną
- W każdej chwili czasu mrówka przemieszcza się o odległość d=1
- Po każdym ruchu mrówka pozostawia na drodze jedną jednostkę feromonu
- W chwili t=0 nie ma na żadnej ścieżce żadnej jednostki feromonu.

- Wyjście mrówki z mrowiska
- Mrówka zostawia za sobą ślad feromonowy
- Druga mrówka wychodząca z mrowiska ma do wyboru pójść własną ścieżką albo po śladach feromonowych poprzedniej mrówki, im silniejszy feromon tym większe prawdopodobieństwo że nowa mrówka wybierze tą ścieżkę
  - Silny ślad feromonowy jest wówczas, gdy ścieżka jest często odwiedzana, lub gdy odległość do jedzenia jest bliska (feromon paruje)



- Interesuje nas eksploracja przestrzeni, a nie tylko wyznaczanie trasy
- W związku z tym mrówka wybierać będzie drogę z pewnym prawdopodobieństwem, które będzie prawdopodobne do intensywności feromonów
- Prawdopodobieństwo jest nie tyko funkcją intensywności feromonów, ale również tego co mrówka widzi
- Ścieżka feromonowa nie może trwać wiecznie dlatego konieczne jest żeby feromony parowały

Algorytm

```
procedure ACO
    while(nie koniec)
       Wygeneruj_rozwiązanie()
       Odśwież_znaki_feromonowe()
    end while
  end procedure
```

Algorytm mrówkowy dla przypadku komiwojażera

- Pojedyncze mrówki umieszczone są w każdym z miast. (t=0), w czasie t+1 mrówki przemieszczają się do kolejnego miasta
- Każdy z węzłów grafu zawiera informację o intensywności feromonów prowadzących do tego miasta.
- Niech $T_{ij}(t)$ oznacza intensywność krawędzi (i,j) w czasie t
- Gdy mrówka decyduje do którego miasta powędrować robi to z prawdopodobieństwem, które zależne jest od odległości do tego miasta i intensywności feromonów na krawędzi do niego prowadzącej
- Odległość do następnego miasta określana jest jako to co widzi mrówka, n_{ij}, i zdefiniowana jest jako $1/d_{ij}$,

- W każdej chwili czasu następuje parowanie foromonów. Ilość wyparowanego feromonu p, jest wartością pomiędzy 0 i 1
- Ażeby uniknąć odwiedzania przez mrówkę tego samego miasta stosowana jest tablica Tabu
- $Tabu_k$ jest listą miast dla k-tej mrówki, które zostały przez nią odwiedzone
- Po każdej turze intensywność feromonów na odpowiednich ścieżkach jest uaktualniana zgodnie ze wzorem:

$$T_{ij}(t+n) = (1-\rho)\, T_{ij}(t) + \Delta T_{ij}$$

gdzie ρ współczynnik parowania feromonu

$$\Delta T_{ij} = \begin{cases} \frac{Q}{L_k} \; jeśli\, krawędź\,(i,j)\, należy\, do\, trasy\, mrówki\, k \\ 0\, w\, przeciwnym\, razie \end{cases}$$

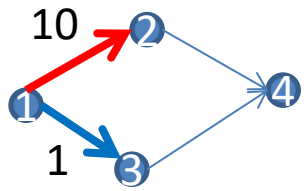
Gdzie Q to pewna dobrana stała, zależna od rozpatrywanego problemu, a L_k to koszt rozwiązania związanego z mrówką k

Prawdopodobieństwo przejścia z i do j

$$p_{ij}^{k} = \begin{cases} \dfrac{T_{ij}^{\alpha}\sigma_{ij}^{\beta}}{\sum_{h \notin Tabu_k} T_{ih}^{\alpha}\sigma_{ih}^{\beta}} & jeżeli\ j \notin Tabu_k \\ 0 & \end{cases}$$

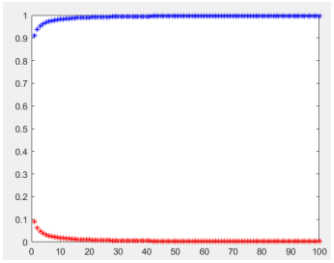
Gdzie sigma to parametr wskazujący na atrakcyjność przejścia, w tym przypadku związana Jest z odległością do rozpatrywanego miasta.
Parametry alfa i beta to parametry regulujące ważność intensywności feromonu względem atrakcyjności

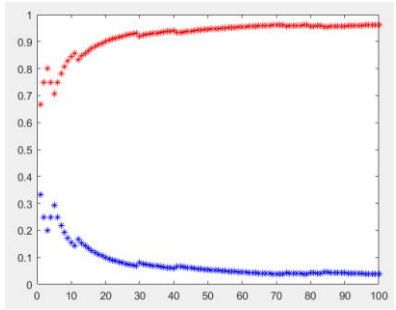
Przykład




```
R=[ 1 2 10 1;
    1 3 1  1;
    2 4 1  1;
    3 4 1  1];
```

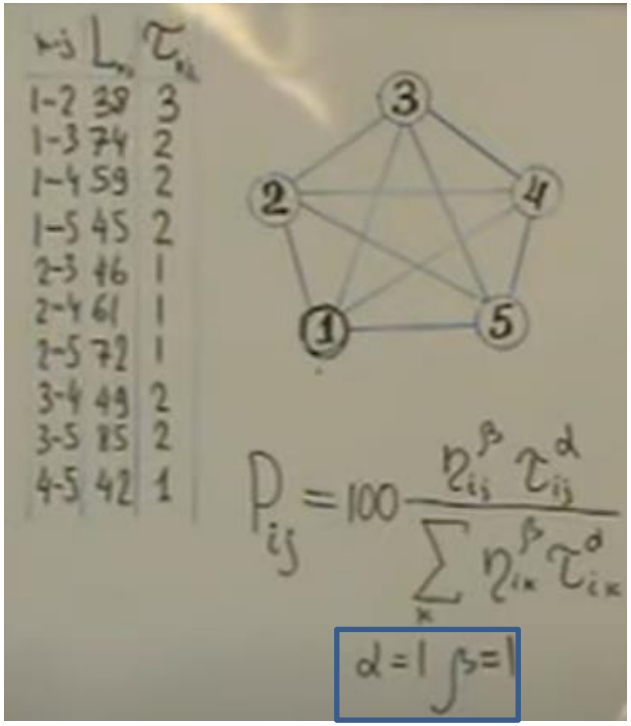
```
R=[ 1 2 1  1;
    1 3 2  1;
    2 4 1  1;
    3 4 1  1];
```

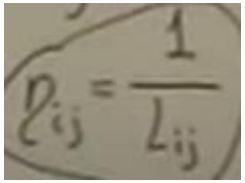




```
Q=1;
for i=1:100
   L=[1];
   P12=R(1,4)*1/R(1,3);
   P12=P12/(R(1,4)*1/R(1,3)+R(2,4)*1/R(2,3));
   P13=R(2,4)*1/R(2,3);
   P13=P13/(R(1,4)*1/R(1,3)+R(2,4)*1/R(2,3));
   disp([P12 P13]);
   plot(i,P12,'*r');
   hold on;
   plot(i,P13,'*b');
   P=rand;
   if P>P12
      %3 ->4
      L=[L,3,4];
      P34=R(4,4)*1/R(4,3);
      P34=P34/(R(4,4)*1/R(4,3));
      D=R(2,3)+R(4,3);
      deltaT=Q/D;
      R(2,4)=R(2,4)+deltaT;
      R(4,4)=R(4,4)+deltaT;
   else
      %2 ->4
      L=[L,2,4];
      P24=R(3,4)*1/R(3,3);
      P24=P34/(R(3,4)*1/R(3,3));
      D=R(1,3)+R(3,3);
      deltaT=Q/D;
      R(1,4)=R(1,4)+deltaT;
      R(3,4)=R(3,4)+deltaT;
    end;
 end;
```

Zagadnienie komiwojażera





$$P_{ij}(t) = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{j \in \Omega} [\tau_{ij}]^{\alpha} * [\eta_{ij}]^{\beta}}, & \text{dla } j \in \Omega \end{cases} \qquad (1)$$

gdzie:

τ_{ij} - natężenie śladu feromonowego na krawędzi,

η_{ij} - wartość funkcji kryterium związana z widocznością punktu j z punktu i,

α - parametr sterujący ważnością intensywności śladu feromonowego τ_{ij},

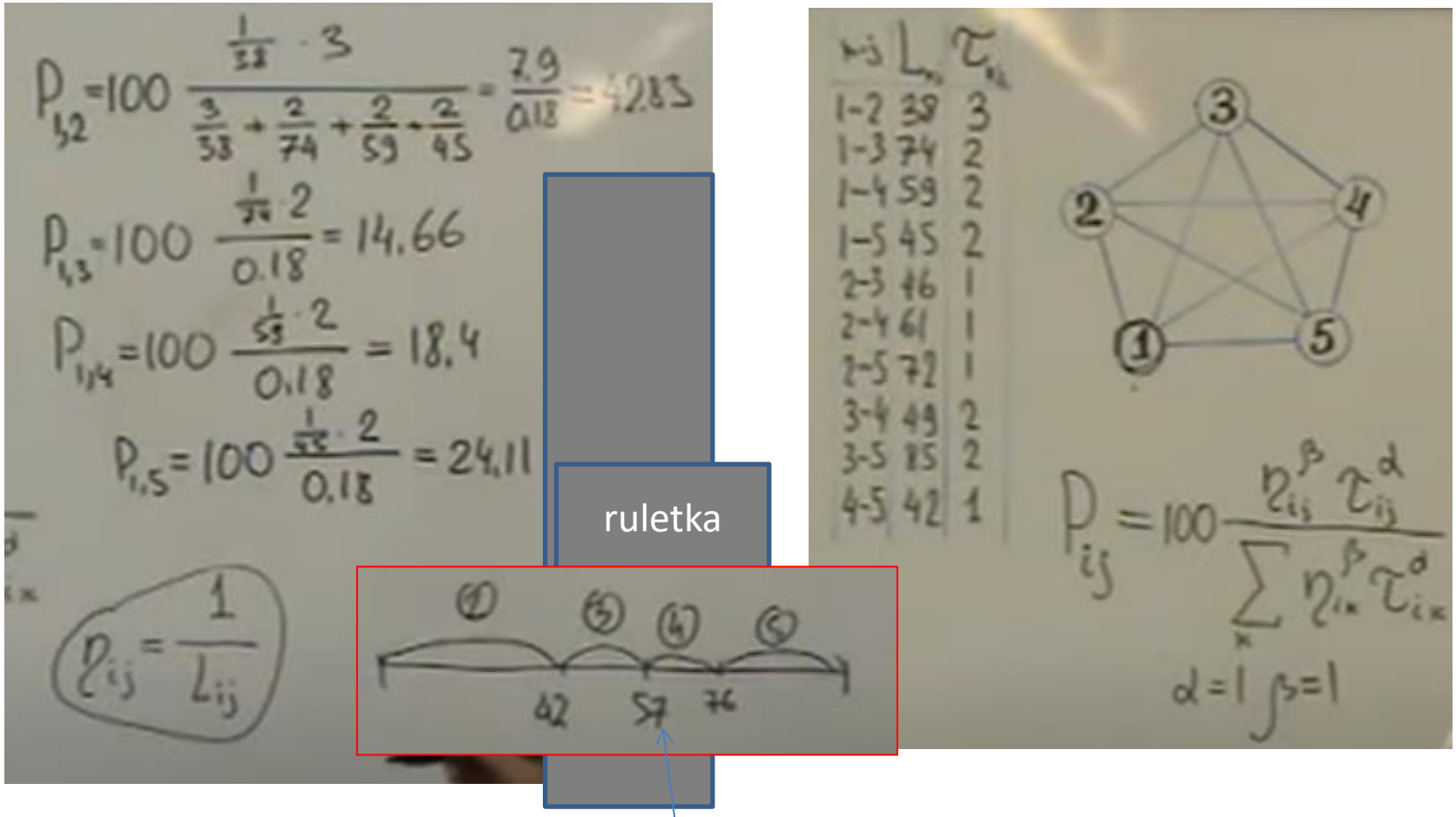
β - parametr sterujący ważnością widoczności następnego miasta,

Ω - zbiór miast nie odwiedzonych dotychczas przez mrówkę.

$$\eta_{ij} = \frac{1}{d_{ij}^2} \qquad (2)$$

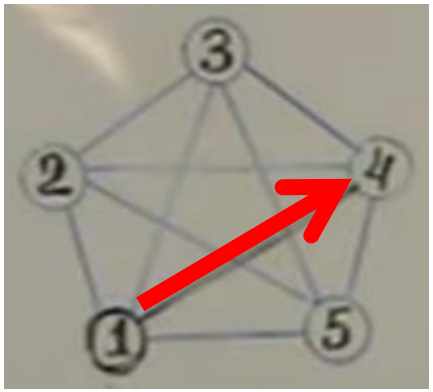
przy czym d_{ij} to długość krawędzi pomiędzy miastami i oraz j.

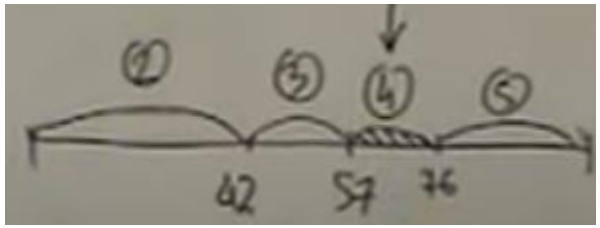
Krok 1



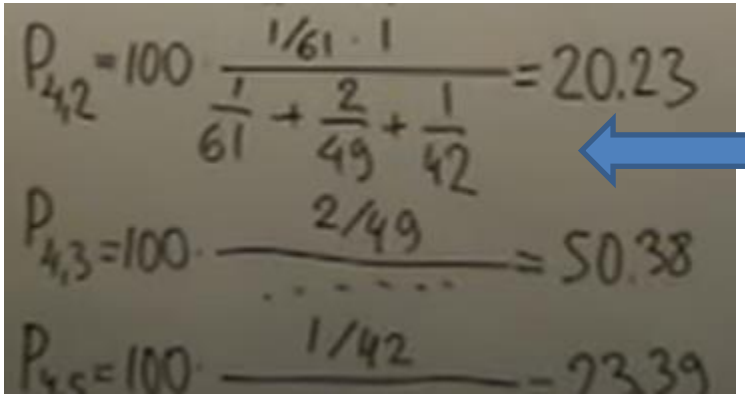
Krok 2

Wybór padł na 4



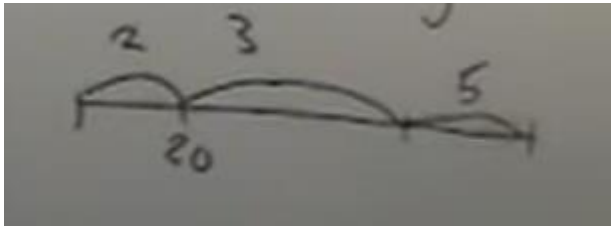


Lista odwiedzonych miast
1



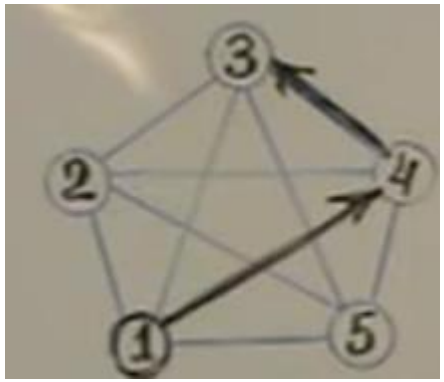
dla trzech pozostałych miast!

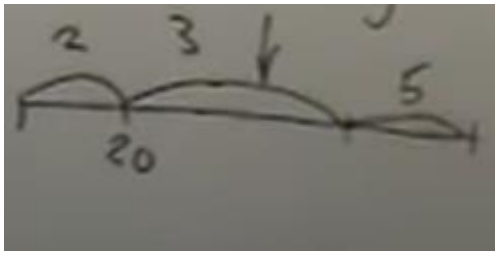
i-j	L_{ij}	τ_{ij}
1-2	39	3
1-3	74	2
1-4	59	2
1-5	45	2
2-3	46	1
2-4	61	1
2-5	72	1
3-4	49	2
3-5	85	2
4-5	42	1

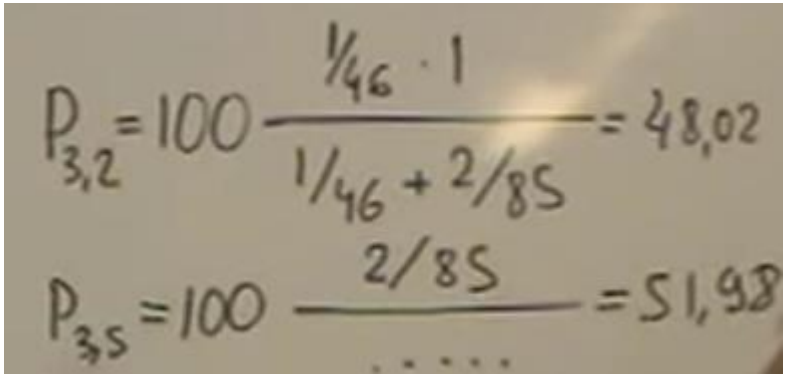


Krok 3

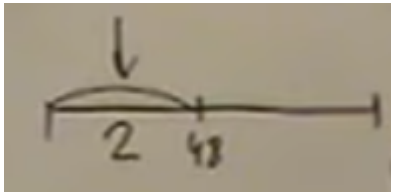
Lista odwiedzonych miast
1,4





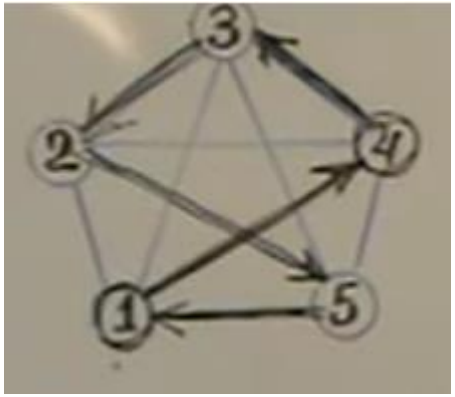


i-j	L_{ij}	τ_{ij}
1-2	39	3
1-3	74	2
1-4	59	2
1-5	45	2
2-3	46	1
2-4	61	1
2-5	72	1
3-4	49	2
3-5	85	2
4-5	42	1



Kolejne kroki

Lista odwiedzonych miast
1,4,3,2,5,1



Długość ścieżki

$$\sum_k L_k = L_0$$

Odświeżanie feromonu w grafie

kj	L_{kj}	τ_{kj}
1-2	38	3
1-3	74	2
1-4	59	2
1-5	45	2
2-3	46	1
2-4	61	1
2-5	72	1
3-4	49	2
3-5	85	2
4-5	42	1

$$\Delta\tau = \frac{Q}{L_0}$$

$$\tau^*_{kj} = \tau_{kj} + \Delta\tau$$

kj – **tylko dla ścieżki L0**,
dla pozostałych – bez zmian

Dodawanie

$$\tau^*_{kj} = (1-p)\tau_{kj} + \Delta\tau$$

Dodawanie + zapomnienie

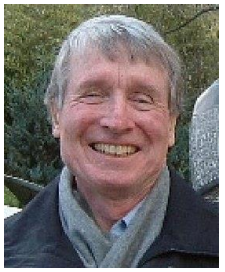
Ant Colony Optimization

https://www.codeproject.com/Articles/644067/Applying-Ant-Colony-Optimization-Algorithms-to-Sol

Algorytmy genetyczne

- Twórcą algorytmu jest Holland (1962).
- Są oparte na mechanizmach doboru naturalnego i dziedziczenia.
- Nie wymagają liczenia gradientu, a więc nadają się do funkcji nieciągłych
- Należą do metod stochastycznych
- Bardzo łatwe do implementacji
- Łatwe do zrównoleglania

Metody ewolucyjne powstały w celu znajdowania przybliżonego rozwiązania problemów optymalizacyjnych w taki sposób, by znajdować wynik w miarę szybko oraz uniknąć lokalnych minimów.



John Henry Holland
1929-2015
Alma mater
University of Michigan
Known for Research on genetic algorithms

Algorytm genetyczny

Algorytm genetyczny to termin ogólny używany do określenia pewnej grupy metod rozwiązywania zagadnień optymalizacji. Wspólną podstawą koncepcyjną tych metod jest **symulowanie ewolucji indywidualnych struktur poprzez proces selekcji i stosowanie operatorów genetycznych**. Za pomocą tych operatorów są tworzone nowe warianty (populacje) rozwiązań. W algorytmach genetycznych stosuje się pojęcia zapożyczone z genetyki naturalnej.

Podstawowe pojęcia

- **Populacja** – zbiór osobników (jednostek) o określonej liczebności, których przystosowanie do środowiska jest określone.
- **Osobniki** populacji w algorytmach genetycznych to zakodowane w postaci chromosomów zbiory parametrów zadania, czyli rozwiązania, określane też jako punkty przestrzeni poszukiwań.
- **Chromosomy** – inaczej łańcuchy lub ciągi kodowe – to uporządkowane ciągi genów.
- **Gen** – nazywany też cecha, znakiem, detektorem – jest to pojedynczy element genotypu, w szczególności chromosomu.
- **Genotyp**, czyli struktura - to zespół chromosomów danego osobnika. Osobnikami populacji mogą być genotypy albo pojedyncze chromosomy.
- **Fenotyp** - zestaw wartości odpowiadający danemu genotypowi, czyli zdekodowana struktura. Jest to zbiór parametrów zadania (rozwiązanie, punkt przestrzeni poszukiwań).
- **Allel** to wartość danego genu (określana też jako wartość cechy lub wariant cechy).
- **Funkcja przystosowania** (ang. fitness function) nazywana też funkcją dopasowania lub funkcją oceny. Stanowi ona miarę przystosowania (dopasowania) danego osobnika w populacji.
- **Generacja** – to kolejna iteracja w algorytmie genetycznym.
- **Pokolenie** (nowe pokolenie lub pokolenie potomków) – to nowo utworzona populacja osobników.

Operacje w GA

W algorytmie genetycznym kolejne populacje generowane są poprzez zastosowanie

następujących operacji:

1) reprodukcja

2) krzyżowanie

3) mutacja

Algorytm genetyczny

1) Utwórz losową populację zawierającą **n** chromosomów

2) Oceń przystosowanie każdego chromosomu.

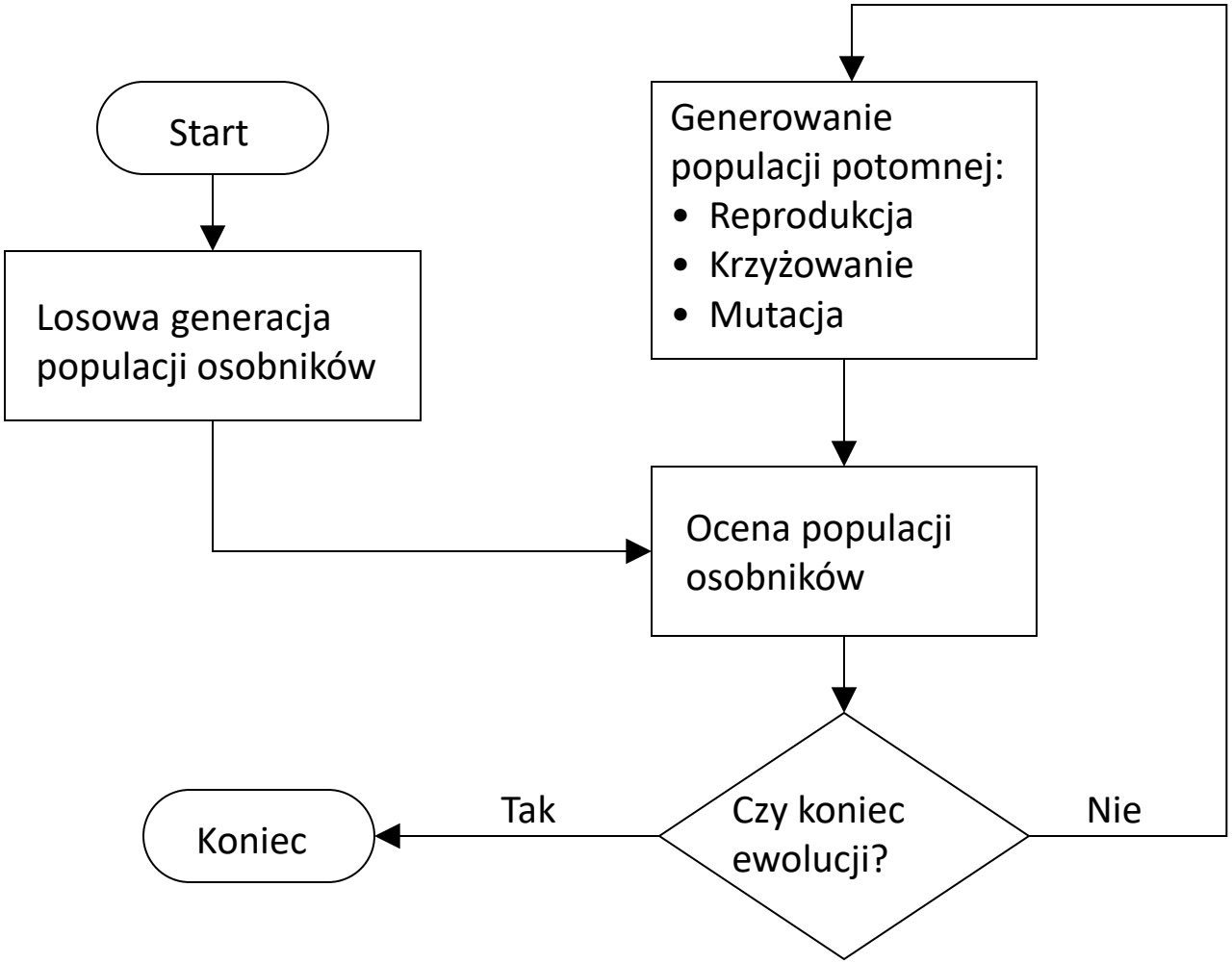
3) Utwórz nową populację przez zastosowanie następujących kroków:

- Selekcja - wybierz dwa chromosomy z populacji w sposób losowy, ale z uwzględnieniem dopasowania
- Z wybranych chromosomów utwórz 2 nowe poprzez zastosowanie operacji krzyżowania
- Geny w chromosomie ulegają mutacji z pewnym prawdopodobieństwem **p**
- Umieść utworzone chromosomy w nowej populacji

4) Sprawdź warunek stopu, jeśli nie spełniony idź do punktu 2.

1) Znalezione zostało rozwiązanie z satysfakcjonującą wartością minimalizowanej (maksymalizowanej) funkcji.
2) Osiągnięta została ustalona liczba generacji.
3) Osiągnięty został maksymalny czas.
4) Osiągnięto plateau tzn. nie następuje poprawa dopasowania.

Standardowy algorytm genetyczny – ogólny schemat



Zalety i wady algorytmów genetycznych

Zalety:

- Odporność - unikanie ekstremów lokalnych, prawdopodobieństwo znalezienia dobrych rozwiązań jest w dużym stopniu niezależne od wyboru punktów początkowych
- Wydajność – duża liczba przetwarzanych schematów - ok. m^3, gdzie m - liczba osobników w populacji
- Łatwość zastosowania w niemal każdym zadaniu optymalizacji

Wady:

- Brak gwarancji zbieżności do optymalnego rozwiązania

Różnice pomiędzy algorytmami genetycznymi a tradycyjnymi metodami szukania

- Algorytmy genetyczne przetwarzają zakodowaną postać parametrów zadania (ciąg kodowy) a nie same parametry
- Poszukiwania prowadzone są w obrębie całej populacji rozwiązań (osobników) a nie pojedynczego rozwiązania
- Wykorzystywana jest tylko funkcja celu (uczenie z krytykiem) bez żadnej dodatkowej informacji naprowadzającej np. pochodnej funkcji celu
- Stosowane są probabilistyczne a nie deterministyczne reguły wyboru

Porównania

Optymizacja klasyczna
Generuje pojedynczy punkt, w każdej iteracji. Sekwencja punktów zbliża optymalne rozwiązanie.

Wybiera następny punkt w sekwencji od deterministycznego obliczenia

Algorytm genetyczny
Generuje populacji punktów, w każdej iteracji. Najlepszym punktem w populacji osiąga optymalne rozwiązanie.

Wybiera następną populację przez obliczenia, który wykorzystuje generatory liczb losowych.

Przykład 1

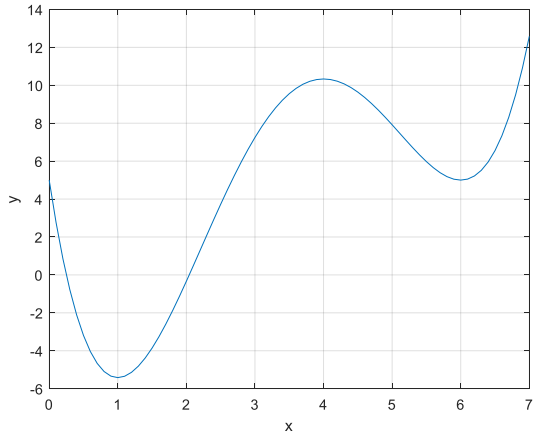
- Załóżmy, że chcemy znaleźć **globalne minimum** funkcji

$$y(x) = 5 - 24x + 17x^2 - \frac{11}{3}x^3 + \frac{1}{4}x^4$$

na odcinku [0;7]

MatLab

- >> x=0:0.1:7;
- y=5-24*x+17*x.^2-11/3*x.^3+1/4*x.^4;
- plot(x, y)
- xlabel('x'), ylabel('y')
- grid on



Ta funkcja ma minimalną wartość w punkcie x = 1. Oczywiście, w punkcie x = 6 funkcja spada do minimum lokalnego. Jeśli użyć minimum gradientu, w zależności od początkowej zbliżenia można osiągnąć na lokalne minimum.

Zasada działania algorytmów genetycznych

Szukamy rozwiązanie na liczbach całych $x \in \{0, 1, 2, 3, 4, 5, 6, 7\}$.

Wybieramy losowo kilka liczb w przedziale[0; 7]: {**2, 3, 5, 4**}.

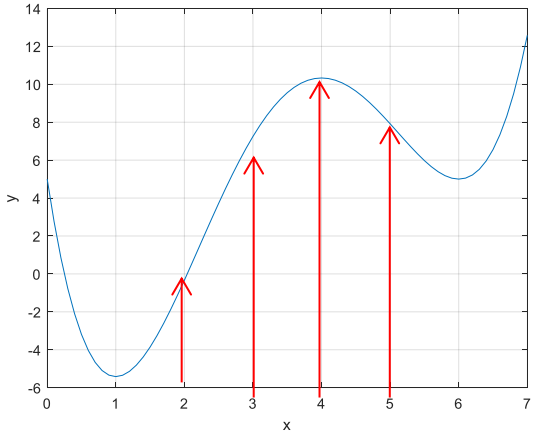
Weźmiemy pod uwagę te liczby w postaci **próbnych rozwiązań** poszukiwania globalnego minimum.
Podstawowa idea algorytmów genetycznych jest organizacja "walki o przetrwaniem" i "dobór naturalny" tych **próbnych rozwiązań**.

Zakodujemy zbiór w postaci binarnej : {**010, 011, 101, 100**}.
To nazywa się **populację**.
Jedno rozwiązanie nazywa się **chromosomem** lub osobnikiem.

Funkcja przystosowania

Jak wiemy, **zasada doboru naturalnego jest konkurencyjne przetrwanie najsilniejszych**.
W naszym przypadku, przydatność osobnika zależy od **funkcji celu (przystosowania)** .

```
>> x=[2, 3, 5, 4];
y=[ ];
for  i=1:4
y(i)=5-24*x(i)+17*x(i)^2-11/3*x(i)^3+1/4*x(i)^4;
end
y
y =
   -0.3333    7.2500    7.9167   10.3333
```

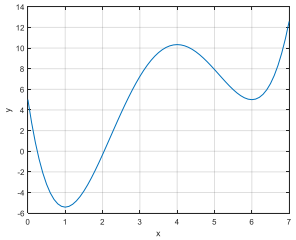


Analiza

Osobnik	argument	dwójkowy zapis	przystosowanie
1	2	010	-0.3333
2	3	011	7.2500
3	5	101	7.9167
4	4	100	10.3333

Generowanie populacji potomnej

Osobnik	kod	drugi osobnik	kod	krzyżowanie	
1	010/2	1	010	1	000/0
2	011/3	4	100		110/6
3	101/5	3	101	2	100/4
4	100/4	1	010		011/3



Osobnik	kod	losowa liczba	numer geny	mutacja		
1	000/0	0,1	3	001/1	5	– 5,42
2	110/6	0,6	–	110/6	5	5
3	100/4	0,5	–	100/4	10,33	10,33
4	011/3	0,2	1	111/7	7,25	12,58

Funkcja przystosowania

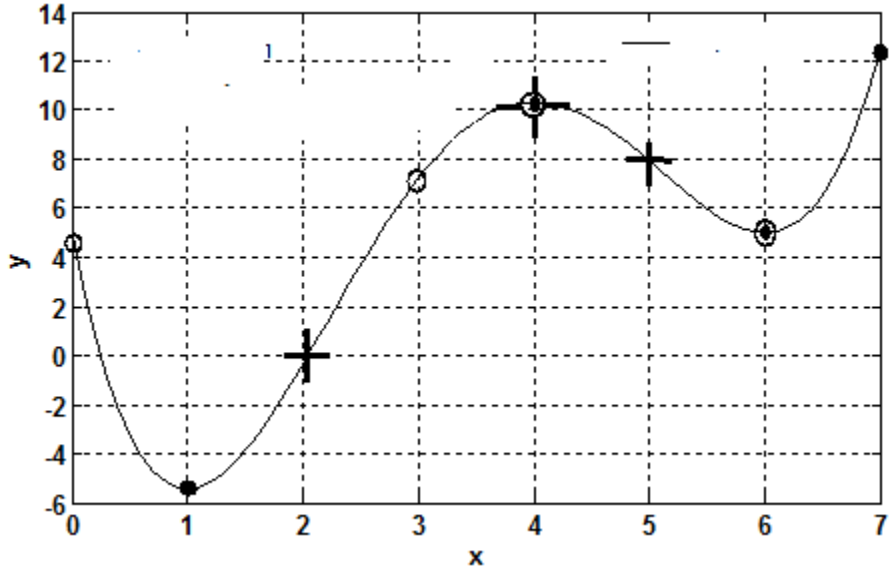
Przed mutacją

```
>> x=[0, 6, 4, 3];
y=[ ];
for  i=1:4
y(i)=5-24*x(i)+17*x(i)^2-11/3*x(i)^3+1/4*x(i)^4;
end
y
y =
   5.0000    5.0000   10.3333    7.2500
```

Po mutacji

```
>> x=[1, 6, 4, 7];
y=[ ];
for  i=1:4
y(i)=5-24*x(i)+17*x(i)^2-11/3*x(i)^3+1/4*x(i)^4;
end
y
y =
  -5.4167    5.0000   10.3333   12.5833
```

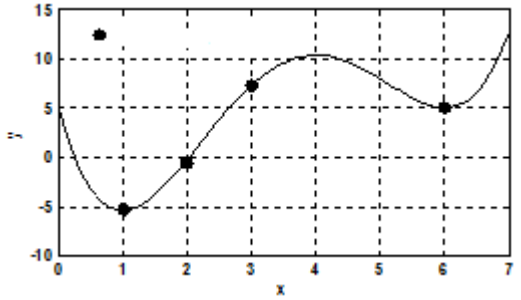
Analiza



Nowa populacja

1	010/2	-0,33	001/1	-5,42
2	011/3	7,25	010/2	-0,33
3	101/5	7,92	110/6	5
4	100/4	10,33	011/3	7,25

```
>> x=[1, 2, 6, 3];
y=[ ];
for  i=1:4
y(i)=5-24*x(i)+17*x(i)^2-
11/3*x(i)^3+1/4*x(i)^4;
end
y
y =
   -5.4167   -0.3333    5.0000    7.2500
```



x=[2, 3, 5, 4];

x=[1, 6, 4, 7];

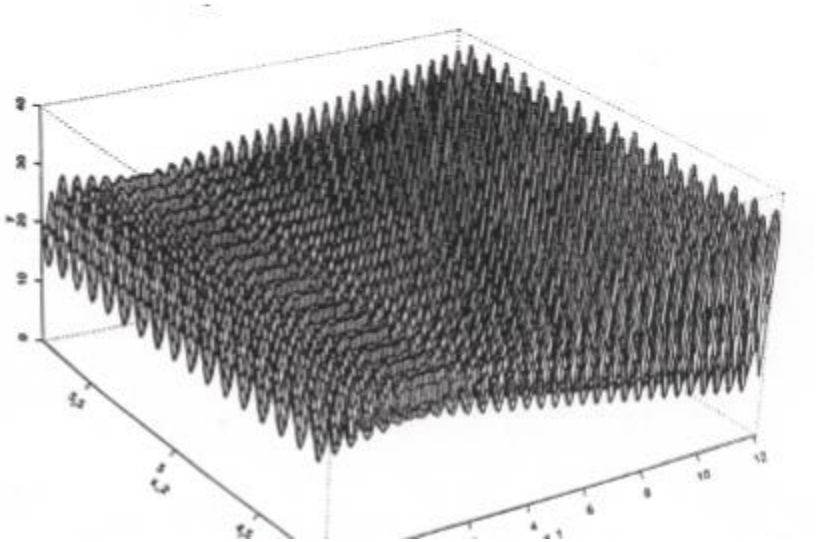
x=[1, 2, 6, 3];

Przykład 2

Maksymalizujemy funkcję:

$$f(x_1, x_2) = 21{,}5 + x_1 \sin(4\pi x_1) + x_2 * \sin(20\pi x_2)$$

Przy ograniczeniach $-3{,}0 < x_1 < 12{,}1$ i $4{,}1 < x_2 < 5{,}8$



Przykład 2 (cd)

Zakładamy że żądana dokładność wynosi cztery najbardziej znaczące cyfry dla każdej zmiennej. Przedział zmienności x_1 ma długość 15,1. Z żądanej dokładności wynika, że przedział [-3,0, 12,1] należy podzielić na przynajmniej 15,1 * 10 000 równych podprzedziałów. Oznacza to, że początkowa część chromosomu musi zawierać m_1= 18 bitów, gdyż

$2^{17}<151000<2^{18}$

Przykład 2 (cd)

Przedział zmienności x_2 ma długość 1,7. Z żądanej dokładności wynika, że przedział [4,1, 5,8] należy podzielić na przynajmniej 1,7 * 10 000 równych podprzedziałów. Oznacza to, że końcowa część chromosomu musi zawierać m_2 =15 bitów, gdyż

$2^{14} < 17\ 000 < 2^{15}$

Całkowita długość chromosomu (wektora rozwiązań) wynosi więc $m = m_1+m_2= 18 + 15 = 33$ bitów.

W pierwszych 18 bitach jest zakodowane x_1, a w pozostałych 15 bitach (19 - 33)x_2

Przykład 2 (cd)

Rozważmy chromosom:

v_i = (010001001011010000111110010100010)

x_1 x_2

m_1=18 m_2=15

x_1= -3,0 +*decimal*(0100010010110100002)(12,1- (-3,0))/(2↑18-1)

x_1= 1,052426

x_2= 4,1 + *decimal*(1111100101000102)(5,8-4,1)/(2↑15-1)

x_2=5,755330

$f(x_1,x_2)$ = f(1,052426;5,755330)= 20.2526680532854

Przykład 2 (cd)

Do optymalizacji funkcji f za pomocą algorytmu genetycznego utworzymy populację o liczbie pop_size =20 chromosomów.

Wszystkie 33 bity w każdym chromosomie są na początku wybierane losowo

Zakładamy że po procesie początkowego wyboru chromosomów otrzymaliśmy następującą populacje

Przykład 2 (cd)

W kroku oceny dekodujemy każdy chromosom i obliczamy funkcję oceniającą dla właśnie rozkodowanych wartości (x_1, x_2). Dostajemy:

V_1 = (10011010000000111111010011011111)
V_2 = (11100010010011011100101010001 1010)
V_3 = (00001000001100100000101011011101)
V_4 = (10001100010110100111100000111 0010)
V_5 = (00011101100101001101011111100 0101)
V_6 = (00010100001001010100101011111 1011)
V_7 = (00100010000011010111101101111 1011)
V_8 = (10000110000111010001011010110 0111)
V_9 = (01000000010110001011000000111 1100)
V_{10} = (00000111100011000001101000011 1011)
V_{11} = (11010001011110110100010101000 0000)
V_{12} = (11101111101000100011000000100 0110)
V_{13} = (01001001100000101010011110010 1001)
V_{14} = (01100111111011010110000110111 1000)
V_{15}=(11101110110111000010001111101 1110)
V_{16}=(11001111000001111110000110100 1011)
V_{17}= (01101011111100111101000110111 1101)
V_{18}=(01110100000000111010011111010 5101)
V_{19}= (00010101001111111111000011000 1100)
V_{20}=(10111001011001111001100010111 1110)

eval(V_1) = f (6,084492, 5,652242) = 26,019600
eval(V_2) = *f* (10,348434, 4,380264) = 7,580015
eval(V_3) =*f*(-2,516603, 4,390381) = 19,526329
eval(V_4) =f (5,278638, 5,593460) = 17,406725
eval(V_5) = f (-1,255173, 4,734458) = 25,341160
eval(V_6) = *f* (-1,811725, 4,391937) = 18,100417
eval(V_7) = *f*(-0,991471, 5,680258) = 16,020812
eval(V_8) =f (4,9i06i8, 4, 7031Ji8) = 17,959701
eval(V_9) = f(0,795406, 5,381472) = 16,127799
eval(V_{10}) = f(-2,554851, 4,793707) = 21,278435
eval(V_{11}) = f(3,130078, 4,996097) = 23,410669
eval(V_{12}) = f(9,356179, 4,239457) = 15,011619
eval(V_{13}) = f(11,134646, 5,378671) = 27,316702
eval(V_{14}) = f(1,335944, 5,151378) = 19,876294
eval(V_{15}) = f(11,089025, 5,054515) = 30,060205
eval(V_{16}) = f(9,211598, 4,993762) = 23,867227
eval(V_{17}) = f(3,367514, 4,571343) = 13,696165
eval(V_{18}) = f(3,843020, 5,158226) = 15,414128
eval(V_{19}) = f(-1746635, 5,395584) = 20,095903
eval(V_{20}) = f(7,935998, 4,757338) = 13,666916

Przykład 3

/dioph.php.htm

Równanie diofantyczne

$a+2b+3c+4d=30$

$1 =< a,b,c,d =< 30$

Liczby całkowite

1	(1,28,15,3)
2	(14,9,2,4)
3	(13,5,7,3)
4	(23,8,16,19)
5	(9,13,5,2)

Populacja i chromosomy

1	|114-30|=84
2	|54-30|=24
3	|56-30|=26
4	|163-30|=133
5	|58-30|=28

Funkcja przystosowania

https://pl.wikipedia.org/wiki/R%C3%B3wnanie_diofantyczne

Przykład 3 (cd)

Wybór rodziców do generacji nowej populacji

1	(1/84)/0.135266 = 8.80%
2	(1/24)/0.135266 = 30.8%
3	(1/26)/0.135266 = 28.4%
4	(1/133)/0.135266 = 5.56%
5	(1/28)/0.135266 = 26.4%

Do wyboru 5 par rodziców (każda z nich będzie miała 1 potomka, Łącznie - 5 nowych decyzji)
Wyobraź sobie, że mamy 10 000 stronną kość,
na 880 stronach jest oznaczony chromosom 1
na 3080 - chromosom 2
na 2640 stronach - chromosom 3,
na 556 - chromosom 4 i
na 2640 stronach zaznaczono chromosom 5.

Przykład 3 (cd)

Rodzice

3	1
5	2
3	5
2	5
5	3

Krzyżowanie i generacja potomków

$a_1 \mid b_1,c_1,d_1$	$a_2 \mid b_2,c_2,d_2$	a_1,b_2,c_2,d_2 or a_2,b_1,c_1,d_1
$a_1,b_1 \mid c_1,d_1$	$a_2,b_2 \mid c_2,d_2$	a_1,b_1,c_2,d_2 or a_2,b_2,c_1,d_1
$a_1,b_1,c_1 \mid d_1$	$a_2,b_2,c_2 \mid d_2$	a_1,b_1,c_1,d_2 or a_2,b_2,c_2,d_1

Przykład 3 (cd)

3	1
5	2
3	5
2	5
5	3

(13 | 5,7,3)	(1 | 28,15,3)	(13,28,15,3)
(9,13 | 5,2)	(14,9 | 2,4)	(9,13,2,4)
(13,5,7 | 3)	(9,13,5 | 2)	(13,5,7,2)
(14 | 9,2,4)	(9 | 13,5,2)	(14,13,5,2)
(13,5 | 7, 3)	(9,13 | 5, 2)	(13,5,5,2)

Nowa populacja

(13,28,15,3)	|126-30|=96
(9,13,2,4)	|57-30|=27
(13,5,7,2)	|57-30|=22
(14,13,5,2)	|63-30|=33
(13,5,5,2)	|46-30|=16

Następna generacja może **mutować**. Na przykład możemy zastąpić jedną z wartości chromosomu losową liczbą całkowitą od 1 do 30.

Reprodukcja
Proces w którym elementy populacji zostają powielone w stosunku zależnym od wartości, jakie przybiera funkcja celu *f* (funkcja przystosowania) inaczej mówiąc przeżywają najlepiej przystosowani.

Niech f_i oznacza dopasowanie *i*-tego osobnika, wówczas do reprodukcji *i*-ty osobnik wybrany zostanie z prawdopodobieństwem p_i

$$p_i = \frac{f_i}{\sum_{i}^{N} f_i}$$

Rodzaje selekcji: turniejowa, rankingowa, elitarna itp.

Selekcja turniejowa
Polega na losowym wyborze z całej populacji kilku osobników, a następnie z puli wybranych osobników wybierany jest ten, który jest najlepiej przystosowany. Cała procedura powtarzana jest tak długo aż osiągnie się zadaną liczbę osobników.

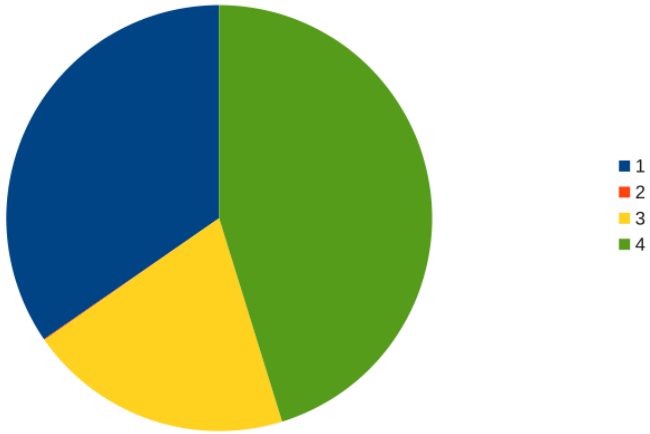
Selekcja rankingowa
Dla każdego osobnika wyliczane jest jego przystosowanie, na tej podstawie osobnicy są sortowani i wybierany jest k najlepszych.

Selekcja elitarna
W selekcji elitarnej do następnej populacji zawsze kopiowany jest najlepszy osobnik, lub k najlepszych osobników. Selekcja tego typu nie jest samodzielna, musi towarzyszyć innemu typowi selekcji.

Selekcja typu ruletka

Przypuśćmy, poszczególnym ciągom kodowym z populacji odpowiada sektor koła ruletki o wymiarze proporcjonalnym do przystosowania. Dla każdego ciągu kodowego możemy wyznaczyć wartość funkcji celu

Nr	Ciąg kodowy	Przystosowanie	%całości
1	10101	441	34.6
2	00001	1	0
3	10000	256	20.1
4	11000	576	45.2
Łącznie		1274	100



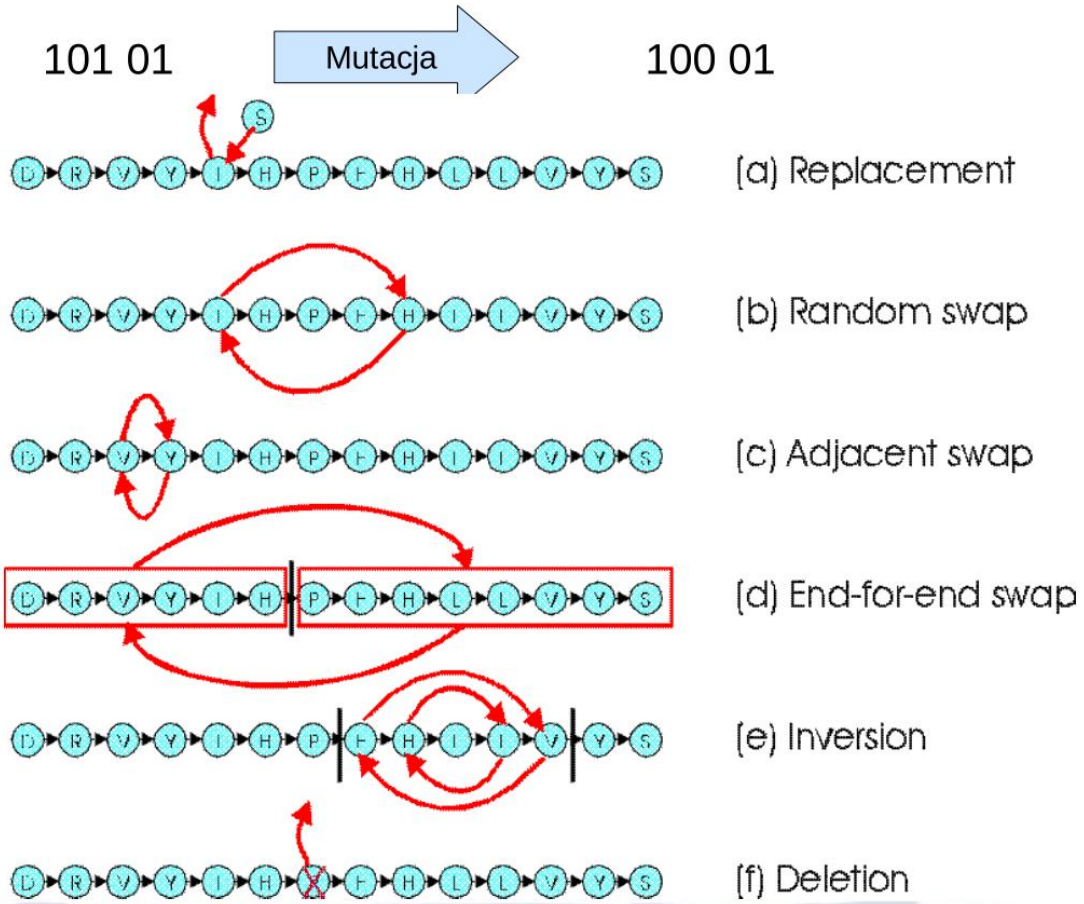
Krzyżowanie proste (jednopunktowe)

W pierwszej fazie kojarzymy w sposób losowy ciągi z puli rodzicielskiej w pary. Każda para przechodzi proces krzyżowania: wybieramy w sposób losowy pozycję k na ciągu kodowym (spośród l-1 pozycji, gdzie długość ciągu), następnie zamieniamy miejscami wszystkie znaki od pozycji k+1 do l włącznie w obu elementach pary.

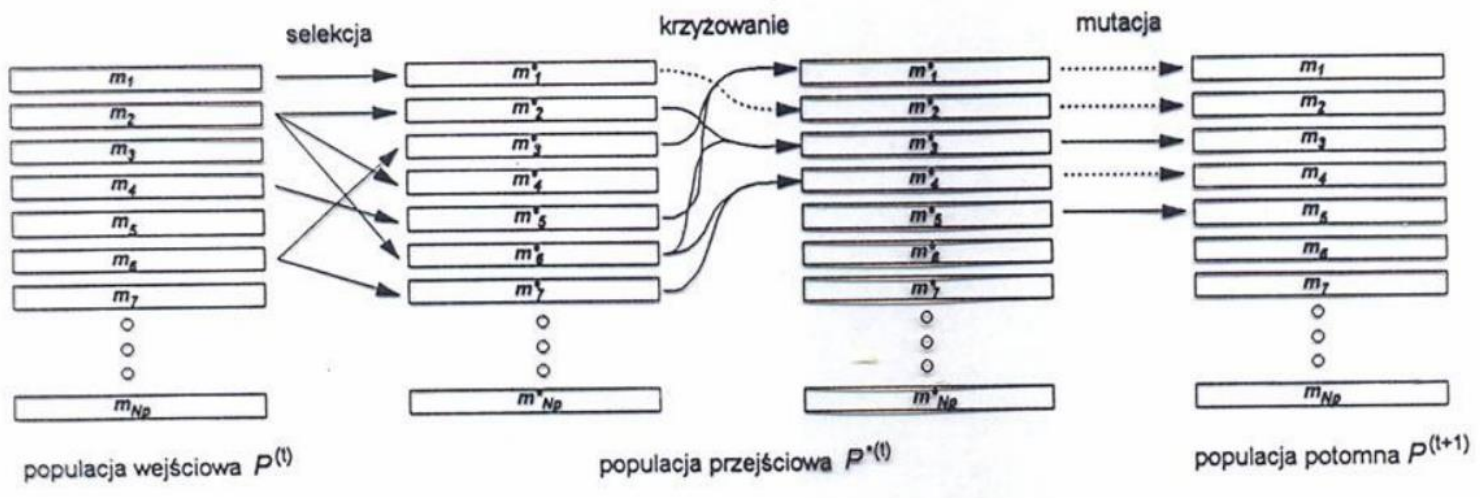
W wyniku krzyżowania mogą powstać nowe ciągi wchodzące w skład nowego pokolenia!

Występuje również inne typy krzyżowań: krzyżowanie dwupunktowe, krzyżowanie wielopunktowe.

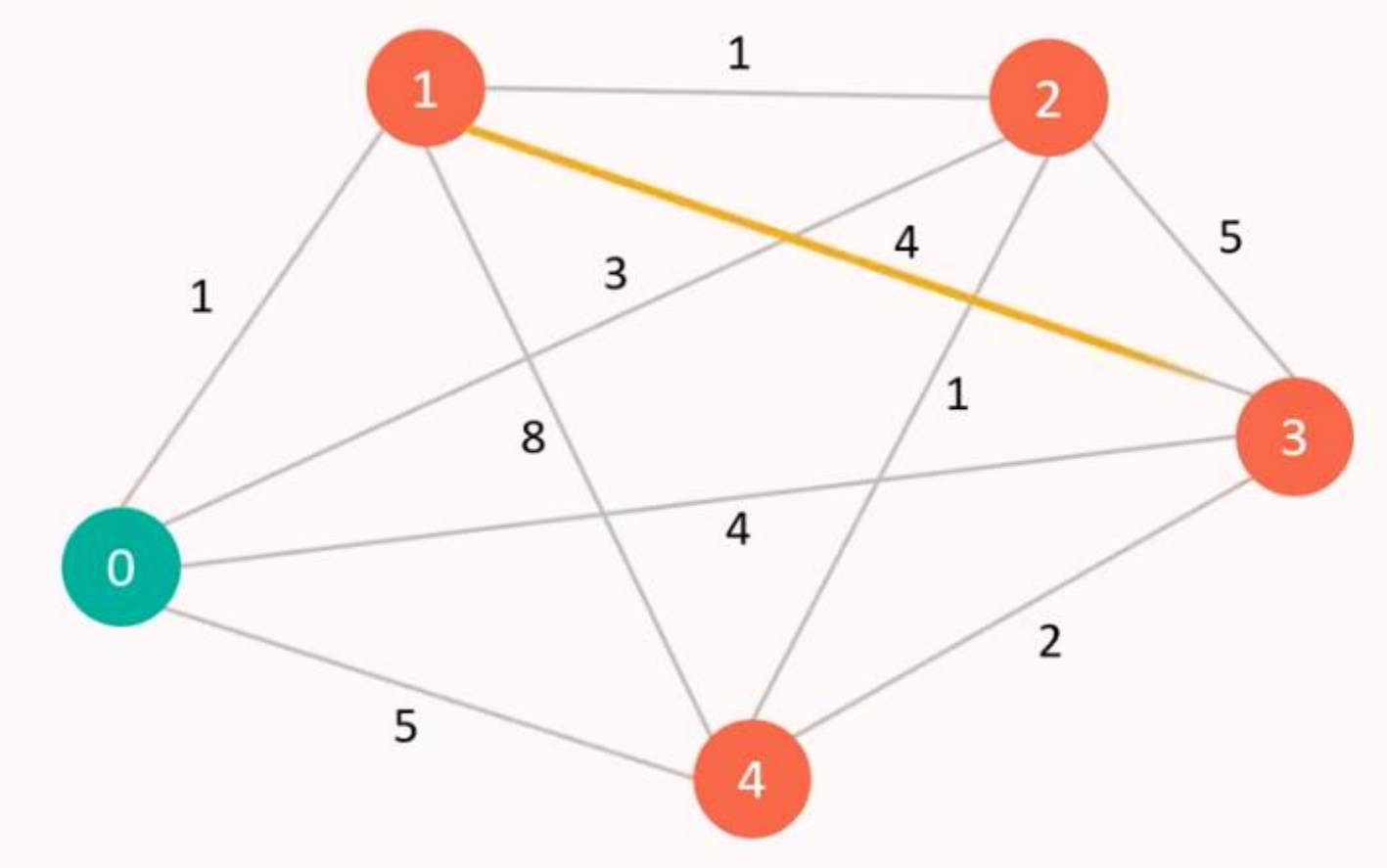
Mutacja


101 01
Mutacja
100 01
(a) Replacement
(b) Random swap
(c) Adjacent swap
(d) End-for-end swap
(e) Inversion
(f) Deletion

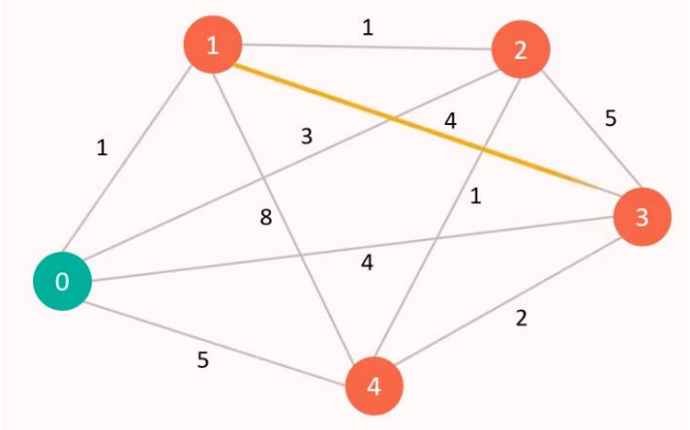
Schemat tworzenia kolejnych populacji



Zagadnienie komiwojażera



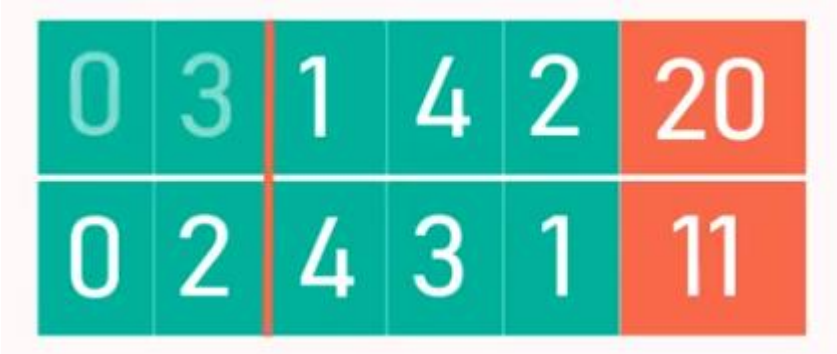
Populacja 1



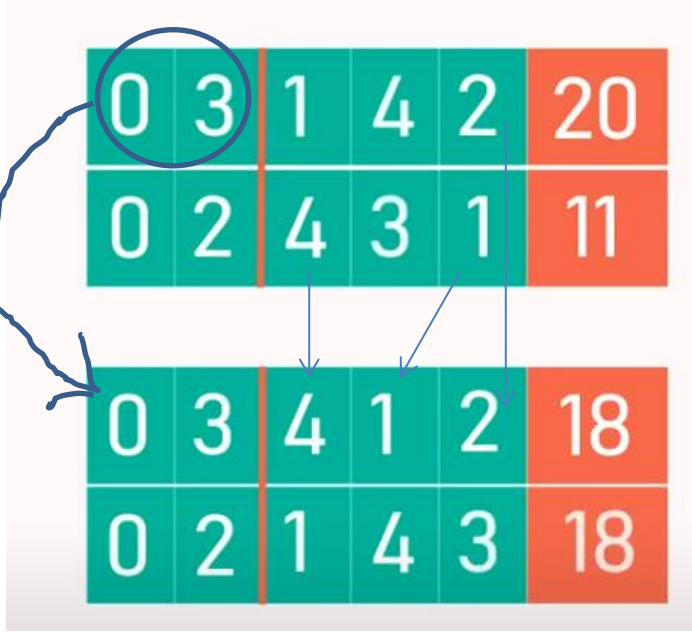
0	1	2	3	4	14
0	1	3	2	4	16
0	3	1	4	2	20
0	2	4	3	1	11
0	2	1	3	4	15
0	3	2	1	4	25
0	3	2	4	1	19

Para chromosom do krzyżowania

0	3	1	4	2	20
0	2	4	3	1	11



Krzyżowanie



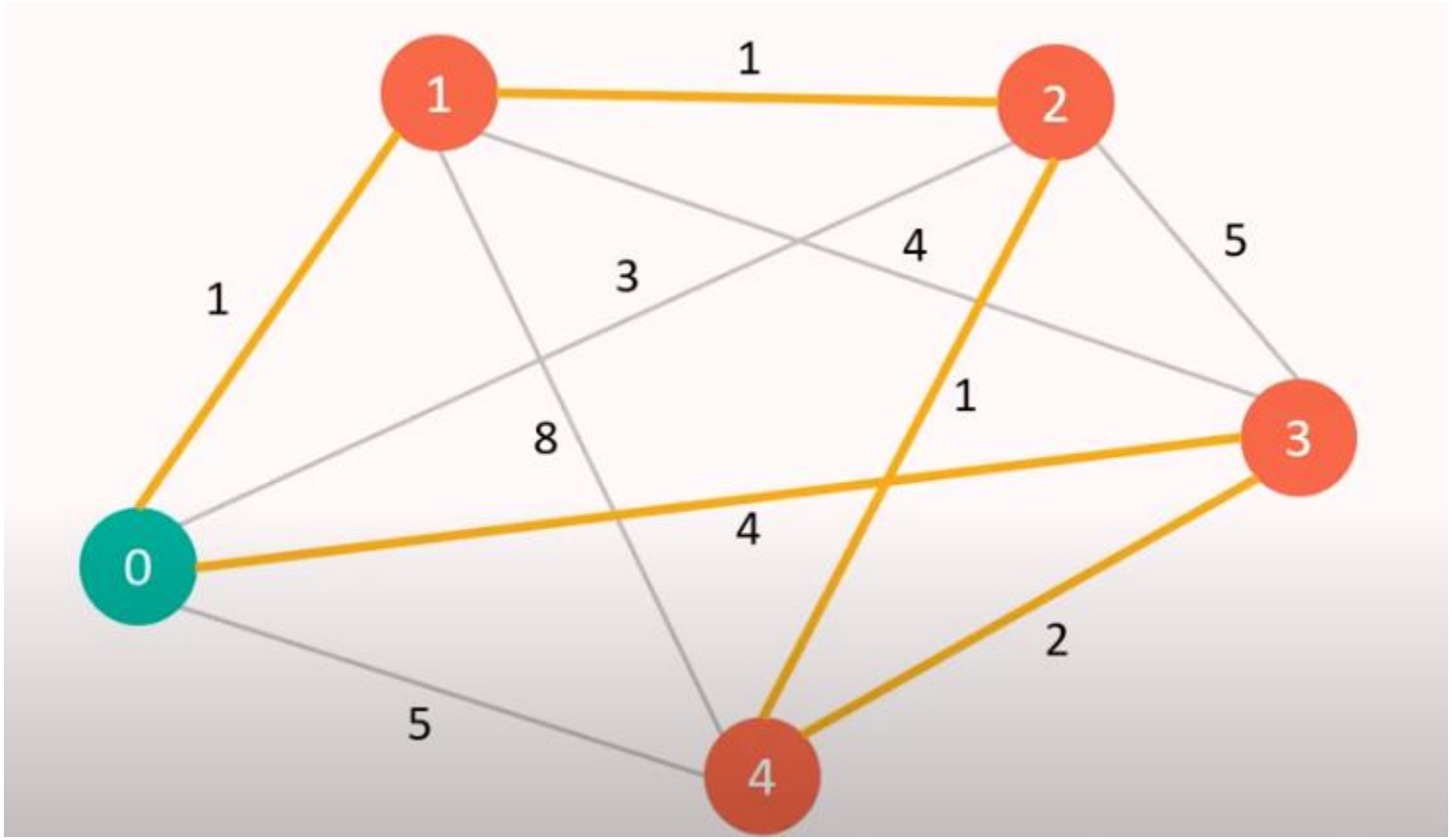
Mutacja

Generujemy losową liczbę L od 0 do 100
Jeżeli L<P – procent mutacji to robimy mutację:
wybieramy dwa losowych genu i wymieniamy je

Nowa populacja

0	2	4	3	1	11
0	1	2	3	4	14
0	2	1	3	4	15
0	1	3	2	4	16
0	2	1	4	3	18
0	1	4	3	2	19
0	3	2	4	1	19
0	3	1	4	2	20
0	3	2	1	4	25

Optymalna ścieżka



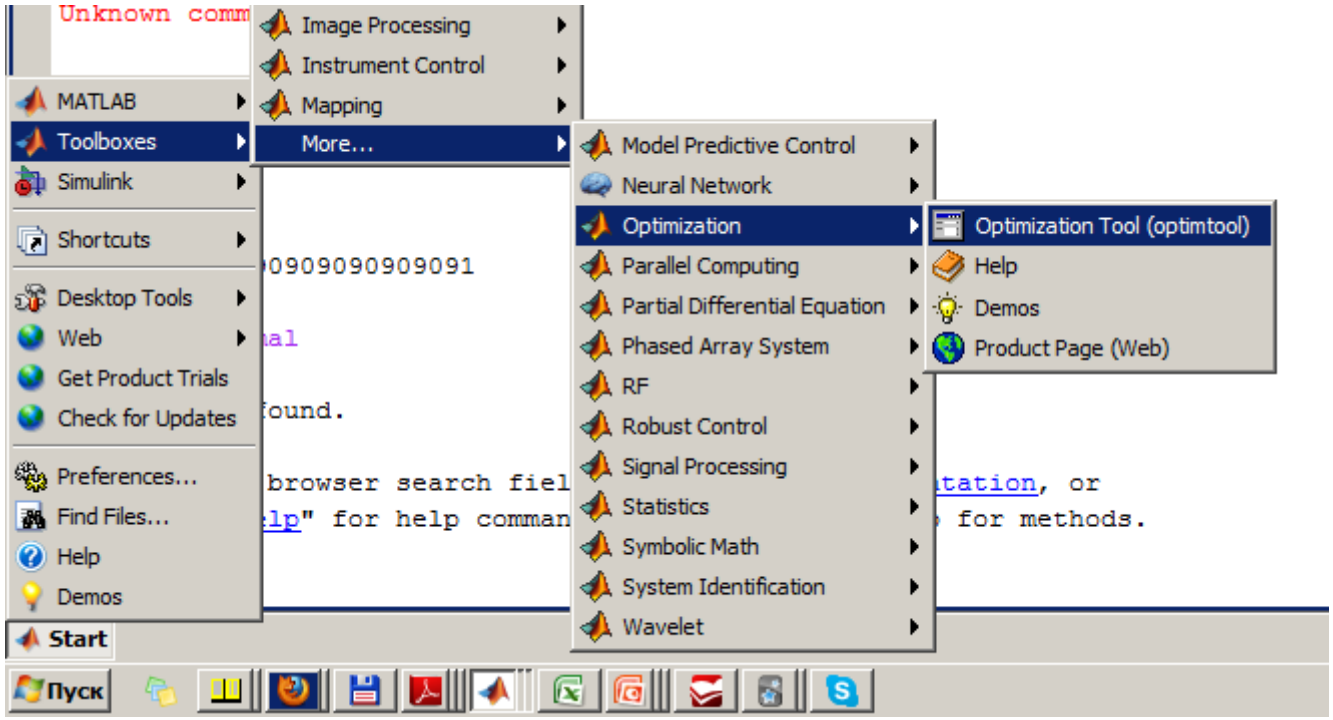
GA w MatLabie

$$f(x_1, x_2) = x_1^2 - 2x_1x_2 + 6x_1 + x_2^2 - 6x_2$$

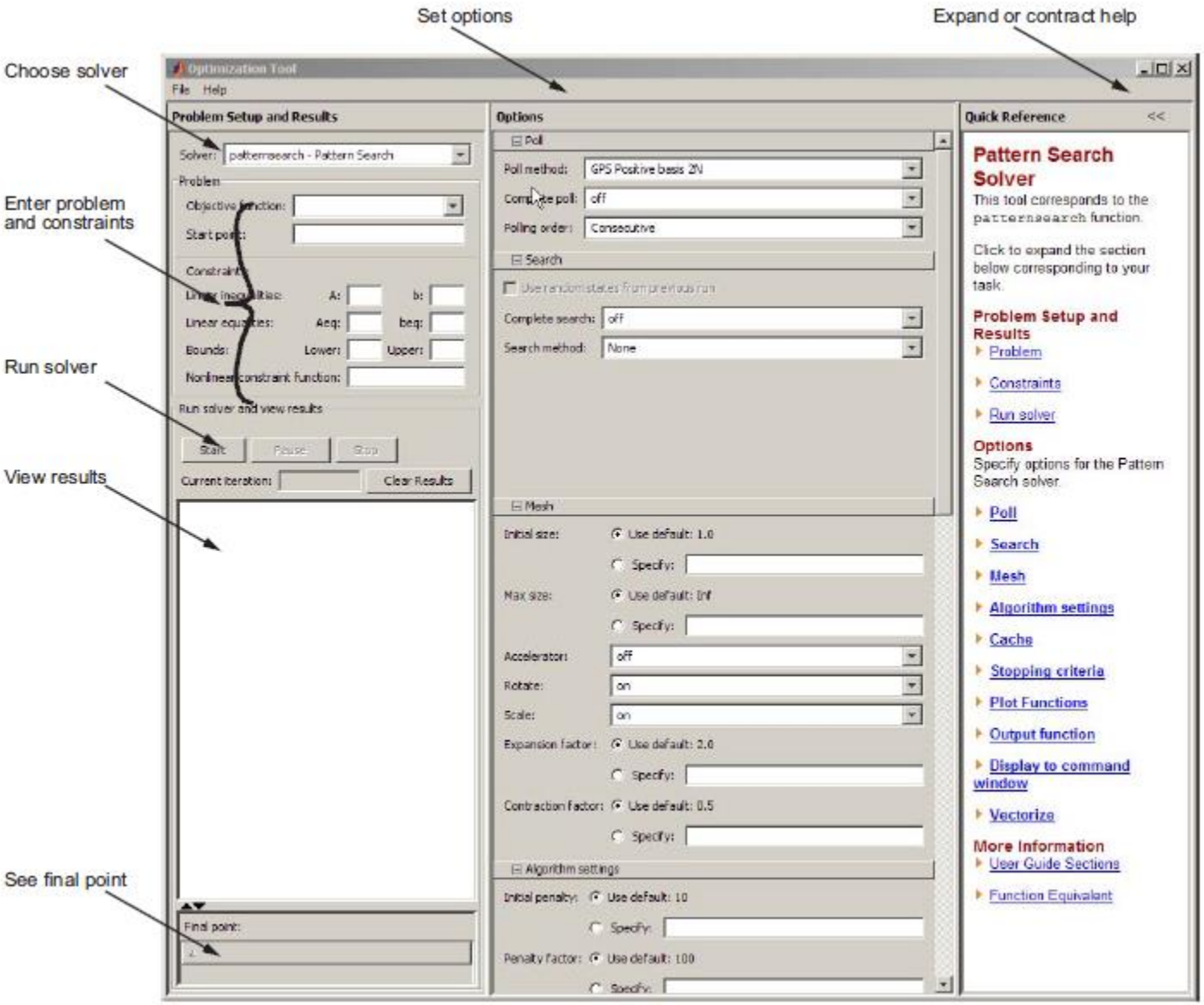
```
function z = my_fun(x)
z = x(1)^2 - 2*x(1)*x(2) + 6*x(1) + x(2)^2 - 6*x(2);
```

```
my_fun([2 3])

ans =

     -5
```

Optymization tool

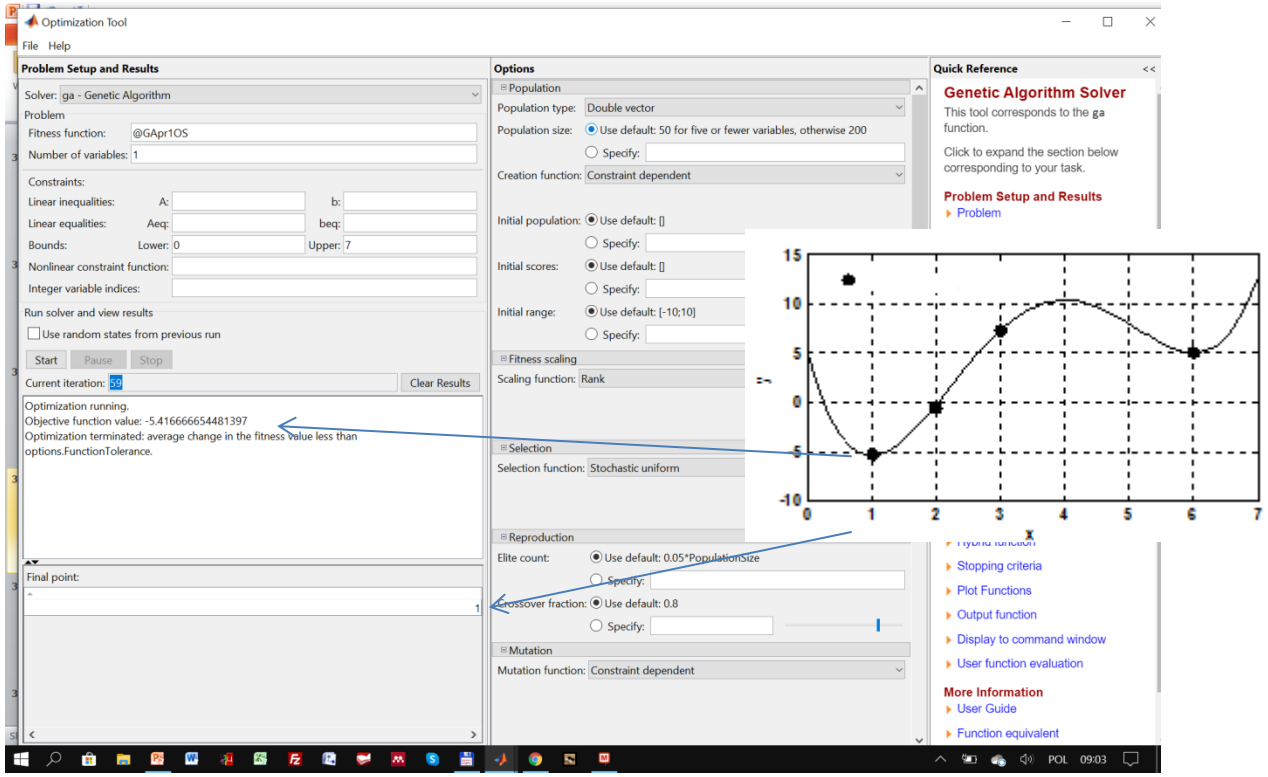


Główne okienko

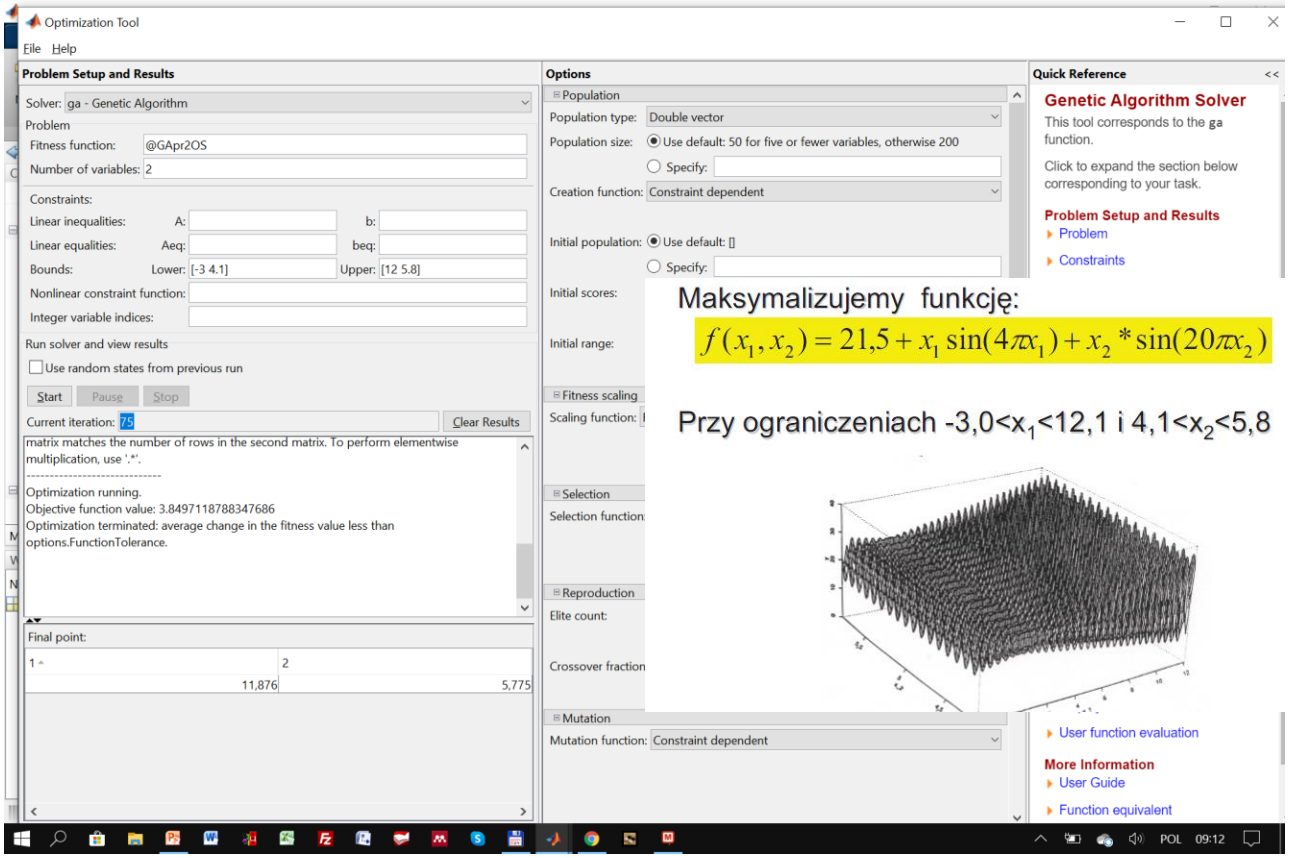


Optimization tool

Przykład 1

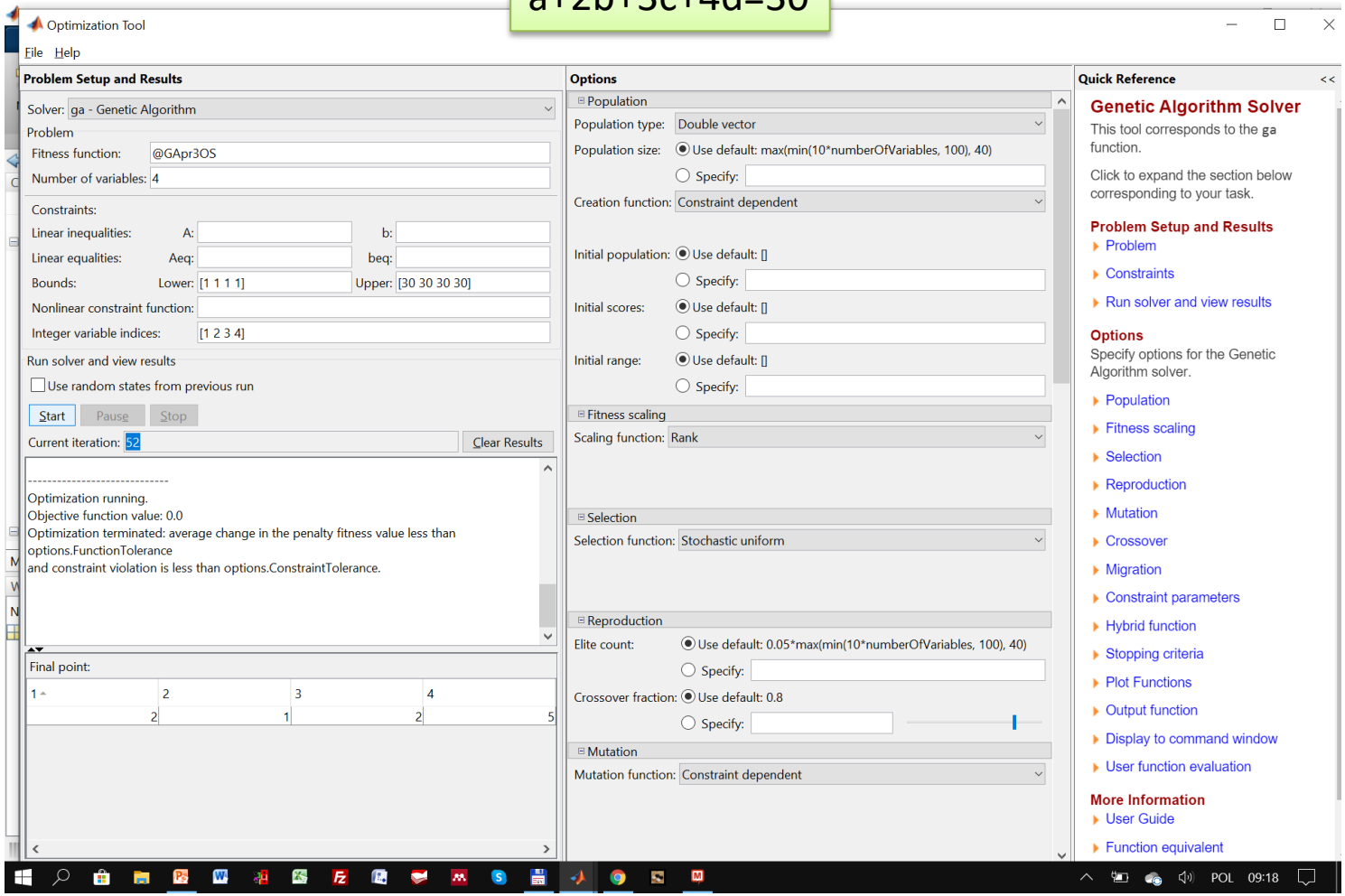


Optimization tool (cd)



Optimization tool (cd)

a+2b+3c+4d=30



Funkcje optymalizacji

Funkcja do optymalizacji $F\ (x, y) = 100(y - x^2)^2 + (1 - x)^2$

ograniczenia: $x, y \in [-100, 100]$

Dla funkcji można znaleźć minimum oraz argumenty minimalizujące za pomocą funkcji **fmincon(@Fun,x0,[],[],[],[],lb,ub,[],options)**
gdzie: Fun - nazwa funkcji, x0 - **wektor początkowy**, lb, ub - ograniczenia odpowiednio dolne i górne, options - opcje optymalizacji
Przykład:

```
>> x0 = [32,-20];
>> lb = [-100,-100];
>> ub = [100,100];
>> options = optimset('LargeScale','off');
>> [x,fval] = fmincon(@FunOptym,x0,[],[],[],[],lb,ub,[],options)
```

Dokonać optymalizacji podanych funkcji w Matlab'ie można za pomocą Algorytmów Genetycznych(gatool)

Funkcja GA

[x fval] = ga(@fitnessfun, nvars, options)

where

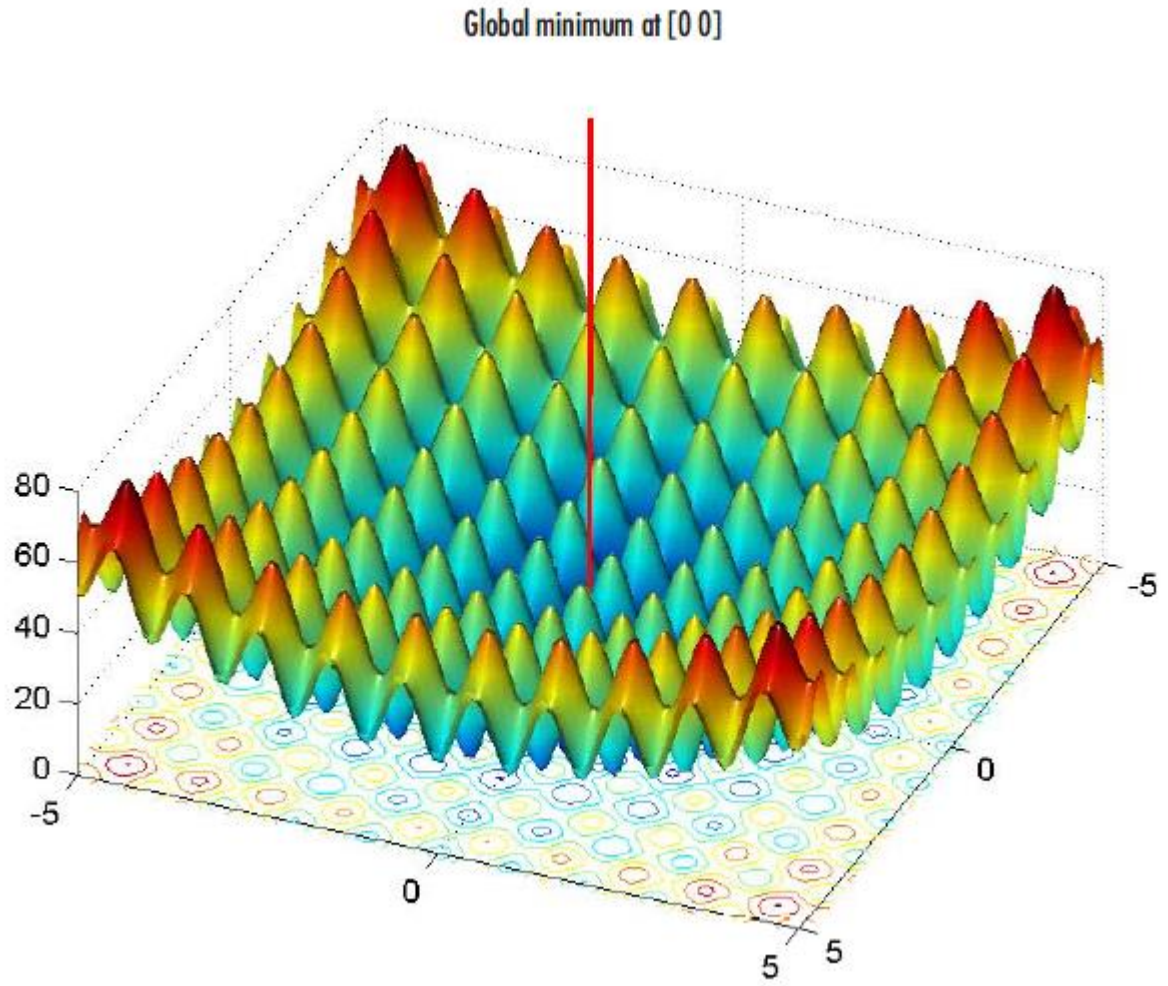
- **@fitnessfun** is a handle to the fitness function.
- **nvars** is the number of independent variables for the fitness function.
- **options** is a structure containing options for the genetic algorithm. If you do not pass in this argument, ga uses its default options.

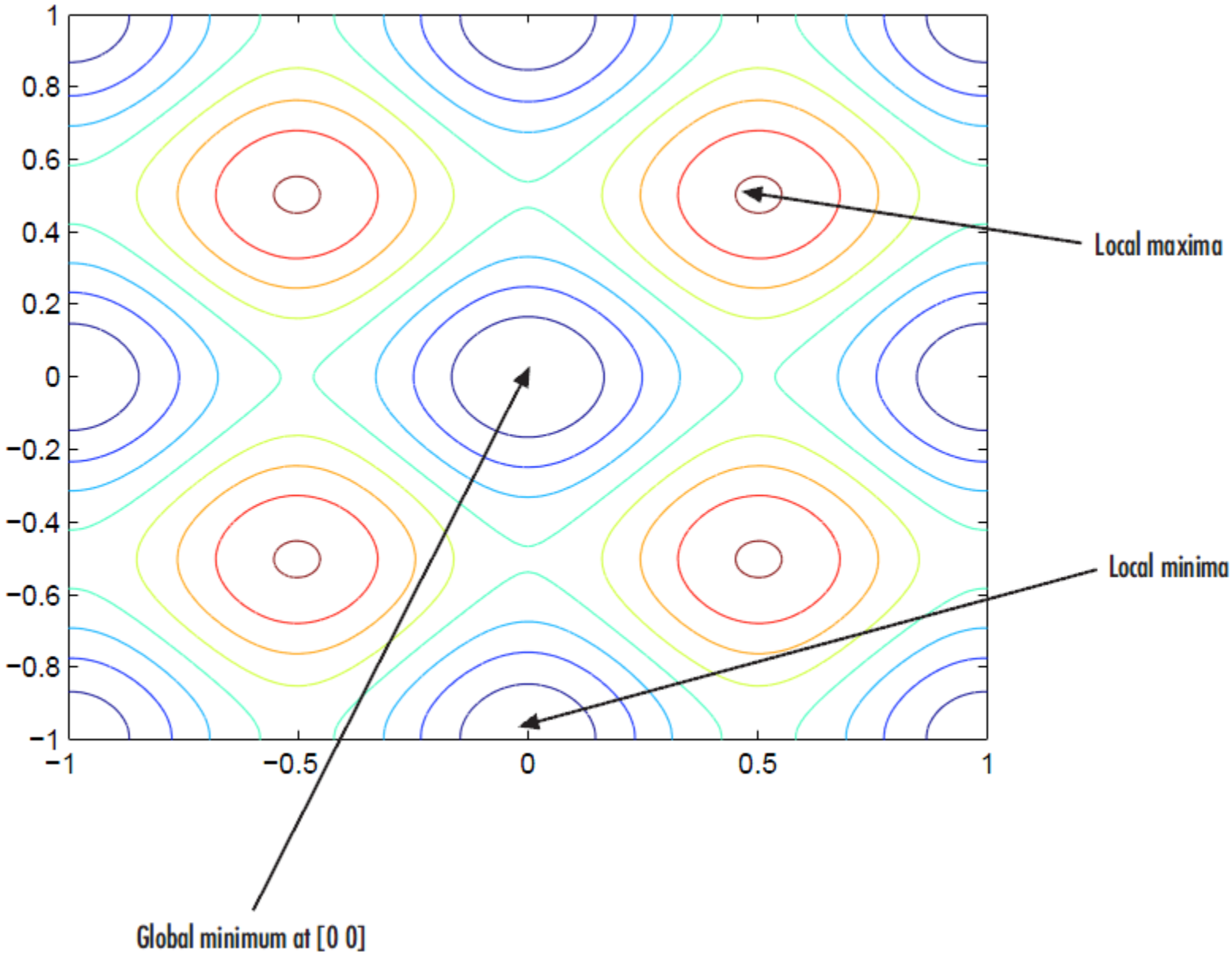
The results are given by

- **x —** Point at which the final value is attained
- **fval —** Final value of the fitness function

Funkcja Rastrigina

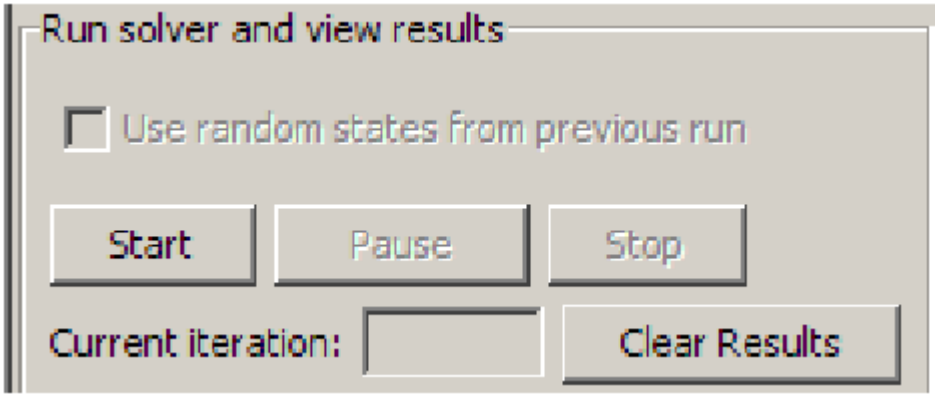
$$Ras(x) = 20 + x_1^2 + x_2^2 - 10\left(\cos 2\pi x_1 + \cos 2\pi x_2\right).$$



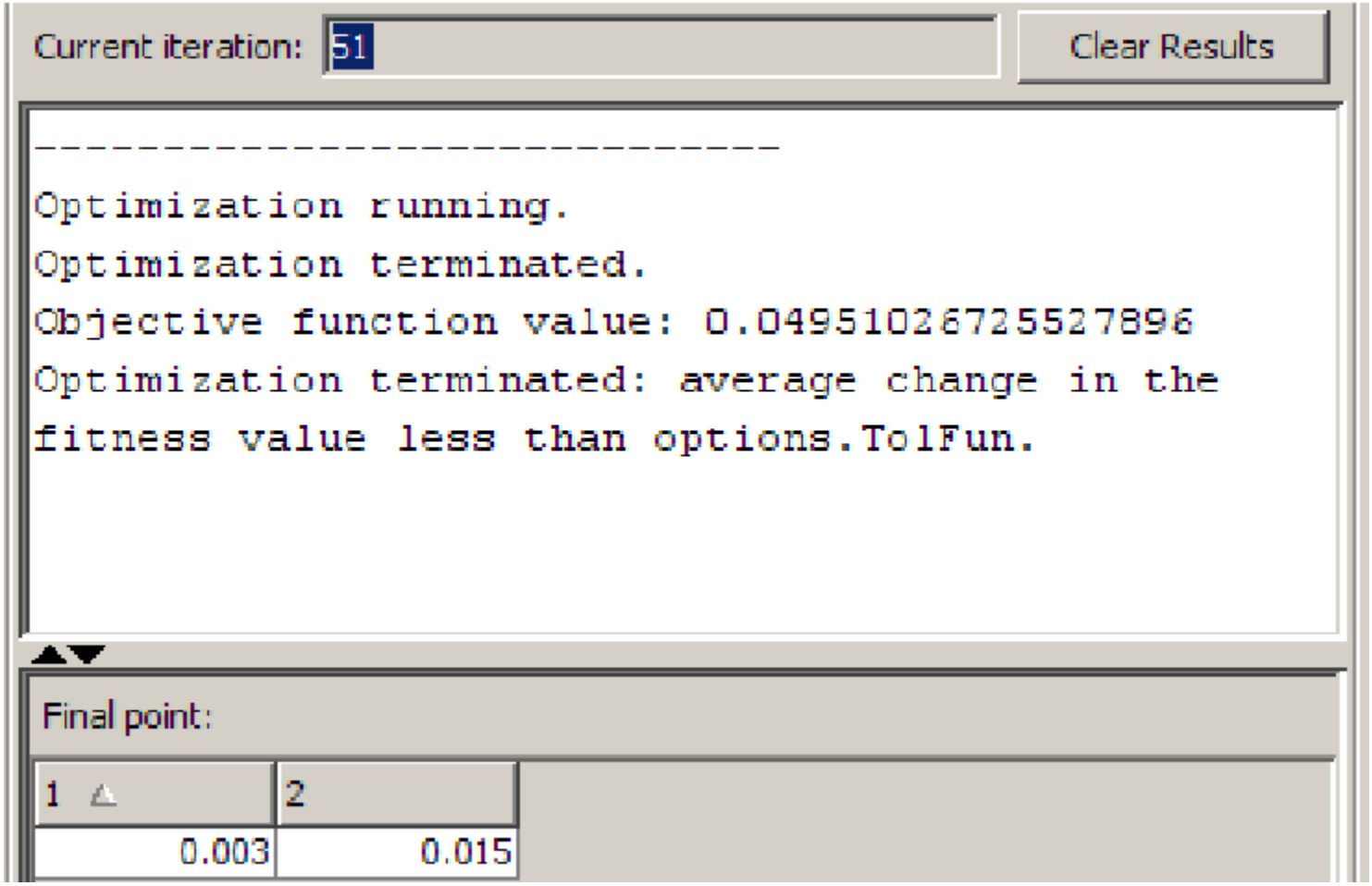

1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
−1
−0.5
0
0.5
1
Local maxima
Local minima
Global minimum at [0 0]

Funkcja


Fitness function:
@rastriginsfcn
Number of variables:
2

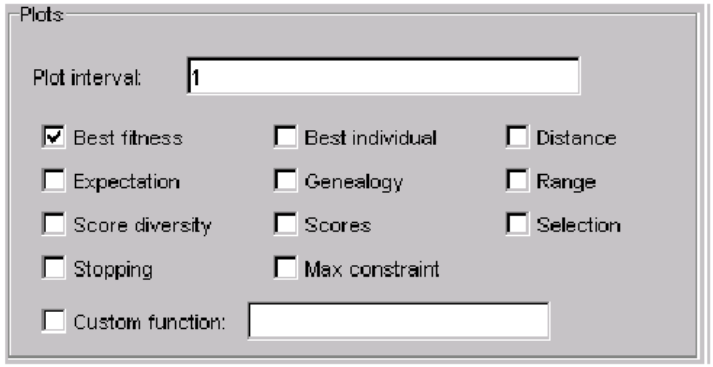

Run solver and view results
Use random states from previous run
Start
Pause
Stop
Current iteration:
Clear Results

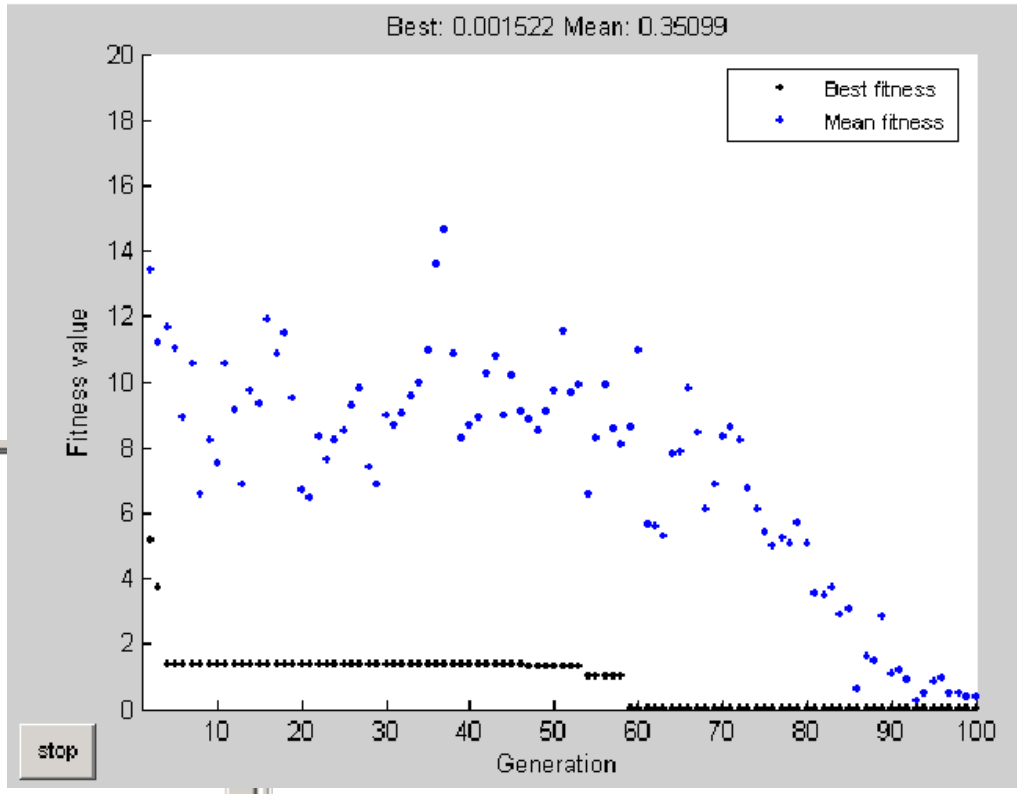
Iteracje

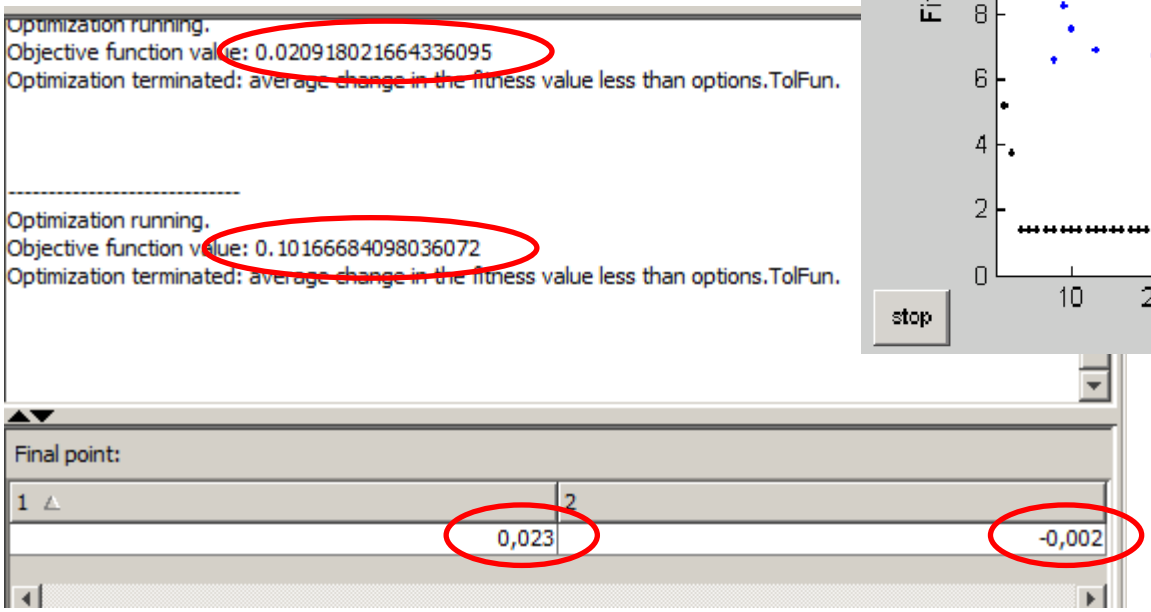

Current iteration: 51
Clear Results

Optimization running.
Optimization terminated.
Objective function value: 0.0495102672552789 6
Optimization terminated: average change in the fitness value less than options.TolFun.
Final point:
1
2
0.003
0.015

Wizualizacja







Kodowanie binarne

Kodowanie pozycyjne:

$$x = \sum_{i=1}^{l} a_i \cdot 2^{i-1}$$

gdzie *x* - parametr rozwiązania,

a_i - element ciągu kodowego $(a_1, a_2, \mathrm{K}, a_l)$

Kodowanie Graya:

$$b_i = \bigoplus_{j=1}^{i} a_j$$

Kod pozycyjny
Przesuwamy w prawo →

```
10110
01011
----- XOR
11101
```

liczba	kod pozycyjny	kod Graya
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

Algorytm genetyczny w nastrojeniu sterownika

Modele procesu

Testowanie metody i ocena jej jakości regulacji przeprowadzono na dwóch różnych modelach. Do testów wybrano modele, dla których znane są nastawy szeroko opisywane w literaturze światowej i istnieje spora ich ilość. Dla procesów o charakterystyce statycznej liniowej wybrano model o transmitancji:

$$G_{MI} = \frac{K_m e^{-s\tau_m}}{(1+sT_m)}, \quad (1)$$

gdzie: K_m – wzmocnienie statyczne obiektu, τ_m – opóźnienie transportowe, T_m – zastępcza stała czasowa obiektu.

Można go interpretować jako połączenie inercji pierwszego rzędu z opóźnieniem. Dla procesów o charakterystyce statycznej nieliniowej wybrano model opisany transmitancją:

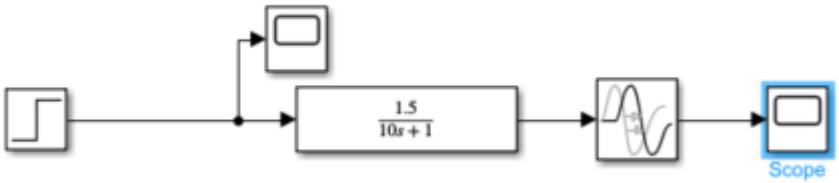
$$G_{MII} = \frac{K_m e^{-s\tau_m}}{s(1+sT_m)}, \quad (2)$$

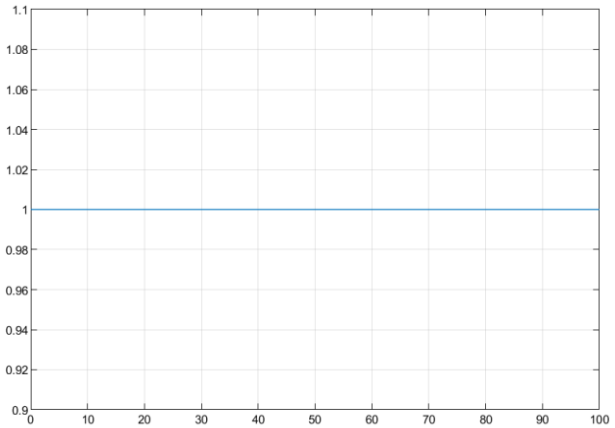
gdzie: K_m – wzmocnienie statyczne obiektu, τ_m – opóźnienie transportowe, T_m – zastępcza stała czasowa obiektu.

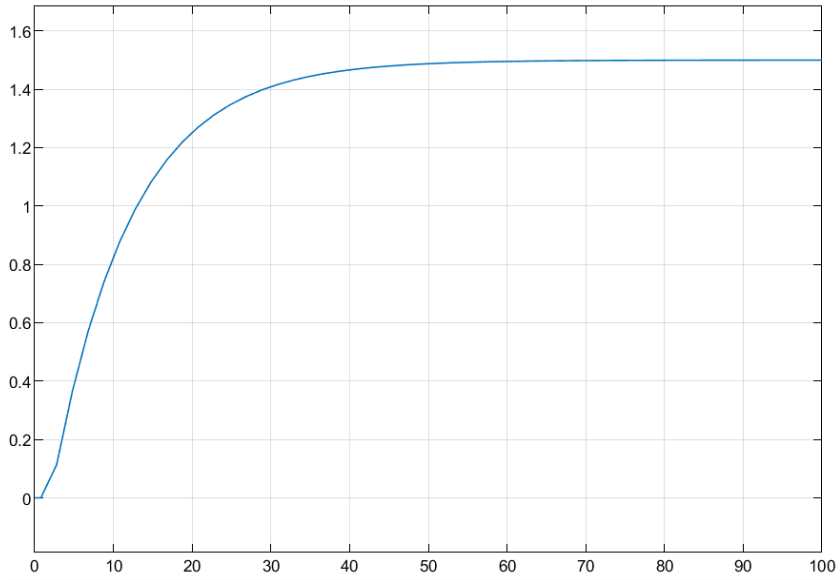
Jest on syntezą członu inercyjnego, całkującego oraz opóźnienia transportowego.

$$K_m = 1{,}5;\ \tau_m = 2\ \text{s};\ T_m = 10\ \text{s}.$$

https://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-article-BSW4-0120-0005/c/Salat.pdf







GAexpWTMO04.slx

PID sterownik

Compensator formula

$$P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}$$

Main | Initialization | Output Saturation | Data Types | State Attributes

Controller parameters

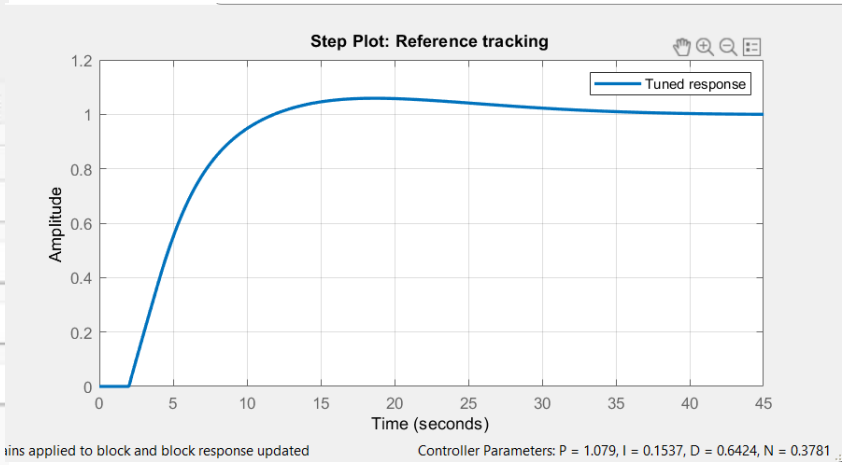
Source: internal

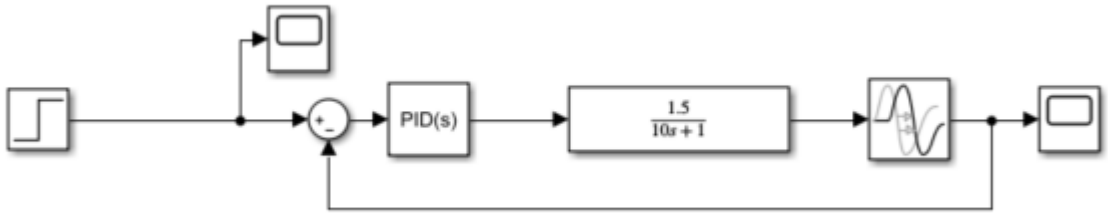
Proportional (P): 1.07936483745769

Integral (I): 0.153715088509939

Derivative (D): 0.642364248235642

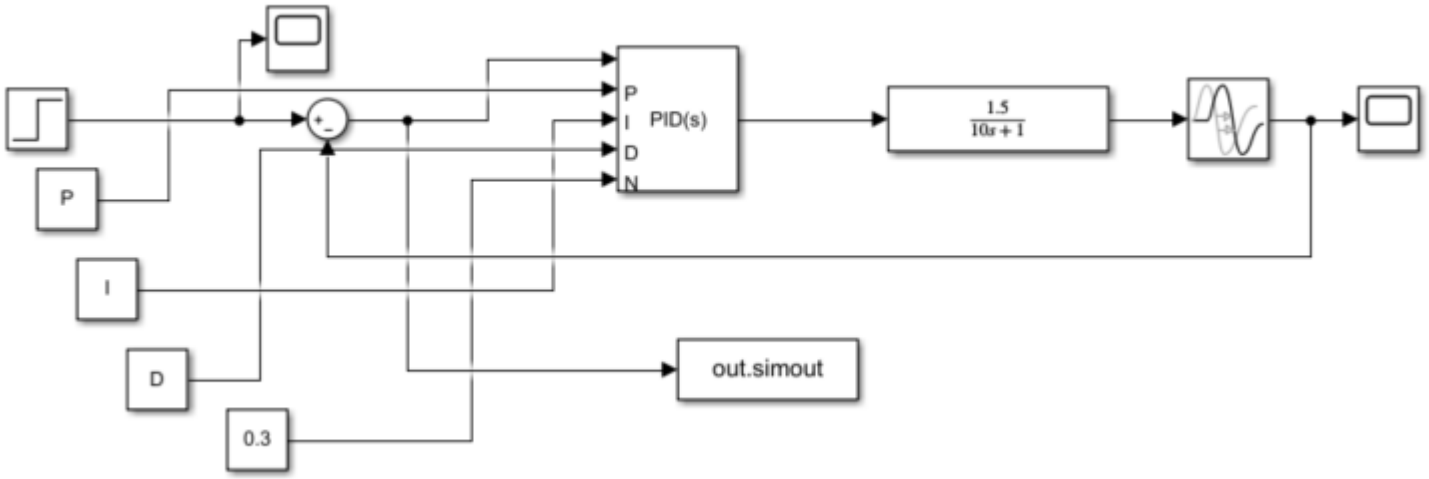
Use filtered derivative





GAexpPIDWTMO04.slx

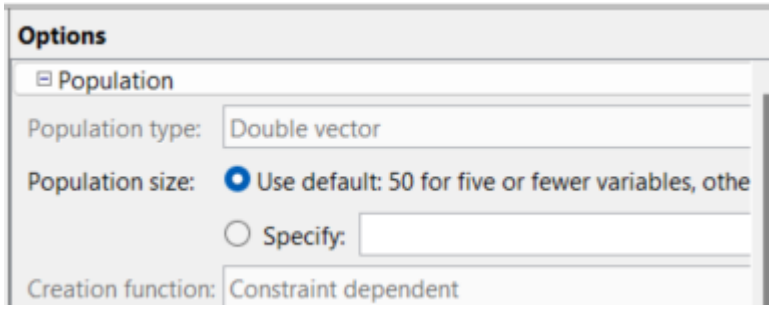
Model do GA

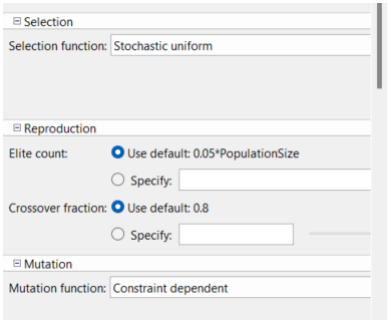


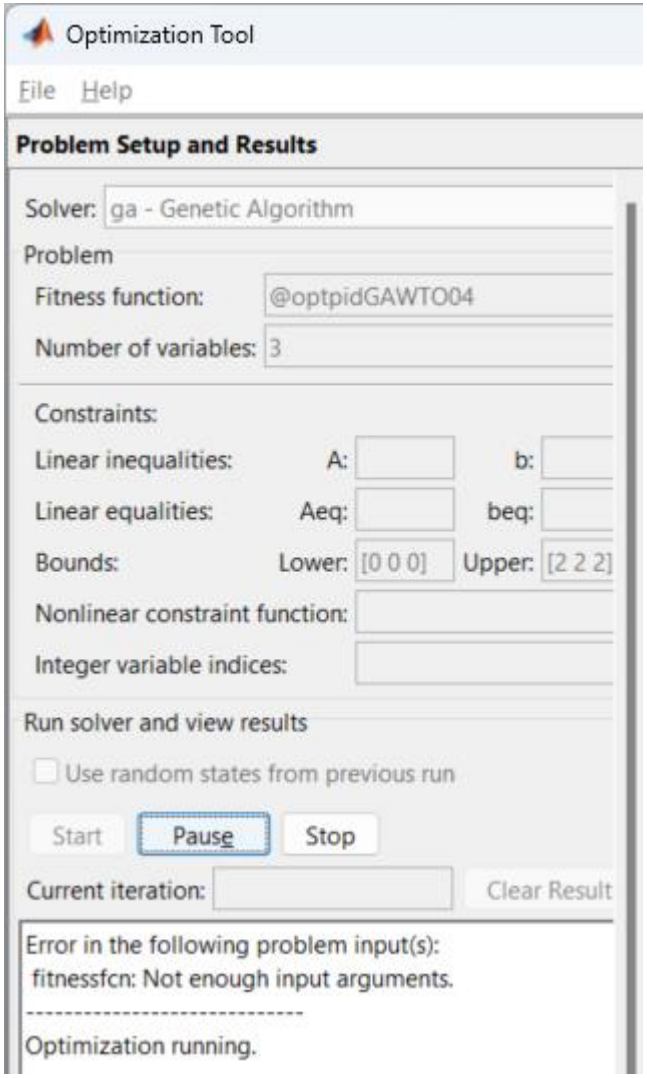
GAexpPIDGAWTMO04.slx

Fintess function i optymalizacja GA

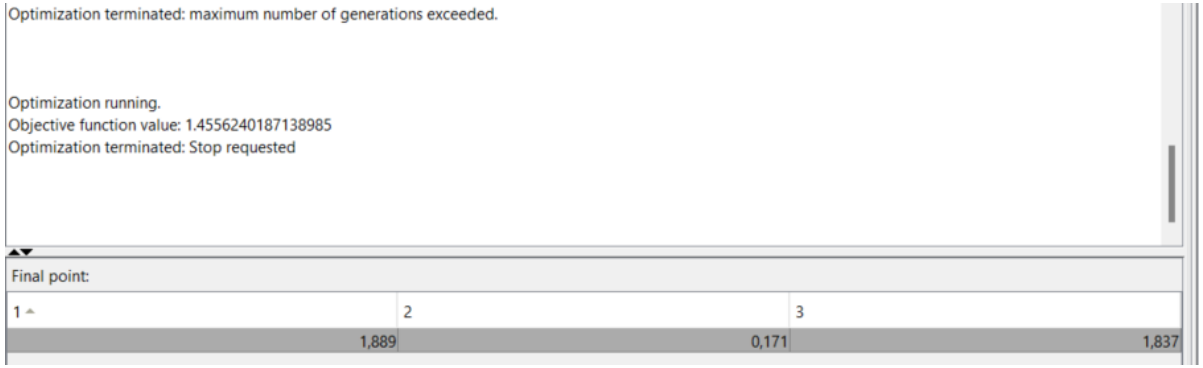
```
function F = optpidGAWTO04(pid)


      P = pid(1);
      I = pid(2);
      D = pid(3);

% {
     a1=1;a2=1;
      simopt = simset('solver','ode5',...
                      'SrcWorkspace','Current');
      [yout] = sim('GAexpPIDGAWTMO04.slx',[0 100],simopt);
      J=yout.simout.data;
      F = norm(J-1);
      %F = norm(yout(10:51)-1);
%}
```



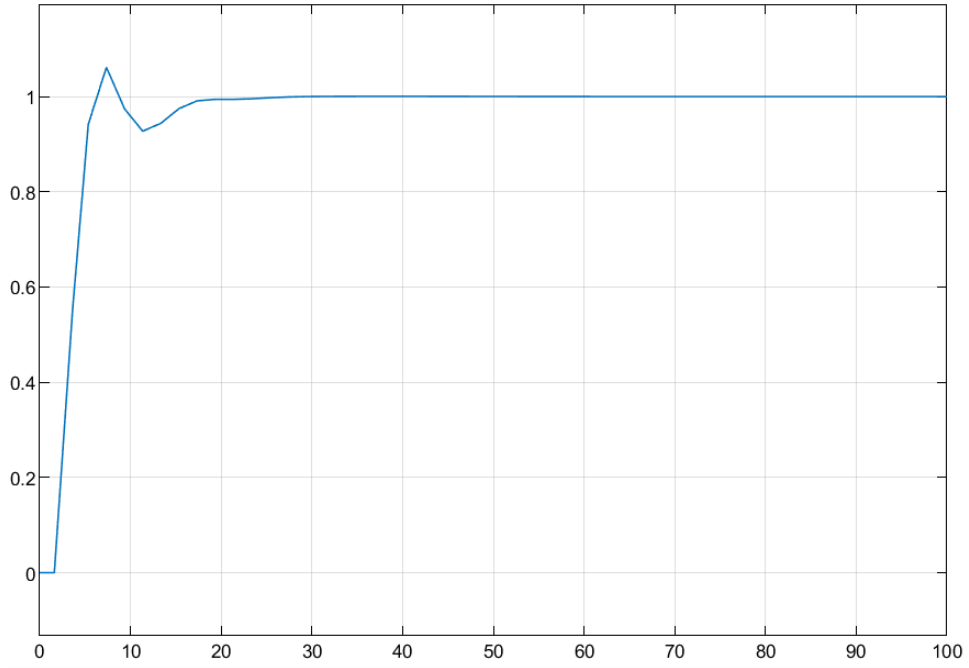




Rozwjązanie

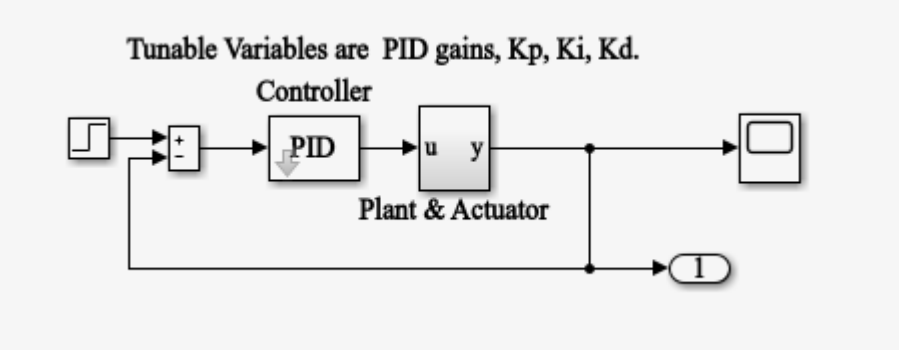


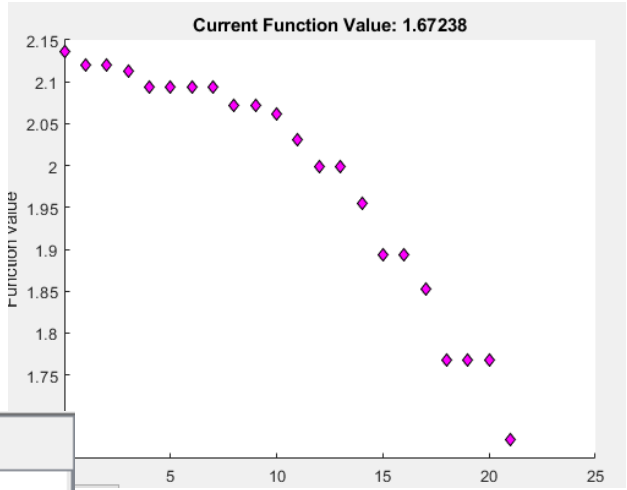
P=1.8893850279014521
I=0.17084802628037954
D=1.8372234359185138



Optymalizacja parametrów sterownika

```
function F = optpidGA(pid)


    Kp = pid(1);
    Ki = pid(2);
    Kd = pid(3);

% {
   a1=1;a2=1;
    simopt = simset('solver','ode5',...
                'SrcWorkspace','Current');
    [tout,xout,yout] = sim('optsim',[0
100],simopt);
    F = norm(yout-1);
    %F = norm(yout(10:51)-1);
%}
```





Final point:

1	2	3
2,586	-6,794	4,961

Regulator PI

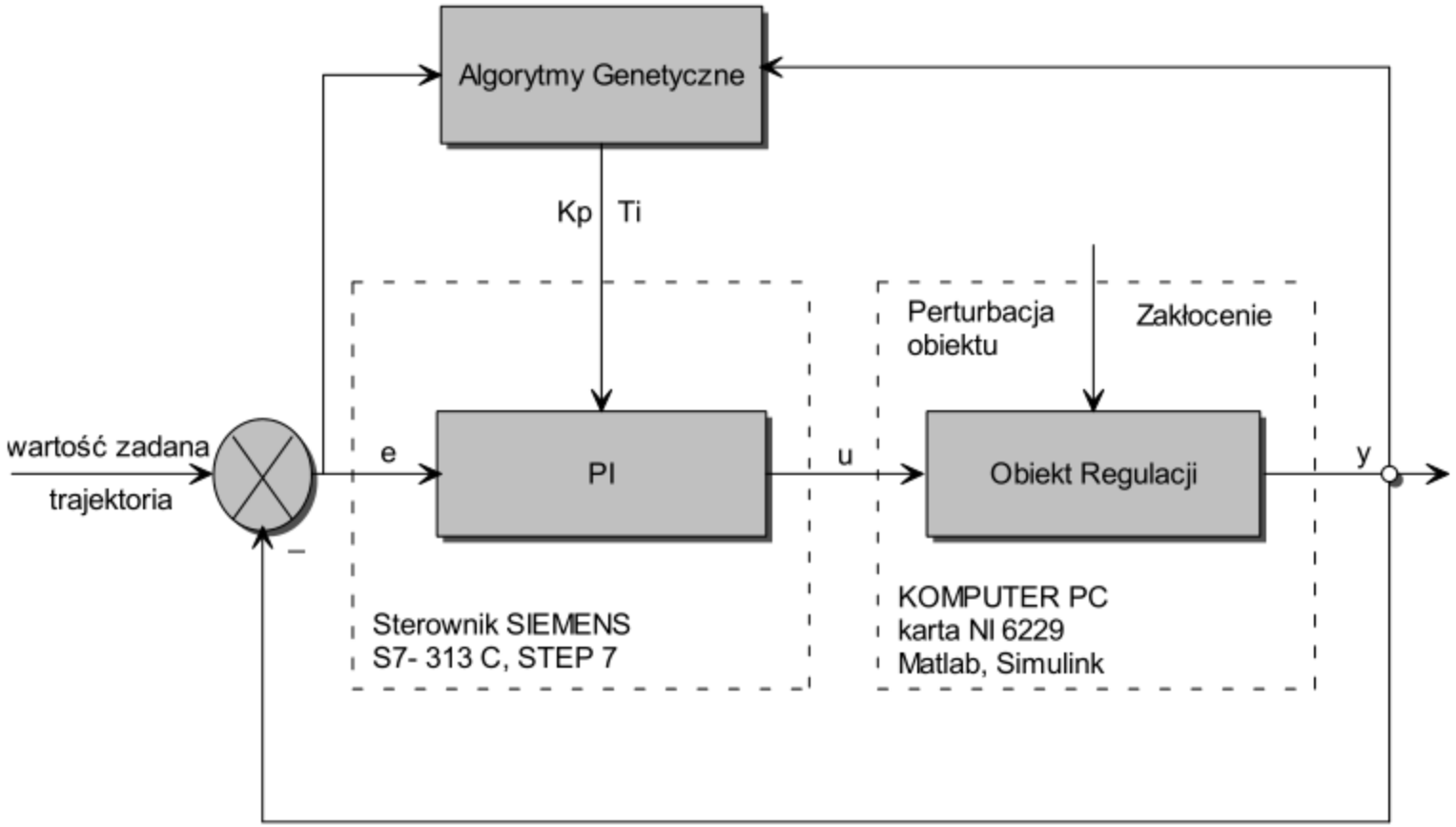
$$PI(s) = k_p\left(1 + \frac{1}{T_i s}\right), \qquad (3)$$

Optymalne parametry są wyszukiwane dopiero wtedy, gdy odpowiedź układu jest jak najbliższa wartości zadanej (wg całkowych wskaźników jakości) co w algorytmach genetycznych sprowadza się do optymalizacji funkcji celu zdefiniowanej jako:

$$J(K_p, T_i) = \sum_{t=0}^{tsim} \left| y_r(t) - y(t) \right|, \qquad (4)$$

gdzie: $y_r(t)$ - chwilowa wartość zadana, $y(t)$ - chwilowa wartość wyjściowa obiektu.

Ogólny schemat działania algorytmu



Parametry GA

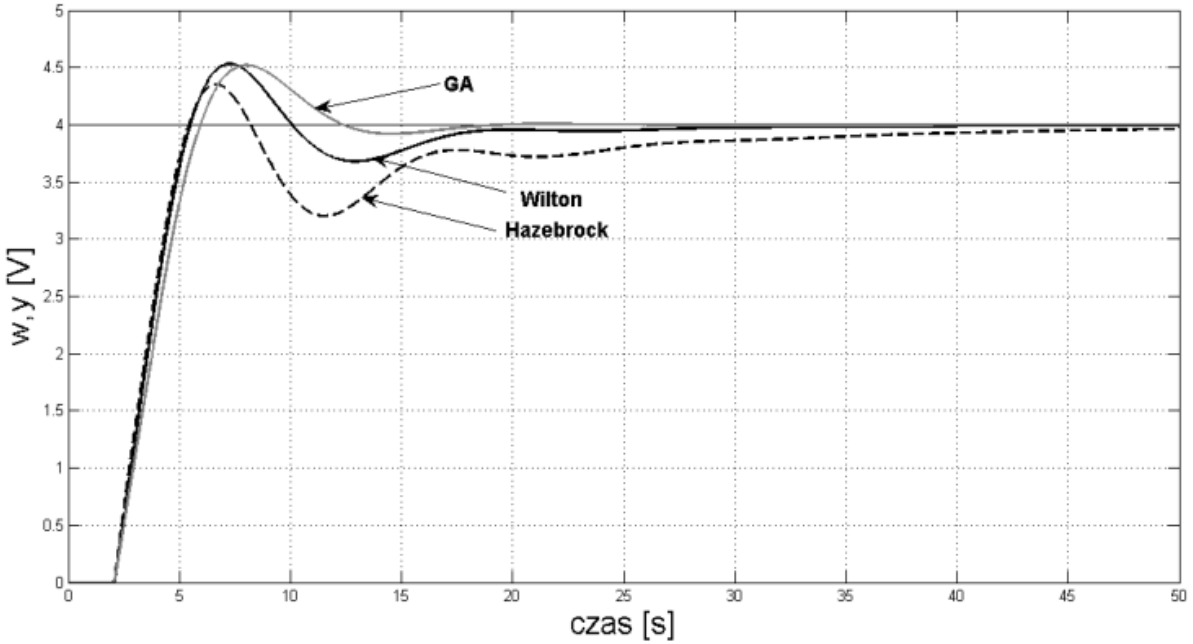
Dobór nastaw regulatora PID został przeprowadzony za pomocą następujących parametrów algorytmu genetycznego:

- Rozmiar populacji 30 osobników
- Liczba generacji 100
- Selekcja metodą turniejową
- Prawdopodobieństwo krzyżowania p_k=0,77
- Prawdopodobieństwo mutacji p_m=0,077

Wyniki – nastawy optymalne

		Model I K_p | T_i [s]	**Model II** K_p | T_i [s]
AG	skok	1,97 | 10,33	0,086 | 78,63
	trajektoria	3,44 | 8,28	0,161 | 73,84
	zakłócenia	1,97 | 10,33	0,086 | 78,63
	perturbacje	1,97 | 10,27	0,066 | 71,79
Wilton		2,25 | 10	-
Hazebrock & Van der Waerder		2,67 | 14,28	-
Shinskey		-	0,053 | 48
Poulin & Pomerleau		-	0,031 | 60,87

Przebieg regulacji dla różnych nastaw przy skokowym wymuszeniu dla modelu I



Przebieg regulacji dla różnych nastaw przy skokowym wymuszeniu dla modelu II