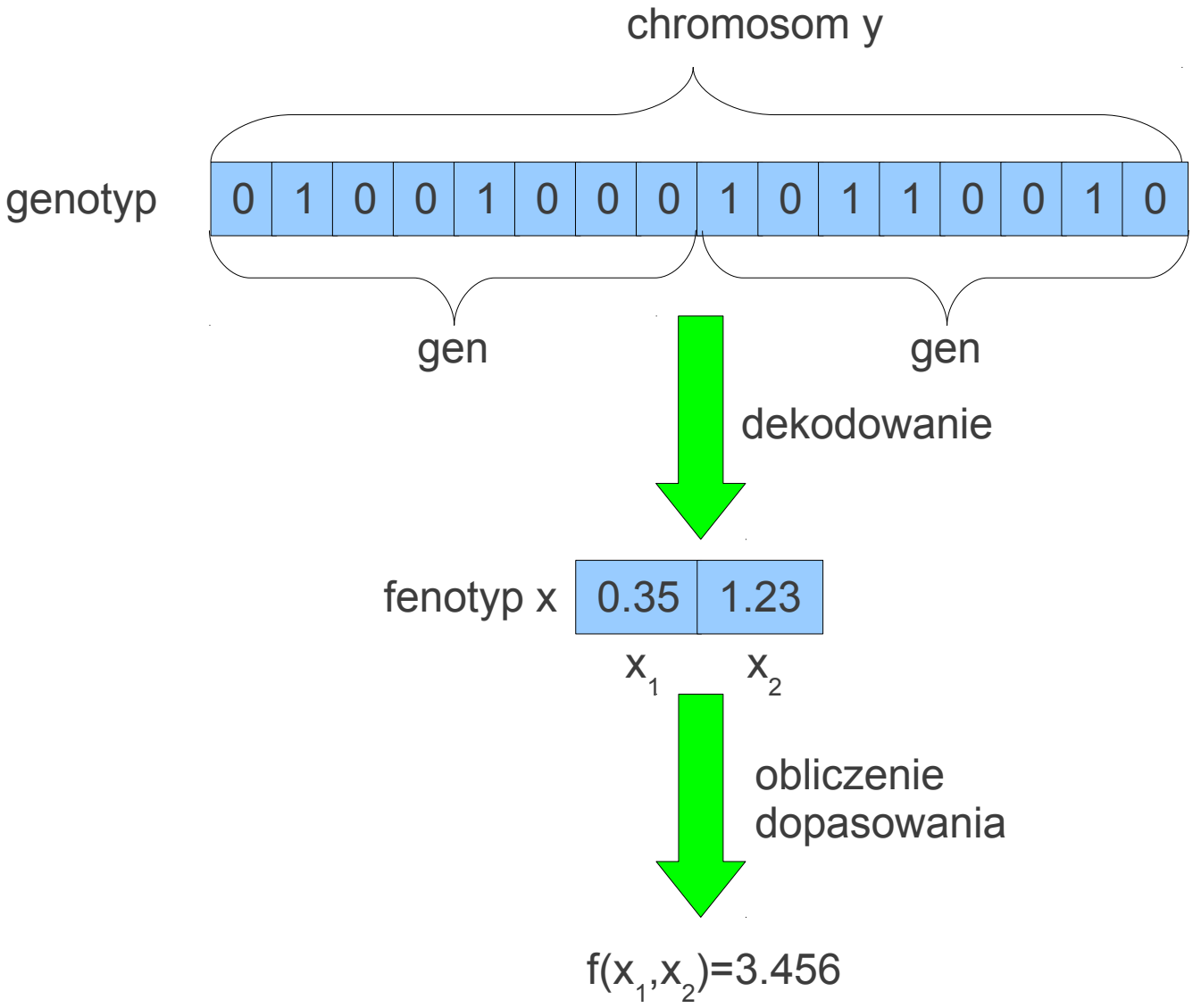
Standardowy algorytm genetyczny

Szybki przegląd

- Opracowany w USA w latach 70.
- Wcześni badacze:
  - John H. Holland. Autor monografii „Adaptation in Natural and Artificial Systems”, wydanej w 1975 r., (teoria schematów próbująca wyjaśnić zasadę działania algorytmu).
  - Kenneth A. de Jong, doktorant Hollanda, w swojej rozprawie doktorskiej sfinalizowanej w 1975 r. zastosował algorytmy genetyczne do optymalizacji funkcji.
  - David E. Goldberg, doktorant Hollanda, autor książki „Genetic Algorithms in Search, Optimization, and Machine Learning” wydanej w 1989 r.
- Algorytm opracowany przez Hollanda będzie nazywany przez nas Standardowym Algorytmem Genetycznym (ang. Simple Genetic Algorithm - SGA). Cechy:
  - Rozwiązania w postaci ciągów binarnych.
  - Operator krzyżowania i mutacji.
  - Selekcja proporcjonalna metodą koła ruletki.

Osobnik w algorytmie genetycznym



Funkcja dopasowania

- Funkcja dopasowania pełni rolę środowiska oceniając potencjalne rozwiązania problemu.
- W prostym algorytmie genetycznym funkcja dopasowania f(y) jest odwzorowaniem ze zbioru ciągów binarnych o długości L w zbiór R+ (liczb rzeczywistych dodatnich).
- W teorii optymalizacji używa się pojęcia funkcji celu. Jeżeli w zadaniu funkcja celu przyjmuje wartości dodatnie i problem polega na jej maksymalizacji, to za funkcję dopasowania można przyjąć funkcję celu (oczywiście z uwzględnieniem kodowania binarnego przez chromosomy).
- Problem 1: chcemy znaleźć maksimum pewnej funkcji celu g(y), która przyjmuje wartości niekoniecznie dodatnie. Przekształcenie:

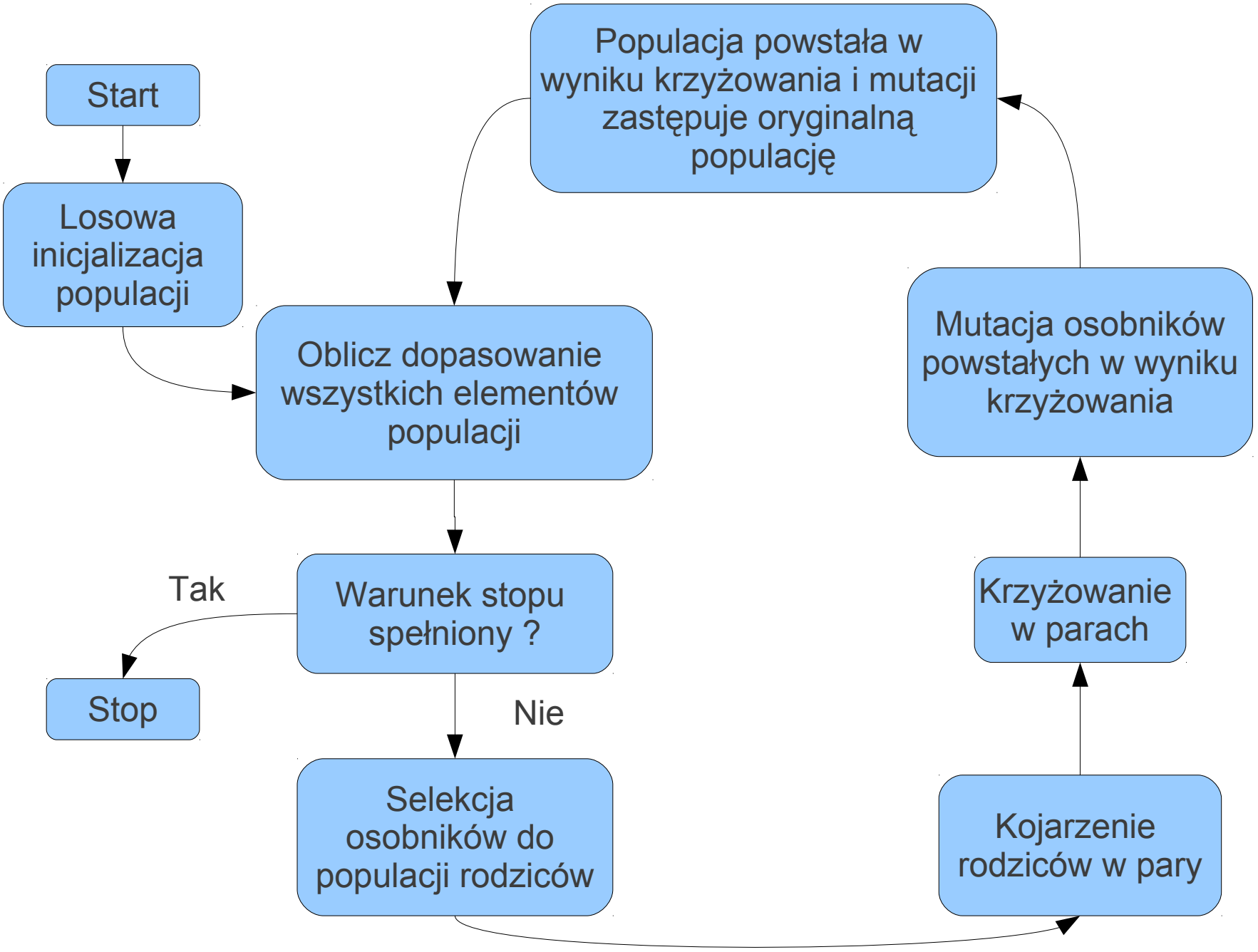
$$f(y) = g(y) + C_1$$

- Problem 2: chcemy znaleźć minimum pewnej funkcji celu g(y). Przekształcenie:

$$f(y) = C_2 - g(y)$$

C_1 i C_2, to odpowiednio duże stałe dodatnie.

Schemat blokowy algorytmu



Standardowy algorytm genetyczny - pseudokod

```
t=0;
initialize(P^0);

while( !termination_condition(P^t)) {
    evaluate(P^t);
    T^t=selection(P^t);
    O^t=crossover(T^t);
    Q^t=mutation(O^t);
    P^t+1=Q^t;
    t=t+1;
}
```

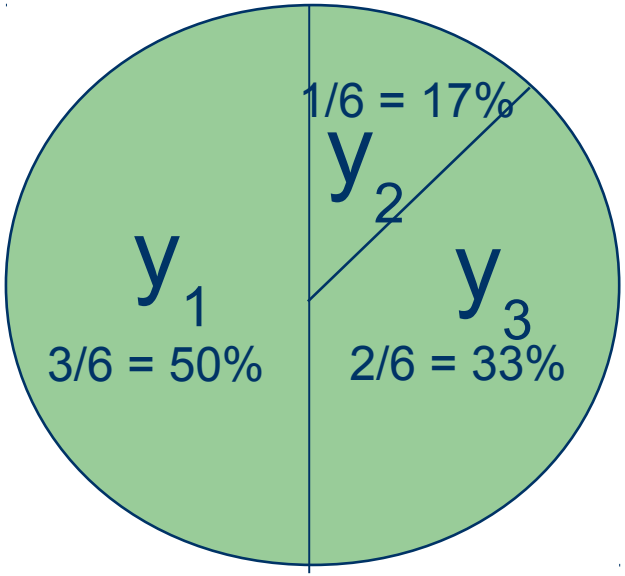
- Warunek stopu: w najprostszej postaci zatrzymanie algorytmu po z góry zadanej liczbie iteracji.
- Mamy do czynienie z sukcesją z całkowitym zastępowaniem: wszyscy rodzice umierają a potomkowie otrzymani w wyniku krzyżowania i mutacji ich zastępują.

Selekcja rodziców

- Idea: lepiej dopasowani osobnicy mają większą szansę bycia wybranym
- Selekcja proporcjonalna: S razy losujemy osobnika (***ze zwracaniem***) z oryginalnej populacji P^t do populacji rodziców T^t, przy czym prawdopodobieństwo wylosowania: $p(y_i)$

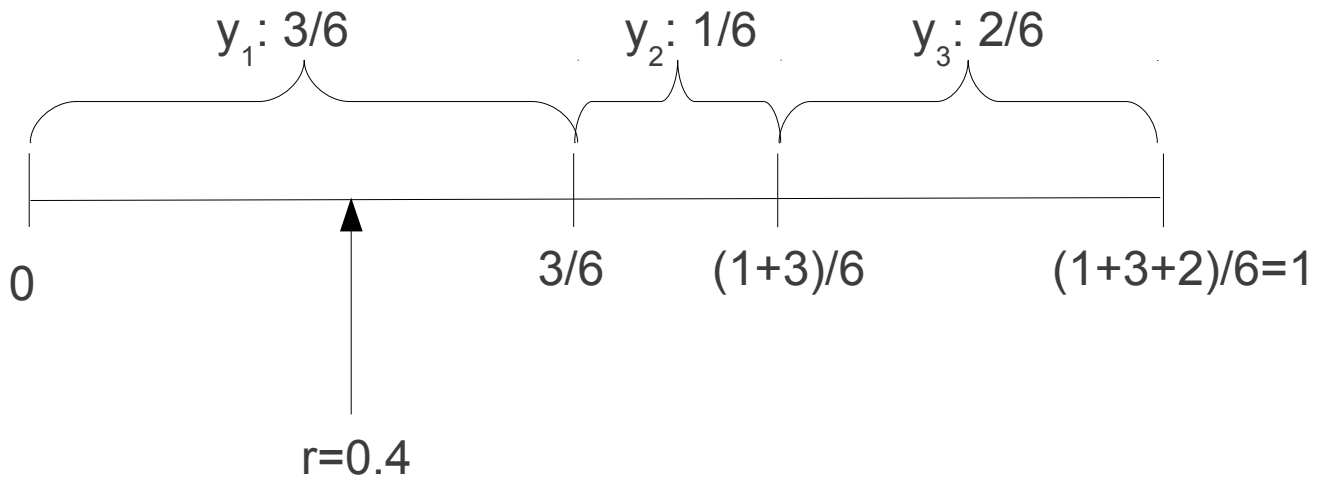
$$p(y_i) = \frac{f(y_i)}{\sum_{i=1}^{S} f(y_j)}$$

- Przykład, S=3, $f(y_1)=3$, $f(y_2)=1$, $f(y_3)=2$



- Oczekiwana liczba kopii każdego z trzech osobników w populacji rodziców:
  - y_1 - 3*3/6=1.5
  - y_2 - 3*1/6=0.5
  - y_3 - 3*2/6=1
- Osobnik o dobrym dopasowaniu ma duże szanse wylosowania kilka razy

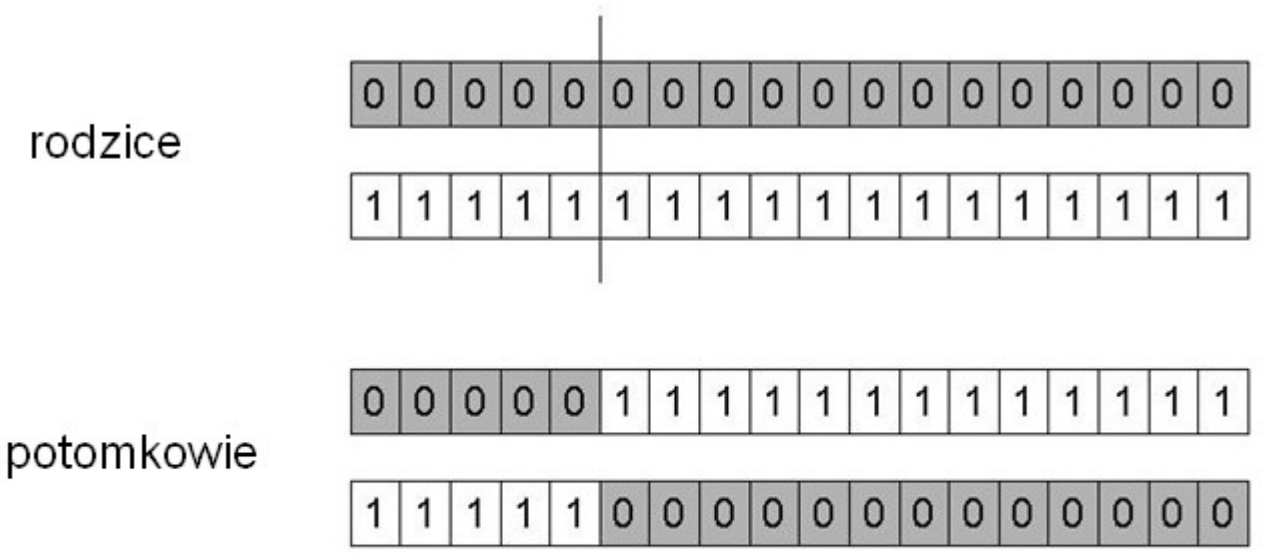
Selekcja: implementacja metodą koła ruletki



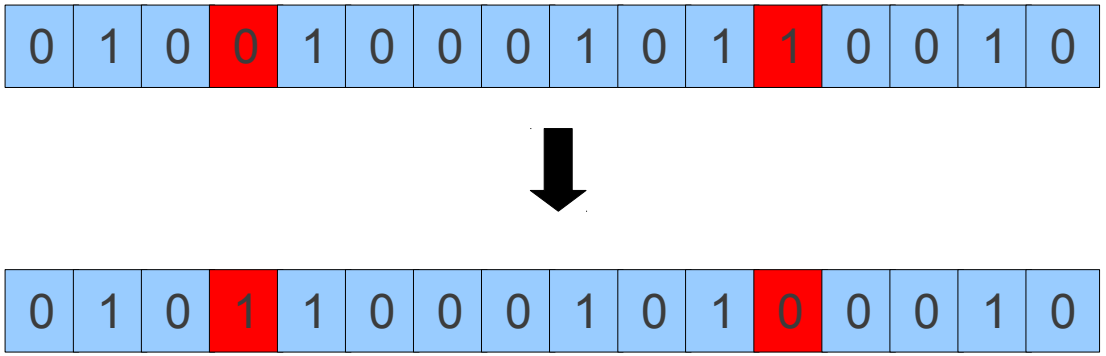
- Dla każdego y_i na przedziale [0,1] odkładamy odcinek o długości $p(y_i)$.
- S razy losujemy liczbę losową z rozkładu jednostajnego na przedziale [0,1].
- Do populacji rodziców wybieramy ten element, na którego odcinek „natrafi” wylosowana liczba losowa r.

Krzyżowanie jednopunktowe

- Operator krzyżowania jest operatorem dwuargumentowym. Przed jego zastosowaniem osobniki w populacji rodziców T^t utworzonej przez selekcję łączone są losowo w pary.
- Dla każdej pary z prawdopodobieństwem p_c (p_c jest parametrem algorytmu, typowo pomiędzy 0.6 a 1) wykonywane jest krzyżowanie:
  - Wybierz losowo punkt krzyżowania w zakresie 1,2,...,L-1.
  - Rozłącz rodziców. Potomkowie otrzymani są w wyniku wymiany fragmentów pomiędzy rodzicami.



Mutacja



- Dla każdego elementu populacji O^t powstałej w wyniku krzyżowania skanuj wszystkie bity.
  - Dla każdego bitu z prawdopodobieństwem p_m (p_m jest parametrem algorytmu) zmień wartość na przeciwną.
  - p_m jest niewielką liczbą (n.p. 0.001 lub wartość pomiędzy 1/S a 1/L). Niewielką, ponieważ mutacja powinna prowadzić do niewielkich zmian w chromosomach.
- Implementacja: dla każdego bitu w każdym chromosomie losuj liczbę r z przedziału [0,1); Zaneguj bit, gdy $r<p_m$

Prosty przykład (D. E. Goldberg)

- Prymitywny problem: $f(x)=x^2$, gdzie $x \in \{0,1,\ldots,31\}$

- Algorytm genetyczny:
  - chromosomy 5 bitowe, kodowanie naturalne np. 01101 ↔ 13
  - rozmiar populacji równy 4.
  - krzyżowanie jednopunktowe + mutacja.
  - selekcja proporcjonalna.
  - Inicjalizacja losowa.

- Zostanie pokazana jedna generacja algorytmu.

Przykład z x^2: selekcja

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

- Osobnik nr 2 o ponadprzeciętnym dopasowaniu został wybrany dwa razy.
- Osobnik nr 3 o kiepskim dopasowaniu nie został wybrany .

Przykład z x^2: krzyżowanie

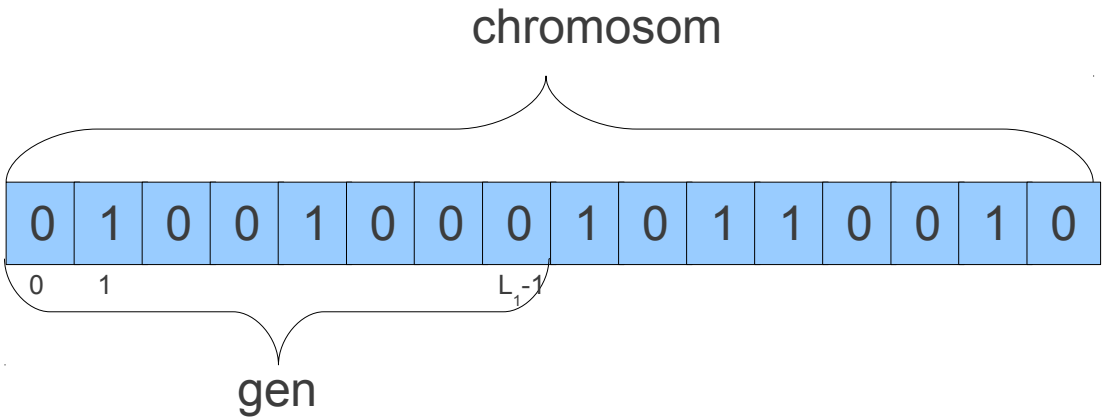
String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

Przykład z x^2: mutacja

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

- Średnie dopasowanie w populacji większe niż na początku !
- Najlepsze dopasowanie w populacji większe niż na początku !
- Czy tak musi być zawsze ?

Kodowanie dla problemu optymalizacji funkcji zmiennych o wartościach rzeczywistych



- L_1 bitów poświęconych na zakodowanie zmiennej rzeczywistej $x \in [a, b]$
- Niech c_i oznacza wartość i-tego bitu (i=0,1,...,L_1-1).
- Wtedy

$$x = a + \frac{b-a}{2^{L_1}-1} \sum_{i=1}^{L_1} c_{L_1-i} * 2^i$$

- Lub prościej, niech U będzie liczbą kodowaną przez L_1 bitów w naturalnym kodzie binarnym, wtedy:

$$x = a + \frac{b-a}{2^{L_1}-1} U$$

- Tylko 2^{L_1} wartości z nieskończonego przedziału [a,b] jest reprezentowanych.
- Większa liczba bitów: większa precyzja (i wolniejsze obliczenia).

Naturalny kod binarny, a kod Gray'a

U	kod NKB	kod Gray'a
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

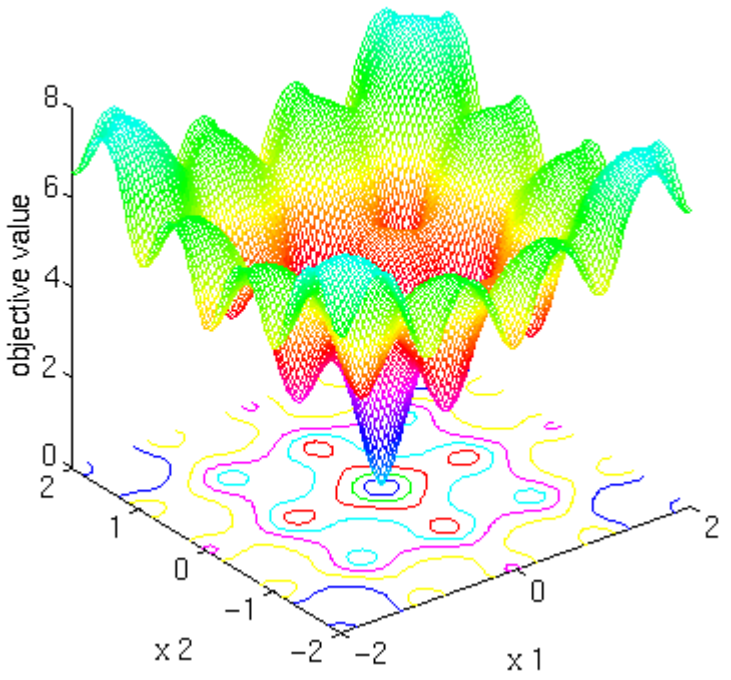
- Kod Gray'a ma ciekawą własność: przejście pomiędzy dwoma sąsiednimi liczbami wymaga zmiany tylko jednego bitu (W przypadku kodu NKB przejście 3↔4 wymaga zmian wszystkich 3 bitów).
- Dzięki temu niewielkie zmiany w fenotypie wymagają niewielkich zmian w genotypie.

Optymalizacja funkcji wielu zmiennych

$$f(\overline{x}) = -c_1 \cdot exp\left(-c_2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$$

$$-exp\left(\frac{1}{n} \cdot \sum_{i=1}^{n} cos(c_3 \cdot x_i)\right) + c_1 + 1$$

$$c_1 = 20,\ c_2 = 0.2,\ c_3 = 2\pi$$



- Posiadając wiedzę, którą dotychczas przekazałem powinniście Państwo być w stanie zastosować algorytm SGA do optymalizacji funkcji wielu zmiennych.
- Na wykresie dwuwymiarowa funkcja Ackley'a mająca minimum f(0,0)=0

Algorytm standardowy nie jest wolny od problemów

- Algorytm był przedmiotem wielu (wczesnych) studiów zarówno eksperymentalnych jak i teoretycznych (teoria schematów, analiza przy pomocy łańcuchów Markowa)
- Krzyżowanie i mutacja działają tylko dla ciągów binarnych (ewentualnie ciągów symboli ze skończonego alfabetu)
- Reprezentacja binarna jest zbyt restrykcyjna. Dla optymalizacji funkcji zmiennych rzeczywistych lepsza by była reprezentacja zmiennopozycyjna.
    - oraz operatory krzyżowania i mutacji działające na tej reprezentacji.
- Model generacyjny sukcesji (wszyscy rodzice wymierają, potomkowie zastępuję rodziców) sprawia, że najlepsze dopasowanie w populacji nie musi monotoniczne rosnąć w kolejnych generacjach.
- Selekcja proporcjonalna nie jest pozbawiona wad (problem ze zbyt słabym albo zbyt silnym naporem selekcyjnym).
    - Superosobnicy we wczesnych fazach algorytmu
    - Prawie identyczne dopasowanie wszystkich osobników w populacji w późnych fazach algorytmu.

Problem z superosobnikami

- We wczesnych fazach ewolucji w populacji może pojawić się jeden (lub kilka) osobnik o dopasowaniu drastycznie lepszym od pozostałych osobników w populacji.

- W takiej sytuacji selekcja proporcjonalna sprawi, że superosobnik otrzyma bardzo dużą liczbę kopii (liczba kopii jest równa dopasowaniu osobnika podzielonemu przez średnie dopasowanie w populacji).

- W krótkim czasie populacja zostanie zdominowana przez (niemal identycznych potomków) superosobnika i algorytm straci możliwość eksploracji przestrzeni rozwiązań.
  - Zjawisko przedwczesnej zbieżności algorytmu (ang. premature convergence)

Słaby napór selekcyjny w późnych fazach algorytmu

- W późnych fazach algorytmu przystosowanie wszystkich osobników w populacji jest w przybliżeniu takie same.
- W takim wypadku każdy osobnik ma podobne szanse na bycie wybranym w procesie selekcji.
- Algorytm ma trudności z preferowaniem bardziej dopasowanych osobników i z dalszym ulepszeniem dopasowania.
- „Błądzenie przypadkowe wśród przeciętniaków” (D.E. Goldberg).

Liniowe skalowanie dopasowania (ang. fitness scaling)

- Prawdopodobieństwo selekcji: $$p(y_i)=\frac{f(y_i)}{\sum_{i=1}^{S} f(y_j)}$$

- Oczekiwana liczba kopii osobnika w populacji T^t (S jest licznością populacji):

$$S*p(y_i)=\frac{S*f(y_i)}{\sum_{i=1}^{S} f(y_j)}=\frac{f(y_i)}{f_{AVG}},$$

gdzie f_{AVG} jest średnim przystosowaniem w populacji.

- Problem z selekcją proporcjonalną pojawia się wtedy gdy f_{MAX} (dopasowanie najlepszego osobnika w populacji jest zbyt duże lub zbyt małe).
- Idea skalowania liniowego: Przekształcić wartości funkcji dopasowania tak aby najlepszy osobnik otrzymał zawsze stałą liczbę kopii równą k.
  - A dopasowanie przeciętnego osobnika w populacji się nie zmieniło.

Liniowe skalowanie dopasowania (2)

- Funkcja dopasowania po przeskalowaniu: $f'(y_i) = a * f(y_i) + b$

przy czym współczynniki a oraz b obliczamy w taki sposób że:

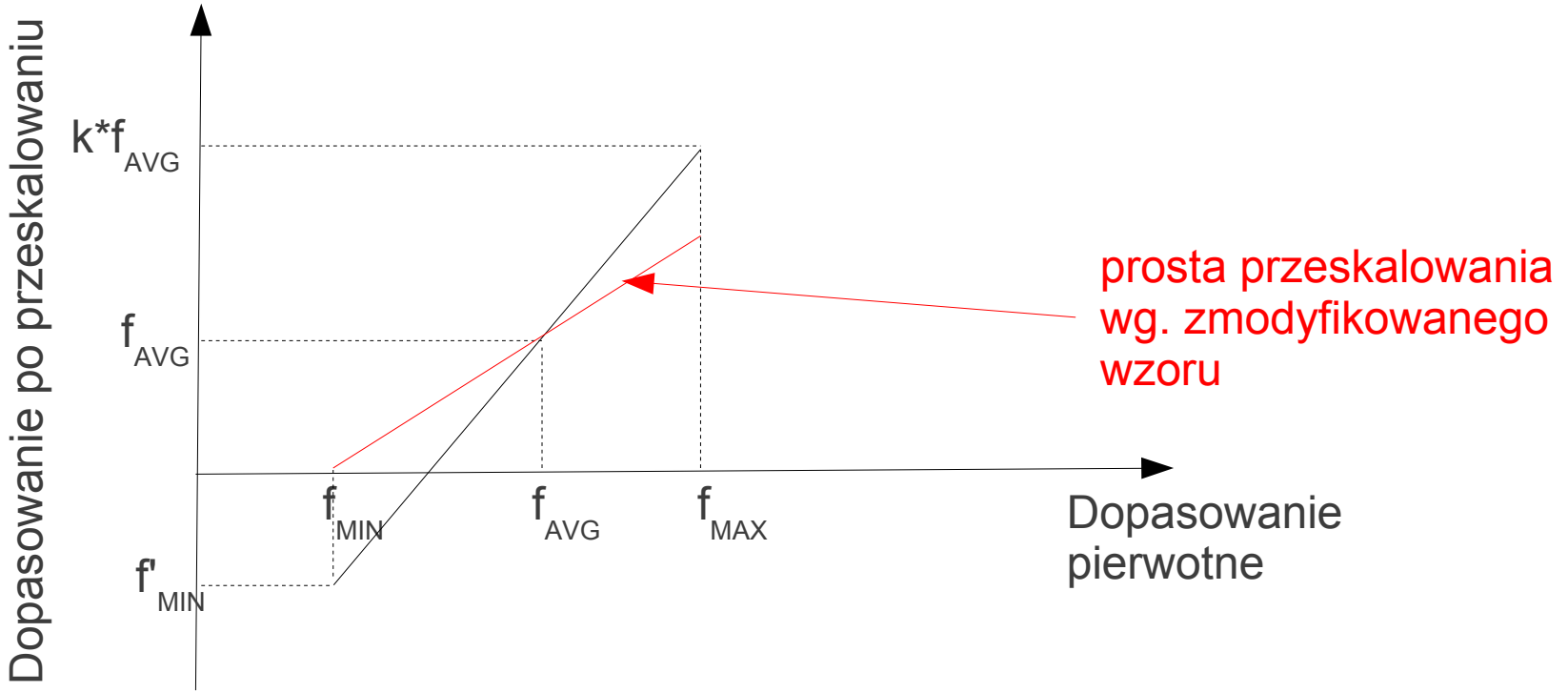
- dopasowanie osobnika przeciętnego się nie zmienia $f_{AVG} = a * f_{AVG} + b$
- najlepszy osobnik otrzyma k kopii $k * f_{AVG} = a * f_{MAX} + b$

przy czym dla typowych populacji (od 50 do 100 osobników) D.E. Goldberg zaleca

$k \in [1.2, 2]$ (k staje się kolejnym parametrem algorytmu)

- Rozwiązując powyższy układ równań liniowych (ze względu na niewiadome a oraz b) znajdujemy współczynniki skalowania.
  - A jak nie potrafimy rozwiązać układu równań, sięgamy do książki Goldberga i przepisujemy kod.
- Wartości dopasowania uzyskane w wyniku skalowania wykorzystane są w procesie selekcji proporcjonalnej, ale ....

Liniowe skalowanie dopasowania - problem



- W dojrzałej populacji (dopasowanie zbliżone, z wyjątkiem kilku „degeneratów”) wyniku skalowania możemy otrzymać ujemne wartości funkcji dopasowania – nie da się zastosować selekcji proporcjonalnej.
- Wyjście (D.E. Goldberg) zastąp warunek: $k * f_{AVG} = a * f_{MAX} + b$

przez warunek:

$$0 = a * f_{MIN} + b$$

(drugi warunek nie ulega zmianie)

Wielokrotne losowanie przy pomocy koła ruletki - problem

- Problemem jest duża wariancja liczby kopii osobnika w populacji rodziców T^t. W rezultacie liczba kopii może dość znacznie się różnić od wartości oczekiwanej.
- Rozważmy populację o rozmiarze S=2 i następujących wartościach funkcji dopasowania:

i	$f(y_i)$	$p(y_i)$	$S*p(y_i)$
1	9	0.9	1.8
2	1	0.1	0.2

- W powyższej populacji dopasowanie osobnika y1 jest znacznie większe. Jednak możliwa jest sytuacja (jej prawdopodobieństwo wynosi 0.1*0.1=0.001), że ani razu nie zostanie on wybrany do populacji rodziców T^t.
- Wyjście: lepsze algorytmy próbkowania populacji, redukujące wariancję liczby kopii osobnika np. stochastic remainder sampling (SRS - Brindle, 1981), stochastic universal sampling (SUS - Baker 1987).

Stochastic Remainder Sampling

i	$f(y_i)$	$p(y_i)$	$S*p(y_i)$	$\lfloor S*p(y_i) \rfloor$	$S*p(y_i)-\lfloor S*p(y_i) \rfloor$
1	4	0.4	2	2	0
2	1	0.1	0.5	0	0.5
3	1	0.1	0.5	0	0.5
4	3	0.3	1.5	1	0.5
5	1	1	0.5	0	0.5

- Najpierw deterministycznie wybierz do populacji O^t $\lfloor S*p(y_i) \rfloor$ kopii każdego z osobników.
- Następnie użyj $S*p(y_i)-\lfloor S*p(y_i) \rfloor$ jako nowej wartości dopasowania przy selekcji pozostałych kopii (tak aby łączny rozmiar populacji był równy S) metodą koła ruletki.
  - W przykładzie w tabelce osobniki $y_2,...,y_5$ rywalizują o dwa pozostałe miejsca.
- Gwarancja że liczba kopii osobnika ≥ $\lfloor S*p(y_i) \rfloor$
- Jeszcze lepszy algorytm SUS pozostawiam jako pracę domową.

Zakończenie

- Dla celów Państwa projektu za algorytm standardowy będziemy uważali algorytm z:
  - próbkowaniem przy pomocy metody SRS albo SUS.
  - liniowym skalowaniem dopasowania.
  - kodowaniem Gray'a w przypadku optymalizacji funkcji zmiennych rzeczywistych.
- Następny wykład: Ulepszenia algorytmu standardowego.
  - Nowe operatory krzyżowania.
  - Nowe algorytmy selekcji.
  - Reprezentacja zmiennopozycyjna i specjalnie zaprojektowane dla niej operatory (Michalewicz, rozdz. 5).