Inteligencja obliczeniowa

Laboratorium 2: Algorytmy genetyczne

Zadanie 1 *(1 pkt)*
W problemie plecakowym pytamy, jakie przedmioty wziąć do plecaka o ograniczonej objętości, by ich wartość była najwyższa. Załóżmy, że jedziemy na obóz, ale w plecaku zmieścimy przedmioty o łącznej wadze 20. Do wyboru mamy nastepujące przemioty:

ITEM	SURVIVALPOINTS	WEIGHT
pocketknife	10.00	1.00
beans	20.00	5.00
potatoes	15.00	10.00
unions	2.00	1.00
sleeping bag	30.00	7.00
rope	10.00	5.00
compass	30.00	1.00

Jakie przedmioty należy wziąć? Korzystając z pakietu R i poniższych wskazówek rozwiąż ten problem za pomocą algorytmu genetycznego.

a) Instalacja (jednorazowo) i załadowanie biblioteki
```
install.packages(genalg)
library(genalg)
```

b) Dodawanie zbioru danych i limitu plecaka
```
dataset <- data.frame(
      item = c("pocketknife", "beans", "potatoes", "unions",
               "sleeping bag", "rope", "compass"),
      survivalpoints = c(10, 20, 15, 2, 30, 10, 30),
      weight = c(1, 5, 10, 1, 7, 5, 1))

weightlimit <- 20
```

c) Chromosomy (rozwiązania) to ciągi 7-bitów. Dla każdego z 7 przedmiotów wybieramy 0 (nie bierzemy tego przedmiotu) lub 1 (bierzemy ten przedmiot do plecaka). Jakim przedmiotom odpowiada ciąg 1001100? Ile są warte wszystkie przedmioty? Sprawdź za pomocą poniższych komend.

```
chromosome = c(1, 0, 0, 1, 1, 0, 0)
dataset[chromosome == 1, ]

cat(chromosome %*% dataset$survivalpoints)
```

d) Najważniejszy etap to zdefiniowanie funkcji fitness. Sprawdza i zwraca ile ważą przedmioty dla rozwiązania x. Jeśli przekroczyły limit plecaka to zwraca 0.

```
fitnessFunc <- function(x) {
     current_solution_survivalpoints <- x %*% dataset$survivalpoints
     current_solution_weight <- x %*% dataset$weight

     if (current_solution_weight > weightlimit)
          return(0) else return(-current_solution_survivalpoints)
}
```

e) Definiujemy algorytm genetyczny wprowadzając odpowiednie parametry i uruchamiamy go. Jakie jest najlepsze rozwiązanie?

```
GAmodel <- rbga.bin(size = 7, popSize = 200, iters = 100,
           mutationChance = 0.01, elitism = T, evalFunc = fitnessFunc)
summary(GAmodel, echo=TRUE)
```

Zadanie 2 *(2 pkt)*
Twoim zadaniem jest wykorzystać algorytm genetyczny (jak w zadaniu 1, punkty b-e) do znajdowania drogi w labiryncie. Skorzystaj z poniższego labiryntu zakodowanego jako macierz dwuwymiarowa 11 x 8 kratek:

```
#,#,#,#,#,#,#,#,#,#,#,#
#,S,_,_,_,#,_,_,_,_,#
#,_,#,#,_,_,_,#,#,_,#
#,_,#,#,#,_,#,#,#,_,#
#,_,_,_,#,_,_,_,#,#,#
#,_,#,_,#,_,#,_,_,_,#
#,_,#,_,#,_,_,_,#,X,#
#,#,#,#,#,#,#,#,#,#,#
```

Krzyżyki # oznaczają ścianę labiryntu. Nie można na nie wchodzić. Podkreślniki oznaczają puste pole, po którym można chodzić. S oznacza punkt startowy, a X punkt docelowy. Naszym zadaniem jest przejść od punktu S do punktu X jedynie po pustych polach poruszając się w 4 standardowe kierunki świata.

Należy sobie odpowiedzieć na dwa bardzo ważne pytania:
1) Co będzie chromosomem (rozwiązaniem)? Jak je zakodować i jaką ma mieć długość?
2) Jak zmierzymy czy dany chromosom jest dobrym rozwiązaniem? Jak ma działać funkcja fitness?

Zanim rzucisz okiem na kolejną stronę zastanów się nad oboma pytaniami.

Ad. 1)
Rozwiązanie to droga przejścia przez labirynt. Ponieważ możemy poruszać się w cztery strony to rozwiązaniem będzie sekwencja ruchów (prawo, lewo, góra, dół) np.
PPPDPPGL
Owe kierunki można zakodować jako dwa bity np. (G=00,P=01,D=11,L=10). Wówczas mamy: 0101011101010010.
Liczbę ruchów (długość chromosomu) można spróbować ustalić na około długość+szerokość labiryntu (czasem potrzebne są korekty), np. 20 ruchów, czyli 40 bitów.
Uwaga! Zmiana kodowania na zerojedynkowe nie jest konieczna, ale to standard, więc jest mile widziana.

Ad. 2)
By sprawdzić czy rozwiązanie jest dobre, należy je zasymulować na labiryncie. Wykonujemy po kolei ruchy z chromosomu. W momencie gdy uderzamy w ścianę (najeżdżamy na #), kończymy poruszanie się i mierzymy jak daleko zaszliśmy. Im bliżej X jesteśmy, tym lepiej. Można zmierzyć odległość od X, i zwrócić ją jako wartość funkcji fitness.

Jeśli X ma współrzędne (9,6), a my rozbiliśmy się na ścianie (5,1) to jak zmierzyć odległość? Zastosuj metrykę taksówkową lub euklidesową.

Jesli dotarliśmy do X bez wchodzenia na ścianę to kończymy działanie i zwracamy najlepszą wartość (czyli w naszym przypadku 0).

***Zadanie 3** *(dla chętnych, 2 dodatkowe punkty)*
Przypomnij sobie jaką postać ma formuła logiczna w koniunkcyjnej postaci normalnej o klauzalach co najwyżej długości 3 (w skrócie: 3-CNF). Czy poniższa formuła o 7 klauzulach w postaci 3-CNF jest spełnialna (tzn. czy istnieje podstawienie zerojedynkowe, które da na wyjściu prawdę)?

$\varphi = (\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (x_2 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee x_2 \vee x_3)$
Sprawdź dla podstawienia $x_1=1$, $x_2=0$, $x_3=1$, $x_4=1$ oraz dla podstawienia $x_1=1$, $x_2=1$, $x_3=1$, $x_4=1$.

Na stronie http://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html (a także na stronie http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html) znajdują się spakowane zakodowane formuły 3-CNF w formacie *dimcas cnf*, wraz z objaśnieniem formatu. Powyższa formuła φ w danym formacie byłaby tabelą postaci:

-1	2	4	0
-2	3	4	0
1	-3	4	0
1	-2	-4	0
2	-3	-4	0
-1	3	-4	0
1	2	3	0

Widać, że indeksy iksów odpowiadają numerom w tabeli, a minusy odpowiadają negacji.

Problem 3-SAT, który odpowiada na pytanie „czy dana formuła 3-CNF jest spełnialna?” jest NP-zupełny, dlatego nie istnieją efektywne algorytmy rozwiązujące ten problem w wielomianowym czasie. Można spróbować za to rozwiązać ten problem za pomocą algorytmu genetycznego. Korzystając z pakietu R i biblioteki *genalg* znajdź podstawienie dla formuły z treści zadania, a następnie również dla jednej z formuł z zamieszczonych na wczesniej wymienionych stronach.

Podpowiedzi:
a) chromosomem będzie podstawienie do formuły, więc trzeba wiedzieć ile zmiennych ma formuła (czy ta informacja jest w formacie dimacs cnf?)
b) funkcją fitness jest liczba klauzul spełnionych po zastosowaniu podstawienia, czyli dla naszego przykładu
fitness(φ, [1,0,1,1]) = 6
fitness(φ, [1,1,1,1]) = 7 (φ spełnialna, [1,1,1,1] to idealne podstawienie, każde 7 klauzul zwraca 1)