Modyfikacje i ulepszenia standardowego algorytmu genetycznego

Przypomnienie: pseudokod SGA

```
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;
}
```

Krzyżowanie czy mutacja

- Który operator jest ważniejszy ? Długa (i nierozstrzygnięta - zależy od problemu) debata w społeczności naukowców. Poniżej przedstawiam mój własny pogląd:
- Oba operatory pełnią ważną funkcję.
- Krzyżowanie pozwala na ***eksplorację*** - odkrywanie nowych obiecujących obszarów w przestrzeni rozwiązań.
  - Osobnik po krzyżowaniu jest diametralnie różny od swoich rodziców - „duże skoki” w przestrzeni rozwiązań
  - Tylko krzyżowanie jest w stanie na połączyć informacje pochodzące z dwóch osobników.
- Mutacja, poprzez niewielkie zmiany (osobnik po mutacji bardzo przypomina rodzica) wykonuje ***eksploatację*** - optymalizację lokalną w obiecującym regionie przestrzeni cech odkrytym przez krzyżowanie.
- Możliwe są algorytmy ewolucyjne działające wyłącznie w oparciu o mutację. Algorytm bez mutacji oparty wyłącznie na krzyżowaniu nie może działać (dlaczego ?).

Kryteria zatrzymania algorytmu

- Do tej pory wspomniałem o zakończeniu algorytmu po zadanej z góry liczbie iteracji. Lepszym kryterium, niewrażliwym na rozmiar populacji S, jest zatrzymanie algorytmu po zadanej z góry liczbie obliczeń funkcji dopasowania.
- Zatrzymanie po osiągnięciu zadanej wartości funkcji dopasowania (ale dla wielu problemów nie znamy optymalnej wartości tej funkcji).
- Zatrzymanie gdy najlepsza wartość dopasowania w populacji nie ulega ulepszeniu przez pewną, zadaną liczbę generacji.
- Zatrzymanie po przekroczeniu maksymalnego wykorzystanego czasu pracy procesora.
- Zatrzymanie, gdy różnorodność osobników w populacji, mierzona np. przez entropię, spadnie poniżej pewnego progu.

Selekcja turniejowa (ang. tournament selection)

- Turniej
  - Wybierz losowo q osobników (***bez zwracania***) z populacji P^t. Rozmiar turnieju q jest parametrem algorytmu.
  - Skopiuj najlepszego z wybranych osobników do populacji rodziców T^t.
  - Zwróć q osobników z powrotem do P^t.
- Organizuj turniej S razy otrzymując kompletną populację rodziców T^t o liczności S.
  - Im większa wartość parametru q, tym większy napór selekcyjny.
- Zalety
  - brak ograniczeń odnośnie optymalizowanej funkcji znanych z selekcji proporcjonalnej.
  - nie wymaga znajomości optymalizowanej funkcji (musimy tylko wiedzieć, kiedy jeden osobnik jest lepszy od drugiego - przydaje się to w zastosowaniach np. w teorii gier).
  - nie wymaga globalnych statystyk populacji, a przez to jest łatwiejsza w implementacji na maszynach równoległych.

Selekcja oparta na rangach (rangowa)

- Próba ominięcia wad selekcji proporcjonalnej poprzez oparcie prawdopodobieństwa selekcji nie na bezwzględnej wartości funkcji dopasowania, ale na jej porównaniu z dopasowaniem innych osobników.
- Nadanie rang: posortuj osobników w populacji w oparciu o wartość funkcji dopasowania. Osobnik o największym dopasowaniu otrzymuje rangę S, osobnik o dopasowaniu drugim co do wielkości rangę S-1,..., osobnik najgorszy rangę 1.
  - Łatwo poradzimy sobie z problemami typu funkcja celu osiąga wartości mniejsze od 0 bądź wymaga minimalizacji.
  - Algorytm wymaga sortowania, o koszcie obliczeniowym O(S*logS), ale jest on z reguły niewielki w porównaniu z kosztem obliczenia funkcji dopasowania.
- Niech $r(y_i)$ jest rangą osobnika y_i.
- Prawdopodobieństwo selekcji $p(y_i)$ jest wyliczone wyłącznie na podstawie $r(y_i)$, a nie na postawie $f(y_i)$.
- Jeżeli znamy już $p(y_i)$, to dalej wykorzystujemy metodę koła ruletki, lub lepiej algorytm SRS albo SUS.

Selekcja rangowa z liniową funkcją prawdopodobieństwa

$$p(y_i) = \frac{2-s}{S} + \frac{2*r(y_i)(s-1)}{S(S-1)}$$

- $p(y_i)$ jest liniową funkcją $r(y_i)$.
- S: rozmiar populacji, 1<s≤2 parametr algorytmu określający liczbę kopii najlepszego osobnika (im większy s, tym większy napór selekcyjny).
- Przykład dla S=3 (Eiben i Smith):

i	$f(y_i)$	$r(y_i)$	$p_{prop}(y_i)$	$p_{rank}(y_i)$ [s=2]	$p_{rank}(y_i)$ [s=1.5]
1	1	1	0.1	0.0	0.0167
2	5	3	0.5	0.67	0.5
3	4	2	0.4	0.33	0.33
Suma	10		1.0	1.0	1.0

widoczny jest większy napór selekcyjny dla s=2 niż dla s=1.5

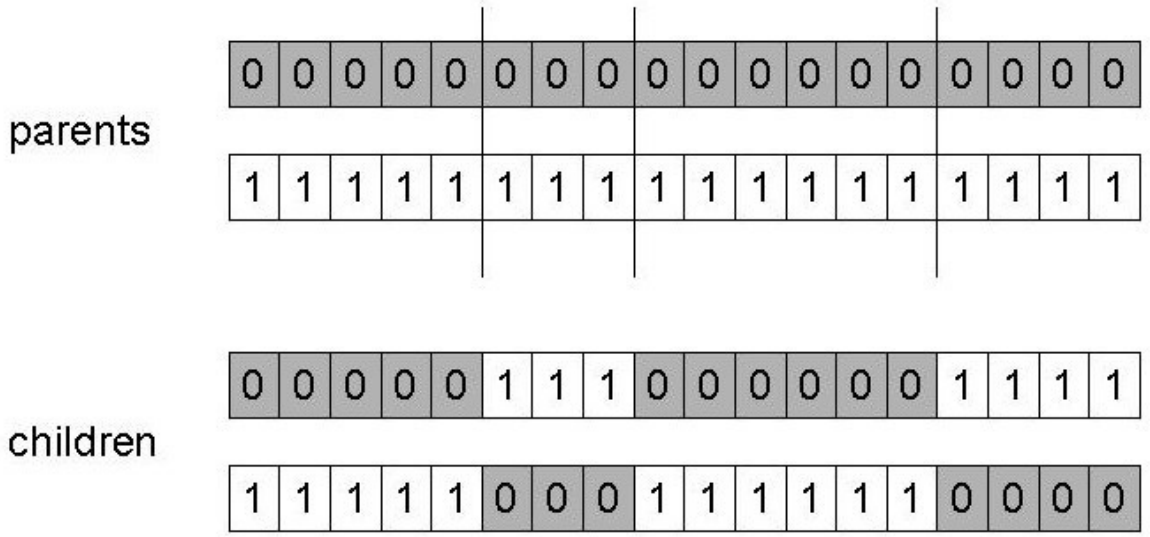
Elitarny model sukcesji (K. A. de Jong, 1975)

- Model sukcesji z całkowitym zastępowaniem rodziców przez potomków, może prowadzić do zjawiska w którym wartość najlepszego dopasowania w populacji w generacji t+1 jest mniejsza (gorsza) niż wartość najlepszego dopasowania w populacji w generacji t.
- Elitaryzm:
  - Zapamiętaj najlepszego y_{Best} osobnika w populacji P^t.
  - Jeżeli dopasowanie najlepszego osobnika w populacji P^{t+1} jest mniejsze od dopasowania y_{Best}, wstaw y_{Best} do populacji P^{t+1} (na przykład na miejsce osobnika o najmniejszym dopasowaniu).
- Algorytm można rozszerzyć na zapamiętywanie η najlepszych osobników (J. Arabas, s. 130).
- Model elitarny przyspiesza zbieżność, ale może również prowadzić do przedwczesnej zbieżności do optimum lokalnego, zwłaszcza przy dużym η.

Algorytm genetyczny stanu stałego (ang. steady state)

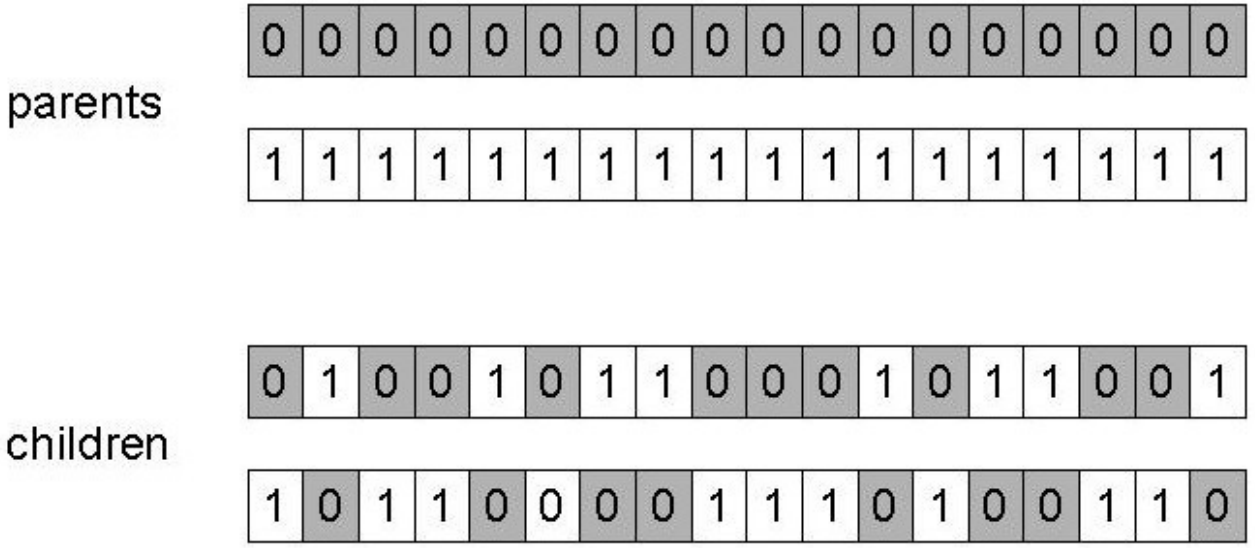
- Algorytm SGA jest algorytmem z generacyjnym modelem sukcesji.
  - każdy osobnik przeżywa dokładnie jedną generację.
  - populacja potomków w pełni zastępuje populację rodziców.
- Na przeciwnym biegunie jest algorytm stanu stałego, w którym w jednej generacji zastępowany jest jeden osobnik w populacji.
  - wybierz (np. przy pomocy selekcji turniejowej) dwóch rodziców.
  - poddaj ich krzyżowaniu i mutacji.
  - wybierz lepszego z potomków.
  - zastąp jednego z osobników populacji.
- Zastępowany osobnik może być (sukcesja):
  - najstarszym osobnikiem w populacji.
  - osobnikiem w populacji o najgorszym dopasowaniu.
  - jednym z rodziców.
  - losowym osobnikiem.
  - osobnikiem najbardziej podobnym do potomka.

Krzyżowanie wielopunktowe (rys. - Eiben & Smith)



- Wybierz losowo n punktów przecięcia.
- Podziel chromosomy według n-punktów.
- Sklej odcinki wymieniając fragmenty chromosomów pomiędzy rodzicami.
- Z krzyżowaniem wielopunktowy eksperymentował K. A. de Jong (1975)

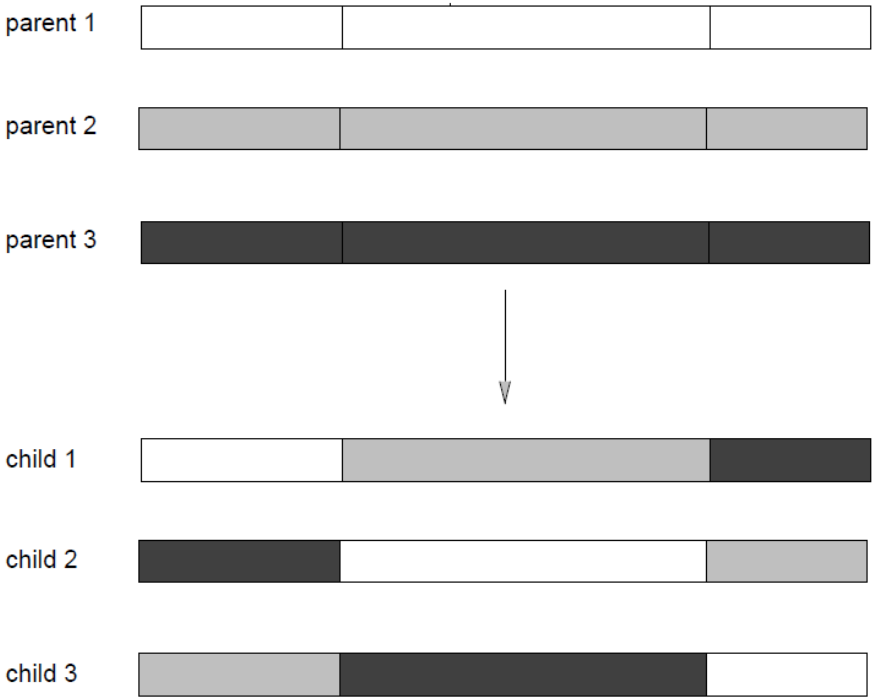
Krzyżowanie jednostajne (ang. uniform, Syswerda 1989)



- Wymiana każdej pary genów pomiędzy chromosomami jest przeprowadzana z prawdopodobieństwem 0.5
- Który operator krzyżowania wybrać ? W eksperymentach najgorzej wypada krzyżowanie jednopunktowe (Eshelman i wsp., 1989), ale brak zdecydowanego zwycięzcy.

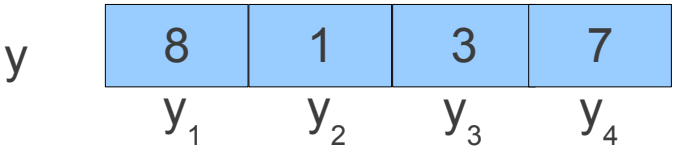
Krzyżowanie wielorodzicielskie (ang. multi-parent)

- Mutacja jest operatorem operującym na jednym rodzicu. Krzyżowanie operuje na dwóch. Projektując operatory nie jesteśmy ograniczeni przez mechanizmy znane w przyrodzie.
- Stąd idea krzyżowania z liczbą rodziców p>2.
- Przykład: krzyżowanie diagonalne (będące uogólnieniem krzyżowania wielopunktowego)



Liczba punktów cięcia
równa liczbie rodziców

Reprezentacja całkowitoliczbowa (ang. integer)



- Chromosom ma postać $y=[y_1, y_2, \ldots, y_L]$

gdzie najczęściej $y_i \in \{a_i, a_i+1, \ldots, b_i-1, b_i\} \subset \mathbb{C}$

- Niektóre problemy optymalizacyjne charakteryzują się naturalnie zmiennymi całkowitymi (np. przetwarzanie obrazów).
- Inne dają się zakodować łatwo przy pomocy liczb całkowitych np. zmienne kategoryczne w eksploracji danych.
- Operatory krzyżowania n-punktowego i jednostajnego działają bez zmian.
- Mutacja: wylosuj losowo liczbę ze zbioru wartości zmiennej całkowitej y_i.
  - Ale rozważmy sytuację w której wartości całkowite kodują zakres wartości zmiennej wyrażającej stopień bólu: {0-Brak, 1-Słaby, 2-Średni, 3-Silny,4-Bardzo Silny, 5 - Nie do zniesienia}
  - Być może operator mutacji powinien być tak skonstruowany aby zmiana 5→4 była o wiele bardziej prawdopodobna niż 5→0.

Reprezentacja zmiennopozycyjna

- Generalnie obecnie uważa się, że dla problemów optymalizacji numerycznej najlepsza jest reprezentacja zmiennopozycyjna, jako „pojęciowo bliższa przestrzeni zadania” (Michalewicz, 1996, s. 129).
- W tej reprezentacji chromosom x ma postać ciągu (wektora) liczb rzeczywistych:

$$x = [x_1, x_2, \ldots, x_L] \qquad \text{gdzie} \qquad x_i \in (a_i, b_i) \subset \mathbb{R}$$

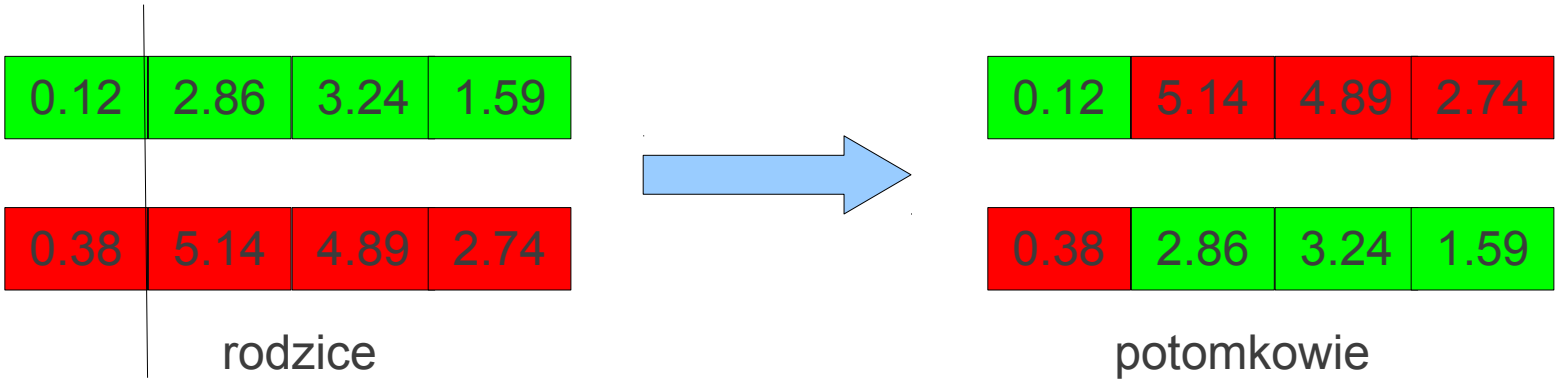
x

0.12	2.86	3.24	1.59
x_1	x_2	x_3	x_4

- Pozostaje zaprojektowanie operatorów krzyżowania i mutacji działających dla tej reprezentacji. Nie jest to trywialne.
- W niektórych złożone problemy optymalizacyjne, np. w mechanice, część zmiennych jest całkowita a cześć rzeczywista.

Pierwsza próba (Michalewicz, 1996, s. 132)

- Operator krzyżowania: standardowe krzyżowanie jednopunktowe, w którym wymieniane są kompletne liczby zmiennoprzecinkowe.



- Operator mutacji: mutując gen x_i wylosuj liczbę z przedziału (a_i,b_i).
  - Implementacja: jeżeli r jest liczbą losową z przedziału (0,1), to

$$x_i' = a_i + r * (b_i - a_i)$$

- Wyniki nie najlepsze (gorsze, niż w przypadku chromosomów binarnych), ponieważ operator mutacji prowadzi do dużych zmian w chromosomie.

Operator mutacji nierównomiernej

- t jest numerem iteracji, T całkowitą liczbą iteracji, po której zatrzymujemy algorytm.

$$x_i' = \begin{cases} x_i + \Delta(t, b_i - x_i) \text{ z prawd. } 0.5 \\ x_i - \Delta(t, x_i - a_i) \text{ z prawd. } 0.5 \end{cases}$$

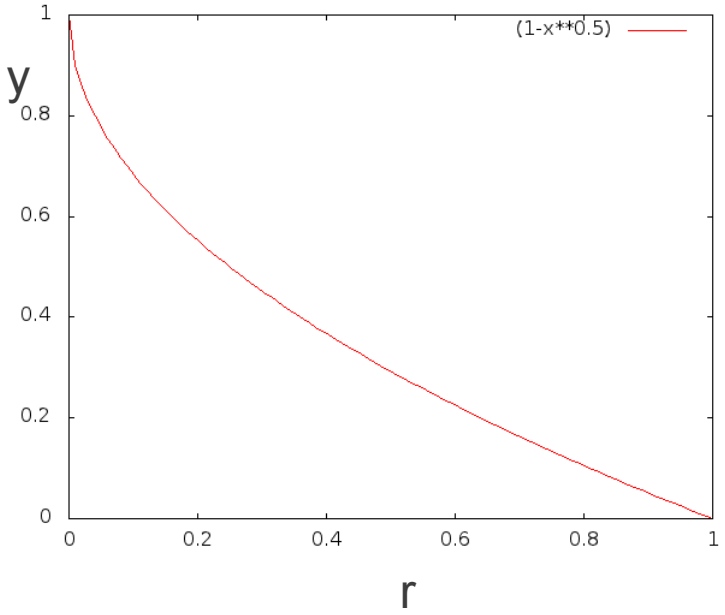
- Funkcja Δ(t,y) zwraca liczbę losową z przedziału [0,y], przy czym wraz ze wzrostem numeru iteracji t rośnie prawdopodobieństwo, że Δ(t,y) jest bliskie zeru. Jest ona dana wzorem (Michalewicz przyjął c=2):

$$\Delta(t, y) = y * (1 - r^{(1 - t/T) * c})$$

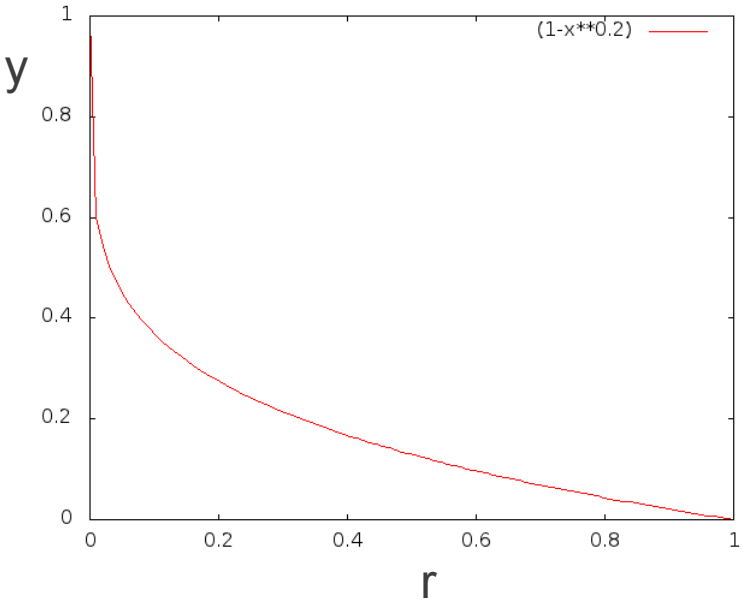
- Gdzie r jest liczbą losową z przedziału [0,1].

Mutacja nierównomierna - funkcja Δ

$$\Delta(t, y) = y * \left(1 - r^{(1 - t/T) * c}\right)$$



t/T=0.5 (algorytm zaawansowany w 50%)



t/T=0.9 (algorytm zaawansowany w 90%)

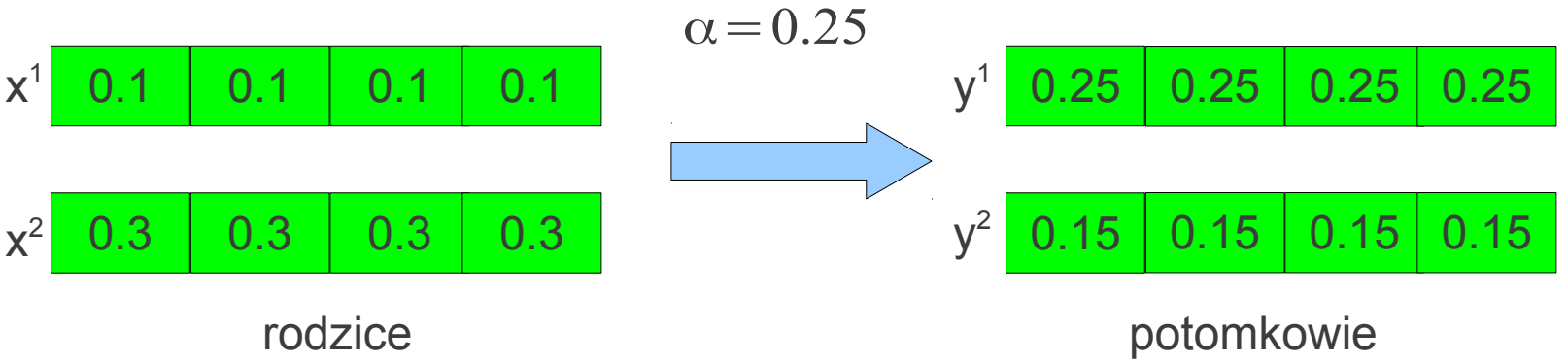
- Im większe zaawansowanie algorytmu, tym mniejsza szansa wylosowania dużych wartości y.

Operatory krzyżowania arytmetycznego (1)

- Operator whole arithmetic crossover z dwóch rodziców x^1 i x^2 tworzy dwóch potomków y^1 oraz y^2.

$$y^1 = \alpha * x^1 + (1-\alpha) * x^2$$
$$y^2 = (1-\alpha) * x^1 + \alpha * x^2$$
$$\text{gdzie } \alpha \in \{0,1\}$$

- Idea: Stwórz potomków ulokowanych „pomiędzy” rodzicami. Przykład:



- Parametr α może być:
  - Stały podczas procesu ewolucji.
  - Losowany z każdą aplikacją operatora krzyżowania.
  - Zmienny w zależności od numeru iteracji algorytmu genetycznego.

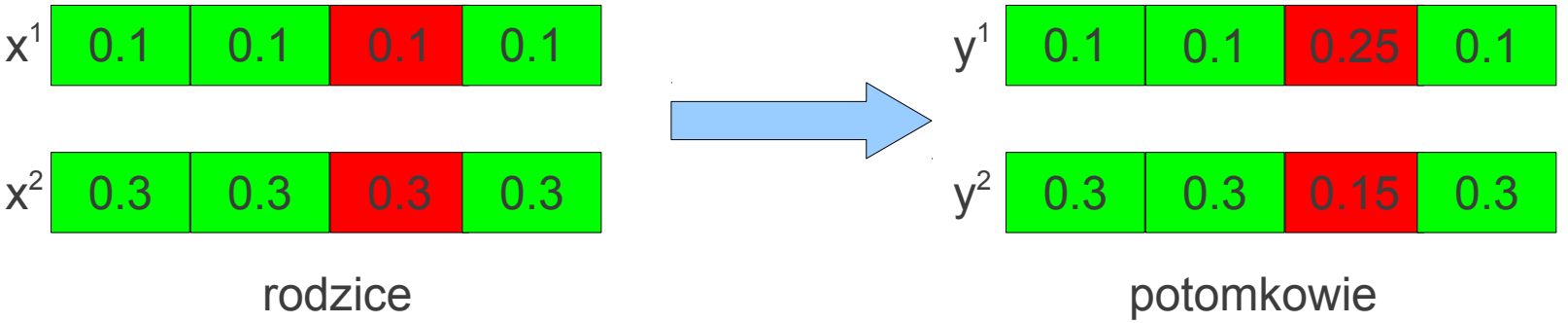
Operatory krzyżowania arytmetycznego (2)

- Operator single arithmetic crossover z dwóch rodziców x^1 i x^2 tworzy dwóch potomków y^1 oraz y^2.

$$y^1 = \{x_{1,}^1 x_{2,}^1 \ldots, x_{k-1}^1, \alpha * x_k^1 + (1-\alpha) x_k^2, x_{k+1}^1, \ldots x_L^1\}$$
$$y^2 = \{x_{1,}^2 x_{2,}^2 \ldots, x_{k-1}^2, \alpha * x_k^2 + (1-\alpha) x_k^1, x_{k+1}^2, \ldots x_L^2\}$$

- Parametr α wybierany podobnie jak poprzednio. k jest losowo wybranym numerem genu. Przykład:

$$\alpha = 0.25,\ k = 3$$



Operatory krzyżowania arytmetycznego (3)

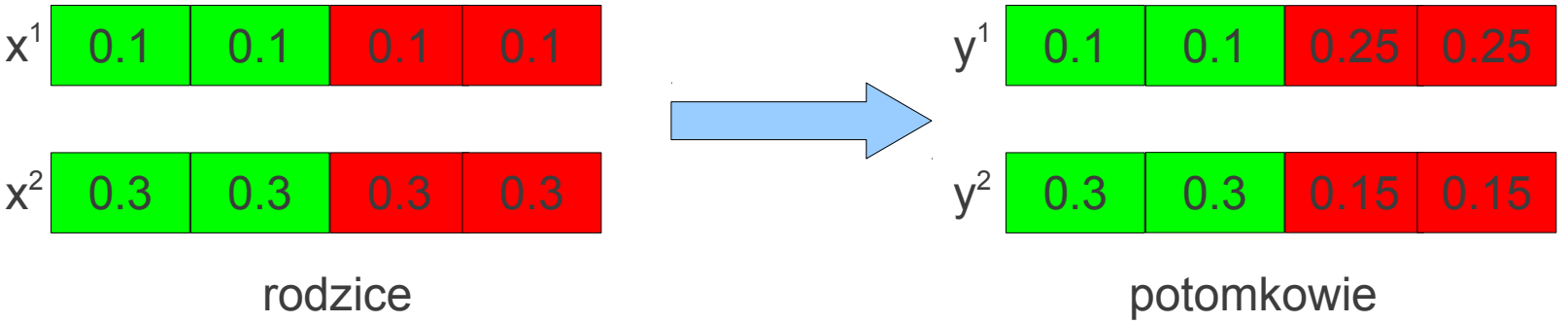
- Operator simple arithmetic crossover z dwóch rodziców x^1 i x^2 tworzy dwóch potomków y^1 oraz y^2., w ten sposób że pierwszych k genów pozostaje bez zmian, a następnych L-k jest obliczanych jak w operatorze whole arithmetic crossover

$$y^1 = \{x^1_1, \ldots, x^1_k, \alpha * x^1_{k+1} + (1-\alpha) x^2_{k+1}, \ldots, \alpha * x^1_L + (1-\alpha) x^2_L\}$$

$$y^2 = \{x^2_1, \ldots, x^2_k, \alpha * x^2_{k+1} + (1-\alpha) x^1_{k+1}, \ldots, \alpha * x^2_L + (1-\alpha) x^1_L\}$$

- Parametr α wybierany podobnie jak poprzednio. k, podobnie jak poprzednio, jest losowo wybranym numerem genu. Przykład:

$$\alpha = 0.25,\ k = 2$$



Podsumowanie

- Algorytm genetyczny z reprezentacją zmiennopozycyjną ***i specjalnymi operatorami*** daje znacznie lepsze wyniki niż algorytm z klasyczną reprezentacją binarną (Michalewicz, 1996, rozdz. 5 oraz 6)
  - reprezentacja i operatory dopasowane do problemu.
- Podobnie jest w przypadku problemów optymalizacji dyskretnej.
  - np. reprezentacja permutacyjna i operatory krzyżowania oraz mutacji dla problemu komiwojażera.
- Za tydzień strategie ewolucyjne (evolution strategies).
  - zaprojektowane w Europie (Niemcy).
  - przeznaczone do optymalizacji numerycznej.
  - temat jednego z projektów.