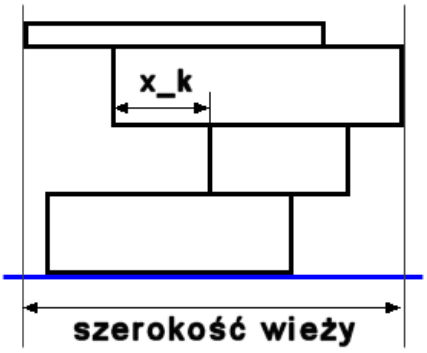
Mini-Projekt Ver. 1.1	Wieża z klocków	Algorytmy genetyczne 2011

Ostateczny termin realizacji projektu

Projekty przesyłamy mailem do 14 lipca 2012 roku. Do 23:59.

Opis problemu

Do dyspozycji mamy N klocków. Każdy klocek ma inne rozmiary opisane przez w_k oraz h_k. Klocki można obracać o 90°. Po obróceniu wartości w_k oraz h_k zamieniają się miejscami. Klocki układamy jeden na drugim do momentu aż któryś z nich nie spadnie lub cała wieża nie ulegnie zawaleniu. Nie można układać jednego klocka na więcej niż jednym klocku. Zadanie polega na znalezieniu takiej kolejności układania klocków, aby ostatnia stabilna konfiguracja wykorzystała jak najwięcej z dostępnych klocków, a jednocześnie, aby szerokość całej wieży (rzuty na podstawę najdalej wysuniętych klocków na lewo i na prawo) była jak największa (rys.1).



Format danych wejściowych.

Dane wejściowe do programu przechowywane są w pliku tekstowym, którego struktura jest następująca:

w pierwszej linijce znajduje się tylko jedna liczba: N – jest to liczba klocków

następne N linijek zawiera opisy każdego klocka, w każdej linijce znajduje się X liczb całkowitych, które oznaczają kolejno:

- k – numer klocka (klocki numerujemy od 0)
- w_k oraz h_k – szerokości oraz wysokość k-tego klocka

Format danych wyjściowych.

Rozwiązanie powinno zostać zapisane do pliku tekstowego.
Plik powinien zawierać dokładnie N linijek. Każda linijka powinna zawierać opis pojedynczego klocka. Pierwsza linijka opisuje klocek wykładany jako pierwszy, druga linijka klocek, który kładziemy na tym pierwszym itd…
Kolejne liczby w każdej linijce powinny być oddzielone spacjami, a ich znaczenie jest następujące:

- k – numer wykładanego klocka
- O – informacja czy klocek jest obrócony czy nie, 1 – oznacza, że klocek został obrócony, 0 – że nie był obracany
- wartości x_k przesunięcie lewego dolnego narożnika klocka który kładziemy, względem lewego górnego narożnika klocka na który kładziemy (rys.1)

Plik musi zawierać dokładnie N linijek. Klocki, których nie da się już poprawnie ułożyć mogą mieć dowolne wartości O i x_k.

Założenia wstępne.

Zakładamy, że N będzie się mogło zmieniać pomiędzy 3 a 300, w_k oraz h_k mogą się zmieniać między 5 a 20.

Projekt.

Każda osoba pisze własny programik (można się oprzeć na przykładach, które już ćwiczyliśmy), który ma rozwiązywać postawione zadanie. Każdy sam musi zdecydować (zgadnąć, potestować, poczytać czy poczekać na natchnienie) jakich metod użyć, jakich parametrów krzyżowania i mutacji oraz jakiej liczności populacji.

Przy układaniu klocków można wykorzystywać jedynie wiedzę o klocku bezpośrednio poprzedzającym układany klocek.

Projekt można realizować w dowolnym systemie i środowisku. Jednak ostateczną wersję do oceny proszę przysłać skompilowaną pod wxDev-C++ z flagą optymalizacji O3 (bez innych optymalizacji). Cały katalog z projektem należy nazwać własnym nazwiskiem i zzipować do pojedynczego pliku o nazwie „nazwisko.zip". W katalogu projektu należy utworzyć podkatalog o nazwie EXE, a w nim umieścić skompilowaną wersję programu nazwaną „nazwisko.ex". Proszę zwrócić uwagę na brak ostatniej literki „e" w nazwie pliku. Chodzi o to, żeby programy pocztowe nie protestowały z uwagi na zawartość pliku wykonywalnego.

Ocena projektu

1. Każdy program zostanie uruchomiony dziesięciokrotnie z różnymi parametrami (dla wszystkich programów jednakowymi).
2. Dla każdego uruchomienia każdego programu policzony zostanie procent użytych klocków (ocena A).
3. Dla każdego uruchomienia programu policzony zostanie stosunek szerokości najszerszego klocka do uzyskanej szerokości wieży w danym zestawie (ocena B).
4. Wynik każdego uruchomienia programu reprezentować będzie pojedynczy punkt w układzie o osiach A,B.
5. Dla każdego programu policzony zostanie środek ciężkości punktów.
6. Środki ciężkości wszystkich programów zostaną umieszczone na wykresie A,B.
7. Rozwiązania niezdominowane (w sensie Pareto) zdobędą 15 punktów.
8. Rozwiązania odległe od nich o nie więcej niż 10% najlepszego rozwiązania w kierunku każdej osi uzyskają 13 punktów.
9. Rozwiązania z drugiego frontu Pareto zdobędą 11 punktów.
10. Rozwiązania odległe od nich o nie więcej niż 10% najlepszego rozwiązania drugiego frontu Pareto w kierunku każdej osi uzyskają 9 punktów.
11. Pozostałe rozwiązanie o ile będą poprawne otrzymają 7 punktów.
12. Programy z istotnymi błędami implementacji uzyskają od 0 do 5 punktów (niezależnie od wyniku).
13. Pozostałe 15 punktów przyznawane będzie za:
    a. Kodowanie 5 punktów
    b. Dokumentację 10 punktów
14. Dokumentacja powinna zawierać co najmniej:

a. Opis zastosowanego algorytmu genetycznego
b. Opis rozwiązania problemu dwukryterialności
c. Szczegółowy opis kodowania
d. Szczegółowy opis inicjalizacji populacji bazowej
e. Opis użytej funkcji dostosowania
f. Opis użytych metod krzyżowania, mutacji oraz selekcji
g. Krótkie uzasadnienie użycia tych a nie innych metod
h. Ewentualne podsumowanie oraz wnioski

Jak sprawdzić czy klocek spadnie lub wieża się przewróci?

Dowolny przedmiot przewraca się, gdy rzut jego środka ciężkości wychodzi poza podstawę na której się wspiera. Niech i numeruje klocki. i=0 oznacza klocek położony najniżej. Algorytm sprawdzania mógłby wyglądać następująco:

1. Kładziemy klocek i=0, którego nie sprawdzamy
2. Dodajemy kolejny klocek i=i+1
3. n=i
4. Liczymy środek ciężkości sumy klocków od n do i.
5. Jeżeli rzut środka ciężkości klocków od n do i pada poza podstawą klocka n, to klocki spadną i algorytm kończy pracę stwierdzeniem, że dało się ułożyć i klocków. KONIEC!
6. n=n-1
7. Jeżeli n>0 idziemy do punktu 4.
8. Udało się ułożyć i-ty klocek. Idziemy do punktu 2.

Uwaga dotycząca czasu wykonania.

Programy będą uruchamiane na komputerze z procesorem **Intel Core 2 Duo P8700 z zegarem 2.53GHz**. Pojedyncze uruchomienie nie powinno trwać dłużej niż 2 minuty.

Gotowe projekty, jak również wszelkie pytania proszę wysyłać na:
tarasiuk@agh.edu.pl