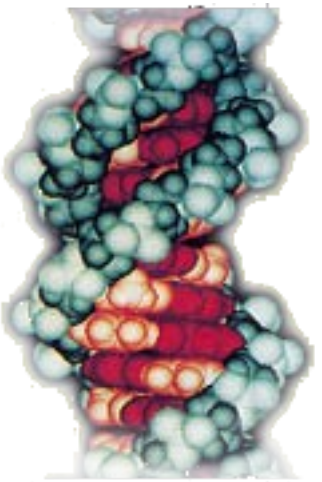
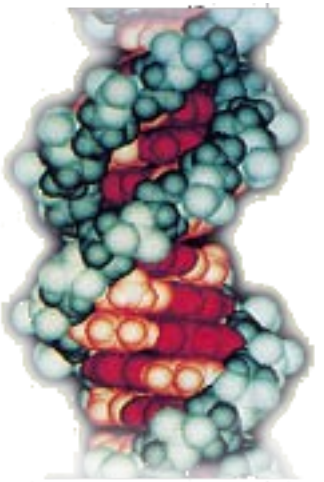
Skuteczność genetycznego systemu wyznaczania reguł w zbiorze danych na przykładzie problemu MONKS



Seminarium Metod Inteligencji Obliczeniowej
Warszawa 21 XI 2007

mgr inż. Marcin Borkowski

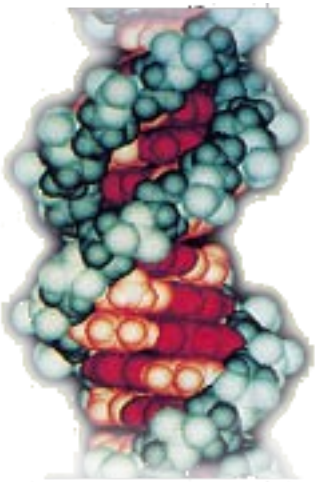
Dziś opowiem:



- Działaniu niszowego AG
- problemie MONKS
- Mojej aplikacji niszowego AG do problemu MONKS -Rule EXtractor (REX)
- Wynikach

Niszowy AG

Nabór



- Stały rozmiar populacji
- Kodowanie binarne
- Nabór:

Osobniki w populacji są sortowane względem ich przystosowania.(za wyjątkiem tych z minimalnym ściskiem, które są zawsze na początku) Osobniki najsłabsze są następnie zastępowane przez wyniki operatorów genetycznych.

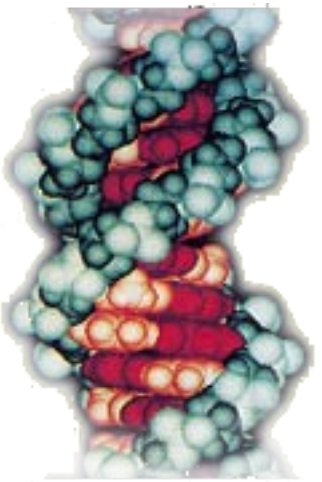
Pozwala to najlepszym osobnikom unikać “śmierci” tak długo aż zostaną wyparte przez osobniki jeszcze lepsze.

Wyniki operatorów są umieszczane w tymczasowej kopii populacji co daje równe szanse także osobnikom spisanym na straty

Nie ma klonowania osobników

Niszowy AG

Nabór przykład



Bieżąca Pop.
0,6441
0,6569
0,6697
0,6825
0,7593
0,7721
0,5673
0,5801
0,5929
0,6057
0,6185
0,6313
0,8489
0,8617
0,8745
0,8873
0,7849
0,7977
0,8105
0,8233
0,8361

Sortowana Pop.
0,5673
0,5801
0,5929
0,6057
0,6185
0,6313
0,6441
0,6569
0,6697
0,6825
0,7593
0,7721
0,7849
0,7977
0,8105
0,8233
0,8361
0,8489
0,8617
0,8745
0,8873

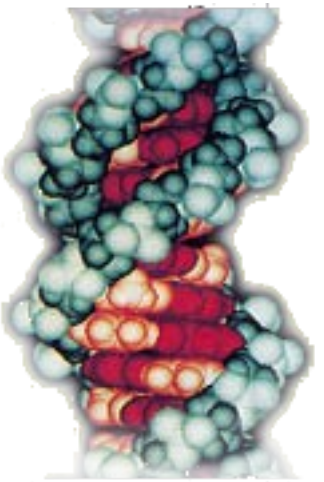
Krzyżowanie

Mutacja

Nowa Populacja
0,9023
0,4582
0,7921
0,6057
0,6185
0,6313
0,6441
0,6569
0,6697
0,6825
0,7593
0,7721
0,7849
0,7977
0,8105
0,8233
0,8361
0,8489
0,8617
0,8745
0,8873

Niszowy AG

Mutacja



- Mutacja sterowana dwoma parametrami:

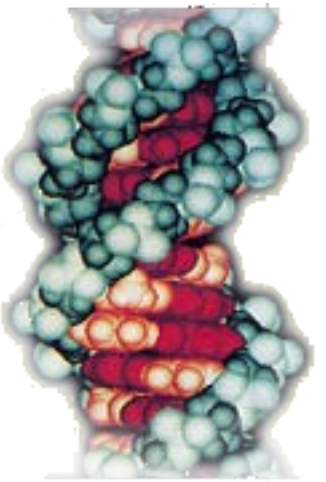
Każdy osobnik podlega mutacji z zadanym prawdopodobieństwem mutacji struktury *pms*.

Każdy bit osobnika wylosowanego do mutacji podlega mutacji (inwersja) z prawdopodobieństwem *pm*.

Wynik mutacji wypiera z populacji osobnika najsłabszego z jeszcze niepodmienionych

Niszowy AG

Krzyżowanie



- Krzyżowanie klasyczne jednopunktowe z prawdopodobieństwem *pc*

  Z sumy wartości oczekiwanych *pms* i *pc* wynika ze można oczekiwać, że najlepsze osobniki nie zostaną zniszczone

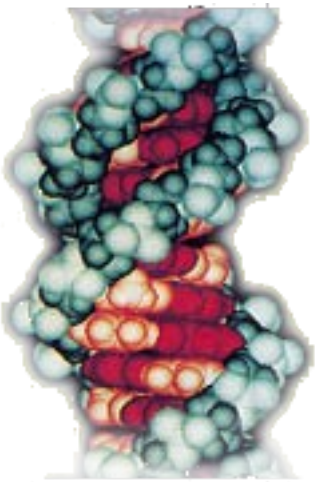
  Nabór nie namnaża super osobników, dłuższe przetrwanie w populacji gwarantuje ich pozycja w rankingu

  Kopie powstają rzadko na skutek znoszących się mutacji lub krzyżowania już istniejących kopii

  Jeśli jednak już powstaną osobniki identyczne to nie będą one jednocześnie zajmować wysokiej pozycji w populacji – patrz ścisk

Niszowy AG

Ocena



- Funkcja oceny:

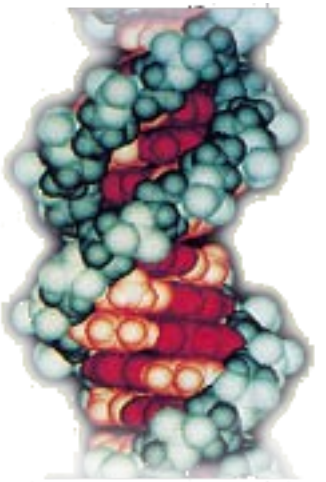
Ponieważ rozważane są głównie problemy dopasowania wzorca do serii danych konieczne jest policzenie wielu ocen – stopnia dopasowania dla wszystkich możliwych pozycji wzorca, do AG brana jest wartość uśredniona

Ocena rozbita jest na dwa etapy, poza zwyczajową ewaluacją osobnika następuje redukcja oceny na skutek działania czynnika ścisku – patrz ścisk

Ścisk wyznaczany jest dopiero gdy znane są wszystkie funkcje oceny z pierwszego etapu

Niszowy AG

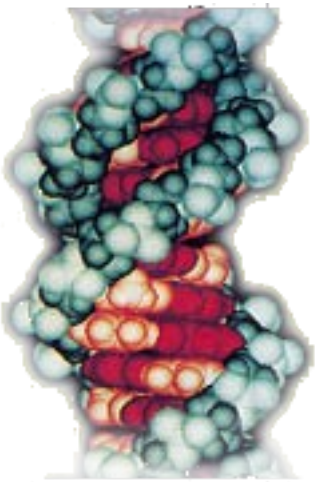
Nisze-ścisk



- Rozwiązań dostarczanych przez AG jest wiele, są to najlepsze optima w przeszukiwanej przestrzeni. Ich ilość zależy od badanej przestrzeni (ilość ekstremów) i rozmiaru populacji
- Przy odpowiednio dobranym parametrze podobieństwa osobników jedno optimum przekłada się na jedną niszę
- Przeszukiwanie wielu optimów naraz jest możliwe dzięki ściskowi

Niszowy AG

Ścisk



W zamyśle ma działać jako narzędzie limitujące ilość podobnych do siebie osobników zamieszkujących tą samą niszę poprzez redukowanie ich wartości dopasowania.

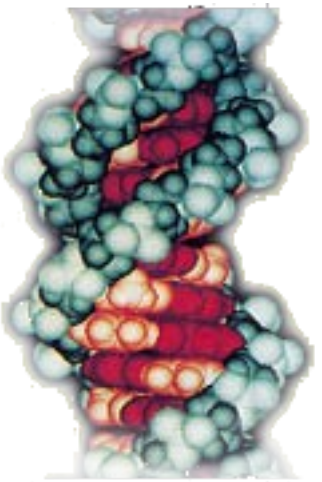
Nie wszystkie osobniki w niszy będą tak samo zredukowane - redukcja jest proporcjonalna do dotychczasowej wartości dopasowania.

Poza numeryczną wartością dopasowania pod uwagę brane są także inne czynniki np:. Długość wzorca, ilość wzorców uczących, przydatność itp.

Osobniki są podobne wtedy, gdy dla tych samych zadanych problemów dają podobne wyniki (podobna fizyczna budowa nie wystarcza)

Niszowy AG

Ścisk-przykład:



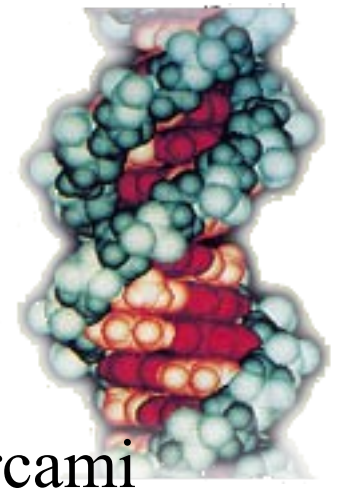
Osobniki A, B i C są podobne, tj. zajmują tą samą niszę

	Ocena	Długość wzorca	Suma ścisku	Ocena z ściskiem
Osobnik A:	*0,901*	*5*	*1*	**0,451**
Osobnik B:	*0,943*	*6*	*0*	**0,943**
Osobnik C:	*0,901*	*7*	*2*	**0,300**

Porównanie	**A**	**B**
Ścisk	1	0
Porównanie	A	C
Ścisk	0	1
Porównanie	B	C
Ścisk	**0**	**1**

Niszowy AG

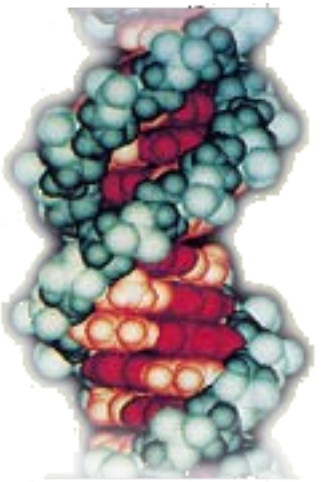
Ścisk



- Ściśk wspomaga rozłączne pokrycie serii danych wzorcami
- Osobniki należące do więcej niż jednej niszy “cierpią” na większy ścisk – są praktycznie eliminowane z rozwiązania.
- Sam ścisk nie gwarantuje powstania pełnego pokrycia przykładów uczących aby zwiększyć szansę powstawania takich układów w trakcie wyznaczania ścisku każdej parze osobników nie kolidujących ze sobą doliczany jest mały (ok 1%) bonus do wartości dopasowania
- Dodatkowo do wyliczenia ścisku uwzględnia się fakt czy jeden z porównywanych wzorców nie współgra z najlepszymi osobnikami w populacji np.:

Niszowy AG

Ścisk

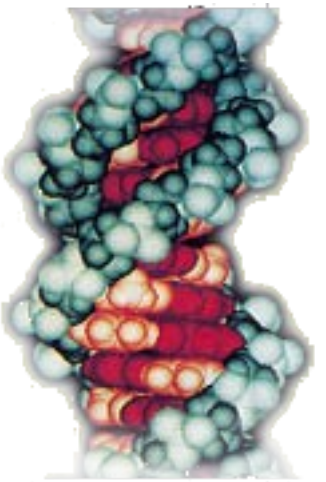


	Pokrycie danych uczących	początkowa wartość dopasowania	Ścisk	Min. ścisku osobników dominujących	Ścisk po korekcie	Końcowe dopasowanie
A	000010000	0,001	2	3	1	0,001
B	000111000	0,003	3	1	3,1	0,001
C	111100000	0,004	1	brak	1	0,004
D	000001111	0,004	1	brak	1	0,004

Po korekcie osobnik A pozostaje w uprzywilejowanej puli osobników chronionych na szczycie populacji

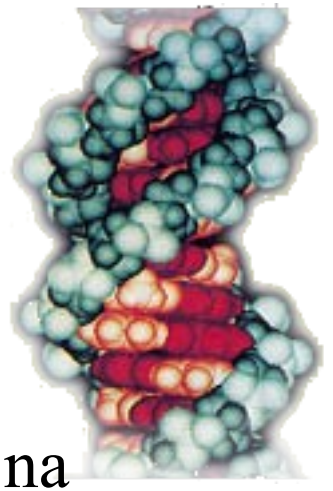
Niszowy AG

Uwagi



- Algorytm wychodzi od losowej populacji
- Po osiągnięciu pierwszych wyników zawsze można oczekiwać lepszych lub zmienić problem i pozwolić populacji przemigrować do nowych nisz
- Nisze pozwalają szukać rozwiązań kompleksowych w jednym przebiegu AG, np.: wyszukać zbiór rozłącznych wzorców opisujących serię danych.
- Większość metodologii wyszukuje po jednym wzorcu na raz, a następnie dokonuje integracji rozwiązania, usuwa sprzeczności
- Niszowy AG nie wymaga integracji rozwiązań, ani usuwania sprzeczności

Problem MONKS



- MONKS sztucznie stworzone zadanie, które powstało na potrzeby testów porównawczych różnych algorytmów – klasyfikatorów. (1991 rok)
- W zadaniu występują roboty opisane 6-cioma zmiennymi:

  x0: head shape $\in$ {round, square, octagon} lub {1,2,3}

  x1: body shape $\in$ {round, square, octagon} lub {1,2,3}

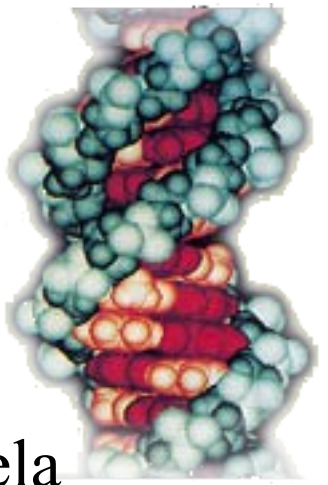
  x2: is smiling $\in$ {yes, no} lub {1,2}

  x3: holding $\in$ {sword, balloon, flag} lub {1,2,3}

  x4: jacket color $\in$ {red, yellow, green, blue} lub {1,2,3,4}

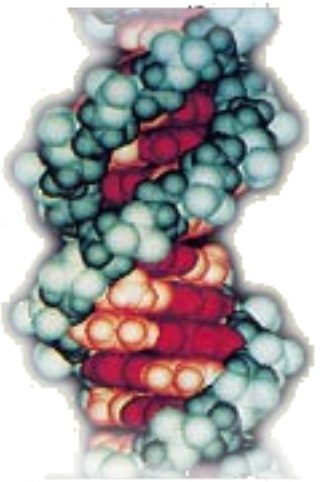
  x5: has tie $\in$ {yes, no} lub {1,2}

Problem MONKS



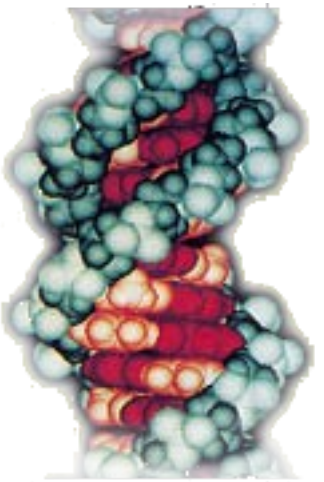
- Klasyfikacja jest binarna t.j. dane uczące i testowe dzielą się na 2 klasy
- Wszystkich możliwych stanów robotów jest 432
- Zadania są postawione w następujący sposób:
  - zbiór danych uczących z rozróżnieniem które przykłady należą do której klasy (ewentualnie 2 zbiory, po jednym na klasę)
  - zbiór testowy - wszystkie 432 przypadki
- 3 zadania testowe:

Problem MONKS



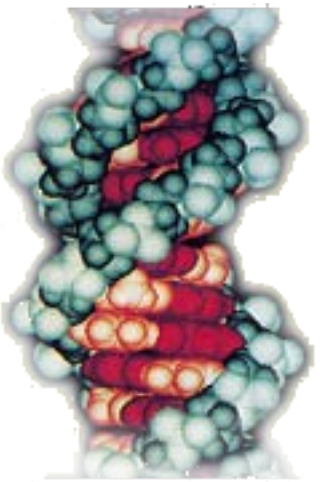
- MONKS 1
  - (head shape = body shape) or (jacket color = red)
  - (x0=x1)or(x4=1)
  - 124 przykłady uczące po 62 na klasę.
  - Dane bez szumu
  - Zadanie rozgrzewka, powinno być łatwe dla wszystkich metod używających normalnej formy dysjunkcyjnej

Problem MONKS



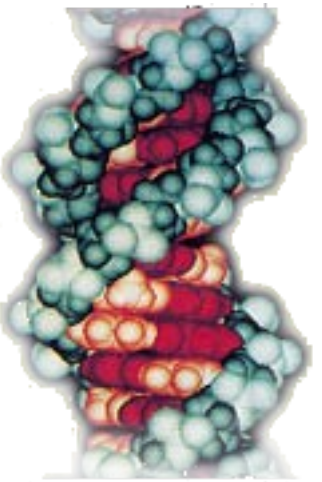
- MONKS 2
  - Dokładnie 2 z 6 atrybutów przyjmuje pierwszą wartość ze swojej dziedziny
  - ????
  - 169 przykłady uczące (klasy 64 i 105)
  - Dane bez szumu
  - Zadanie trudne do zapisania w klasycznych formach (koniunkcyjnej lub dysjunkcyjnej)

Problem MONKS



- MONKS 3
  - (jacket color is green and holding a sword) or (jacket color is not blue and body shape is not octagon)
  - (x4=3 and x3=1) or (x4!=3 and x1 != 3)
  - 122 przykłady uczące (klasy 60 i 62)
  - Dane z 5% szumem (5 wadliwych klasyfikacji)
  - Zadanie trudne ze względu na szum

Opis REX



- Genetyczny Algorytm wyszukiwania formuł matematycznych
- Reguły są zapisane w postaci:

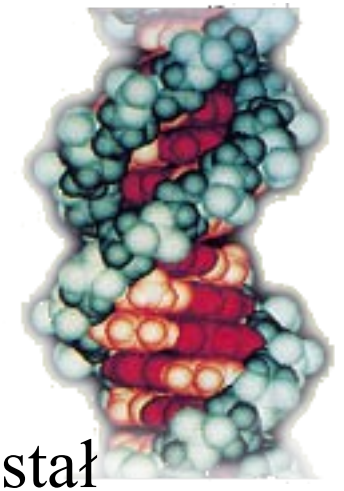
  (W11 and W12 and ... and W1a) xor

  (W21 and W22 and ... and W2b) xor

  ...

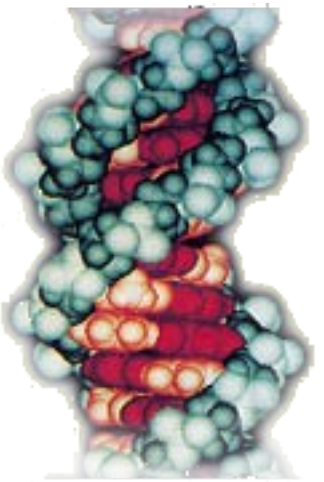
  (Wm1 and Wm2 and ... and Wmn)
- gdzie Każde wyrażenie Wxy ma postać relacji (=,!=,>,<,>=,<=) 2 wyrażeń arytmetycznych zawierających operatory (+,-,*,/), zmienne występujące w problemie oraz stałe ze odgórnie zdefiniowanego zbioru

Opis REX



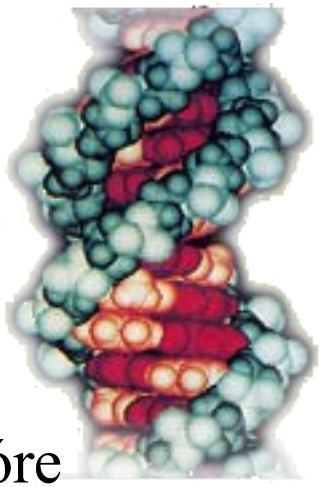
- Problem nawiasów w wyrażeniach arytmetycznych został wyeliminowany poprzez użycie Odwrotnej Notacji Polskiej
- Przykład reguły (z MONKS 1)
  - (x4 x0 / x1!= and 1 x4 =)xor
  - (x1 x0 =)
- Co należy czytać:
  - (x0 /x4 != x1 and x4 = 1)xor
  - (x0 = x1)

Opis REX



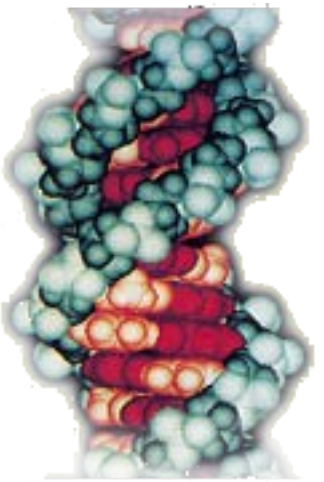
- Osobniki z AG reprezentują sekcje:
  - (Wx1 and Wx2 and ... and Wxn)
- Nisze, ścisk i rozłączne pokrycie całego zbioru uczącego gwarantują poprawne wyrażenie operatora xor
- Dane uczące są podzielone na 2 zbiory: pozytywne przykłady i negatywne przykłady
- Proces genetyczny jest uruchamiany na zadaną z góry ilość iteracji, po każdej zakończonej generacji osobniki są sprawdzane pod kontem możliwego wyniku

Opis REX



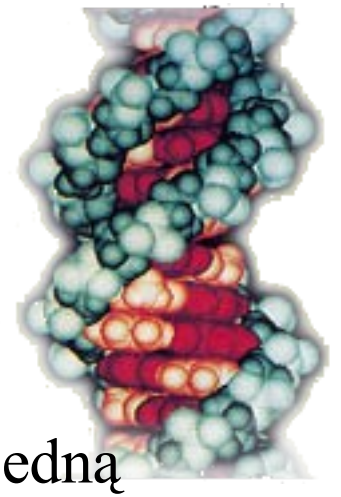
- Wynik to zbiór osobników z minimalnym ściskiem (1.0), które nie dla wszystkich przykładów negatywnych dają logiczną wartość *fałsz*
- System zlicza dla takich osobników ilość przykładów dla których jedna z reguł (zawsze jedna) daję wynik *prawda* i jeśli ilość ta odpowiada rozmiarowi zbioru pozytywnych przykładów ogłasza wynik
- Wraz z takim wynikiem formuły są weryfikowane na zbiorze testowym, ale wynik tej weryfikacji nie jest brany pod uwagę podczas dalszego procesu
- System przechodzi kolejne iteracje AG i wyświetla kolejne, zazwyczaj lepsze wyniki. (krótsze formuły, lepsze uogólnienie na zbiorze testowym)

REX kodowanie



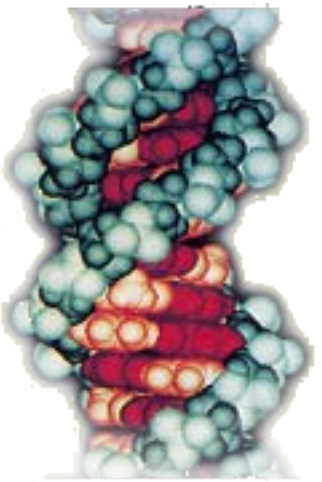
- Kodowanie binarne
- Chromosom jest podzielony na n sekcji po k bitów
- Każda sekcja koduje liczbę $[0,2^k-1]$
  - 0=, 1> , 2>= (}), 3<=({), 4 <, 5!=(!)
  - 6/, 7-, 8+, 9*
  - od 10 do 10 + p - 1 kodowane są indeksy zmiennych ze zbioru uczącego np(p=6 jak w MONKS): 10– x0, 11–x1, 12–x2, 13–x3 , 14–x4, 15-x5
  - od 10+p do 10 + p + t -1 kodowane są stałe, przykład z MONKS(t=4): 16-1.0, 17-2.0, 18-3.0, 19-4.0
  - Reszta liczb ma na stałe przypisaną tą samą nie zerową losową wartość (20-31)

REX kodowanie



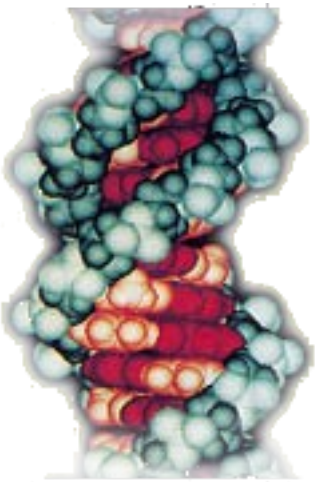
- Wszystkie symbole aż do wystąpienia znaku relacji budują jedną sekcje (Wxy), symbole po ostatniej relacji są ignorowane, np.:
  - x0 x0 = x3x4 < x4 x3 * 3 > x0 4 / * koduje:
  - (x0 = x0) and (x4 < x3) and (3 > x3 * x4)
- Przed ocenieniem lub wyświetleniem osobnika system dokonuje szeregu prostych redukcji (które nie zmieniają binarnej reprezentacji):
  - redukcja nadmiarowych operatorów
    - x1*+x2+*4= redukuje do x1x2+4=
  - redukcja nadmiarowych zmiennych i stałych
    - x2x0x1x3+= redukuje do x0x1x3+=
  - redukcja za krótkich sekcji
    - x1x2+= zostaje usunięte

REX kodowanie



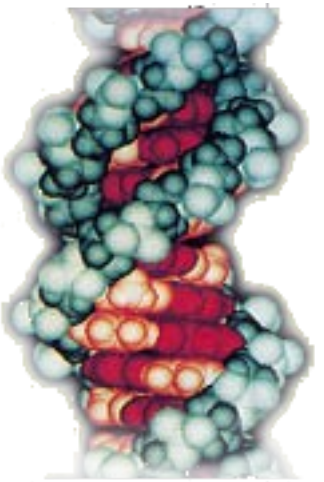
- redukcja tautologii i nieużytecznych sekcji
  - x1x1=x2x2/1= zostanie usunięte
  - x104> w przypadku MONKS zostanie usunięte
  - za tautologie jest uważana sekcja która jest prawdziwa dla wszystkich przykładów z obu zbiorów uczących
- normalizacja sekcji
  - x1x2< zamienia na x2x1> (analogicznie <=)
  - x1x0= zamienia x0x1= (analogicznie !=)
  - x2x1+x3= zamienia na x1x2+x3= (analogicznie *)
  - normalizacja wspomaga redukcje (nast. slajd)

REX kodowanie



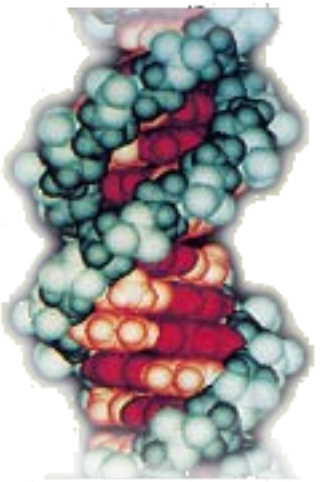
- redukcja identycznych lub pokrywających się sekcji
  - x0x1=x0x1= redukuje się do x0x1=
  - x0x1=x0x1} redukuje się do x0x1= (analogicznie {)
  - x0x1!x0x1> redukuje się do x0x1> (analogicznie <)
  - x0x1!x1x0> też będzie zredukowane
- Wyniki redukcji nie są wpisywane do chromosomu, ale osobniki z dużym redukowanym narzutem mają ograniczone możliwości rozwoju i utrudniają działanie AG.
- Za każdy znak usunięty podczas redukcji zmniejsza **współczynnik redukcji (WR)**, który jest proporcjonalny do ilości znaków usuniętych. Np. chromosom bez redukcji (WR=1.0), chromosom w połowie zredukowany (WR=0.5)

REX funkcja oceny



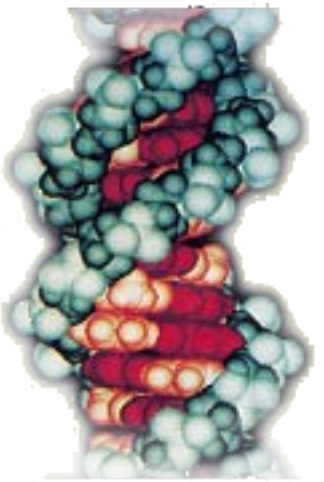
- Dla ocenianego osobnika wyznacza się:
  - procentowe pokrycie zbioru pozytywnych przykładów (**PP**)
  - ilość niepoprawnych przypadków w zbiorze negatywnych przykładów (**NP**)
  - Współczynnik redukcji (**WR**)
  - Współczynnik wartości (**WW**):
    - każda sekcja dodaje do WW w zależności od procentowej ilości użytych zmiennych (**uz**) i typu relacji:
      - (3.0*1.5*uz) dla =
      - (0.6*1.5*uz) dla <>
      - (0.4*1.5*uz) dla {}
      - (0.3*1.5*uz) dla !

REX funkcja oceny



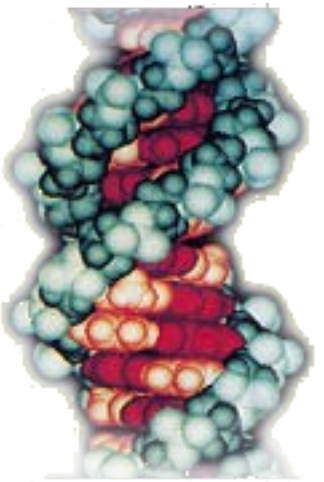
- Przykład (z MONKS 6 zmiennych):
  - x1x2=x13x1*>
  - 3.0*1.5*0.33+0.6*1.5*0.16 = 1.485 + 0.144 = 1.629
- Wartość oceny osobnika to:
  - (PP*WR*WW)/(2*NP)

REX wyniki



- Test MONKS1 (wersja 1 REX)
  - Bez zbioru negatywnych przykładów !
  - Za to wszystkie pozytywne przykłady były w zbiorze uczącym
  - Parametry:
    - maksymalna ilość symboli na stosie RPN 30 (5bitów każdy)
    - populacja 100 osobników
    - ilość generacji 3000
    - czas około 2 minut

REX wyniki

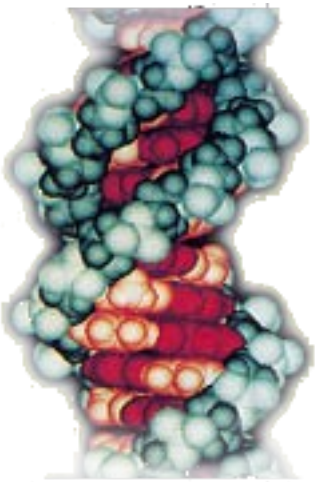


- Test MONKS1 (wersja 1 REX)
  - Wyniki:
    - x1x0!x4x5x0+>x4x0*x0=
    - x1x0=1.0x1*x0=1.0x0*x1=
  - Oczywiste redukcje, których nie umie wykonać REX

    (x0!=x1 and x4=1) xor

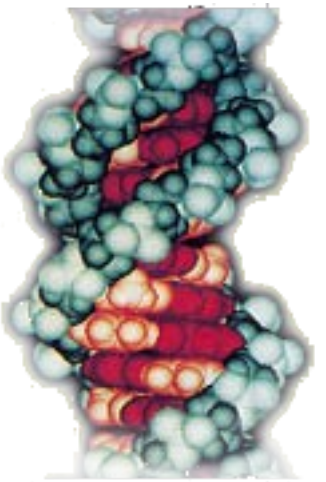
    (x0=x1)
  - To jest formuła równoważna tej podanej w MONKS1
  - 100% poprawności, czytelny i krótki wynik
  - 100% powtarzalność poprawnego wyniku

REX wyniki



- Test MONKS1 (wersja 2 REX - aktualna)
  - Test zgodny z warunkami MONKS 1
  - Dane uczące ze zbiorów Thrun'a
  - Parametry:
    - maksymalna ilość symboli na stosie RPN 20 (5bitów każdy)
    - populacja 200 osobników
    - ilość generacji 5000, średnio dobre wyniki pojawiają się już około 2500 generacji
    - czas około 5 minut

REX wyniki

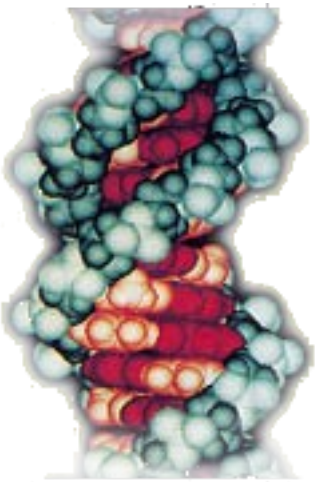


- Test MONKS1 (wersja 2 REX - aktualna)
  - Wyniki:
    - x5x5/x4=x4x0/x1!1.0x4=
    - x5x1/x5*x0=x1x0=
  - Oczywiste redukcje, których nie umie wykonać REX

    (x4=1.0 and x0!=x1) xor

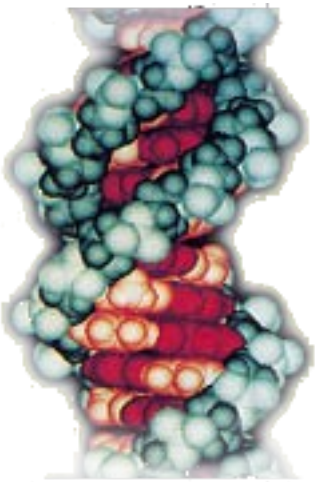
    (x0=x1)
  - To jest formuła równoważna tej podanej w MONKS1
  - 100% poprawności, czytelny i krótki wynik
  - 40% powtarzalność poprawnego wyniku
  - Pozostałe 60% bardzo blisko rzędu 95% poprawności (to można jeszcze dość łatwo poprawić – prace w toku)

REX wyniki



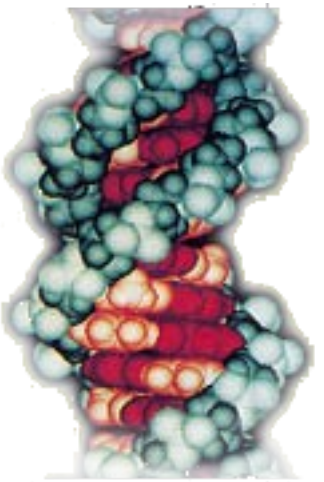
- Test MONKS2
  - Test z pełnymi zbiorami uczącymi (oba, brak problemu uogólniania)
  - Parametry:
    - maksymalna ilość symboli na stosie RPN 55 (5bitów każdy)
    - populacja 500 osobników
    - ilość generacji 20 000, średnio dobre wyniki pojawiają się dopiero pod koniec
    - czas około 4 godzin

REX wyniki



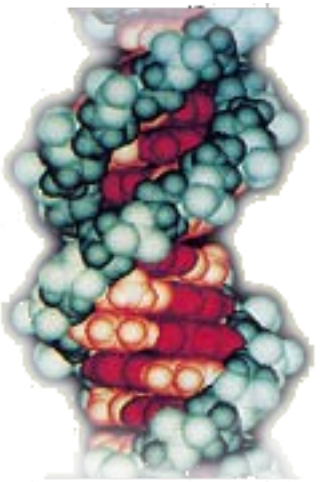
- Test MONKS2
  - Wynik (przykładowy, dość krótki):
    - 13 osobników (13 XOR)
    - 101 sekcji w sumie w tym 5 negacji
  - Pozostałe wyniki – zbliżona ilość sekcji, do 19 osobników
  - 100% poprawności, czytelny ale długi wynik
  - 70% powtarzalność poprawnego wyniku
  - Pozostałe 30% bardzo blisko rzędu 98% poprawności (to można jeszcze dość łatwo poprawić, ale wpierw testy na mniejszych zbiorach uczących – prace w toku)

REX wyniki



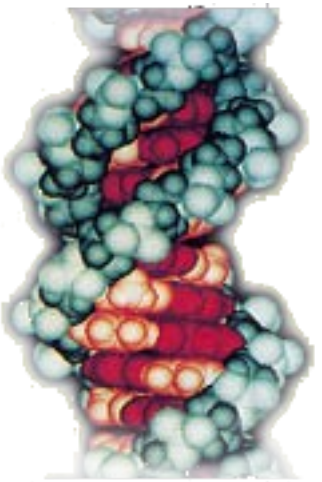
- Test MONKS 3
  - Test prawie zgodny z warunkami MONKS 3 ale bez szumu
  - Dane uczące ze zbiorów Thrun'a po poprawieniu szumu
  - Parametry:
    - maksymalna ilość symboli na stosie RPN 30 (5bitów każdy)
    - populacja 100 osobników
    - ilość generacji 20 000, średnio dobre wyniki pojawiają się już około 8000 generacji
    - czas około 10 minut

REX wyniki



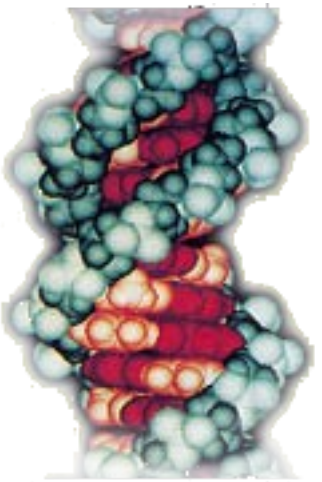
- Test MONKS 3
  - Wyniki:
    - x303x4/}x0x1x0*/x4=x5x1}x4x1=
    - 01x5-x1=x1x5-x0*x0=x403}x5x5/x1=
    - 02x1/x5=x5x1-x0*x0=x403}x5x4*x4=
    - 01x5/x1=x0x1x0*/x5=x403}x5x1=
  - 100% poprawności (zbiór walidacyjny)
  - czytelny i dość krótki wynik
  - 40% powtarzalność poprawnego wyniku
  - 40% bardzo blisko rzędu 99% poprawności
  - 20% dokładność rzędu 95% na zbiorze walidującym

REX dalsze prace



- Drobne zmiany zwiększające powtarzalność wyniku dla MONKS 1 i 3
- Testy MONKS 2 w warunkach zgodnych z założeniami problemu (mniejsze zbiory uczące)
- Testy MONKS 3 z szumem

To już wszystko!



Dziękuję za uwagę,
proszę o pytania i komentarze.

Marcin Borkowski
marcinbo@mini.pw.edu.pl