

# Căutare aleatoare și căutare locală

## 1 CERINȚE

---

1. Să se implementeze o metodă de *căutare aleatoare* (random search) pentru problema rucsacului.
  - Să se genereze o soluție aleatoare și să se verifice dacă este validă.
  - Să se determine calitatea soluției generate.
  - Pentru  $k$  soluții generate aleator, să se determine cea mai bună soluție.
2. Să se implementeze una din cele trei variante ale metodei Hill-Climbing pentru problema rucsacului (cf numar din grupa):
  - a. Random Hill-Climbing
  - b. Steepest Ascent Hill-Climbing
  - c. Next Ascent Hill-Climbing

Efectuați experimente pe cele două instanțe de problema rucsacului primite.

Observații:

- ✓ Aplicația trebuie să fie modularizată și să permită parametrizarea algoritmului și afișarea soluției.
- ✓ Aplicația trebuie să permită rularea algoritmului pe date de test specificate în program cât și încărcarea datelor din fișier.
- ✓ Testați algoritmul pentru o instanță manual setată de mărime mică pentru a urmări performanța. De asemenea, testați algoritmul în diverse variante pentru comparații pe cele 2 instanțe primite în cadrul laboratorului.
- ✓ Rezultatele experimentelor trebuie salvate (cu indicarea setărilor folosite: algoritmul, valori parametri, număr rulări – minim 10 rulări/configurație, calitatea soluției – best/avg).

## 2 TERMEN DE PREDARE

---

- **Lab 2**

**Total Punctaj Tema 1 = 100p**

### 3 PREDAREA TEMEI PRIN MS TEAMS

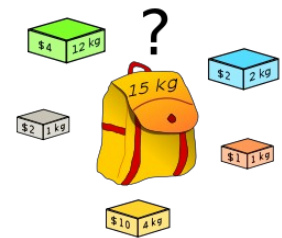
Incarcati urmatoarele fisiere **INAINTE** de a incepe lab-ul in care este setat termenul de predare:

1. O arhiva cu codul sursa
2. Un document (Word/PDF) care sa contina:
  - ✓ Descrierea pe scurt a algoritmului implementat (pseudocod) si principalelor componente (reprezentare solutie, functie de fitness, operatori, etc)
  - ✓ Indicarea parametrilor algoritmului
  - ✓ Tabele/grafice cu rezultatele obtinute (comparatii pentru cel putin 3 seturi de valori ale parametrilor pentru fiecare instanta de problema)
  - ✓ Analiza rezultatelor

### 4 PROBLEMA RUCSACULUI

- $n$  obiecte, fiecare obiect are o valoare ( $v$ ) și o greutate ( $w$ )
- *Obiectiv: puneți în rucsac valoarea maximă fără a depăși greutatea maximă admisă  $W$*
- $x_i = 1$  înseamnă obiectul  $i$  este pus în rucsac
- $x_i = 0$  înseamnă obiectul  $i$  nu este pus în rucsac

$$\begin{aligned} &\text{maximize} \sum_{i=1}^n v_i x_i \\ &\text{subject to} \sum_{i=1}^n w_i x_i \leq W \text{ and } x_i \in \{0, 1\}. \end{aligned}$$



### 5 RANDOM HILL-CLIMBING (RHC)

1. Se selectează un punct aleator  $\mathbf{c}$  (*current*) în spațiul de căutare
2. Se alege un punct  $\mathbf{x}$  din vecinătatea lui  $\mathbf{c}$ :  $\mathbf{N}(\mathbf{c})$ .  
Dacă  $eval(\mathbf{x})$  este mai bun decât  $eval(\mathbf{c})$  atunci  $\mathbf{c} = \mathbf{x}$ .
3. Repetă pasul 2 până când un număr maxim de evaluări se atinge.
4. **Returnează  $\mathbf{c}$ .**

### 6 STEEPEST ASCENT HILL-CLIMBING (SAHC)

1. Se selectează un punct aleator  $\mathbf{c}$  (*current hilltop*) în spațiul de căutare.
2. Se determină toate punctele  $\mathbf{x}$  din vecinătatea lui  $\mathbf{c}$ :  $\mathbf{x} \in \mathbf{N}(\mathbf{c})$
3. Dacă oricare  $\mathbf{x} \in \mathbf{N}(\mathbf{c})$  are un fitness mai bun decât  $\mathbf{c}$  atunci  $\mathbf{c} = \mathbf{x}$ , unde  $\mathbf{x}$  are cea mai bună valoare  $eval(\mathbf{x})$ .

4. Dacă nici un punct  $x \in N(c)$  nu are un fitness mai bun decât  $c$ , se salvează  $c$  și se trece la **pasul 1**. Altfel, se trece la **pasul 2** cu noul  $c$ .
5. După un număr maxim de evaluări, se returnează cel mai bun  $c$  (hilltop).

## 7 NEXT ASCENT HILL-CLIMBING (NAHC)

---

1. Se selectează un punct aleator  $c$  (*current hilltop*) în spațiul de căutare.
2. Se consideră pe rând vecinii  $x$  ai punctului  $c$ . Dacă  $eval(x)$  este mai bun decât  $eval(c)$ , atunci  $c=x$  și nu se mai evaluează restul vecinilor lui  $c$ . Se continuă pasul 2 cu noul  $c$  și se consideră vecinii lui  $c$  mai departe (pornind din același punct din vecinătate unde s-a rămas cu vechiul  $c$ ).
3. Dacă nici un vecin  $x$  al punctului  $c$  nu duce la o evaluare mai bună, se salvează  $c$  și se continuă procesul de la pasul 1.
4. După un număr maxim de evaluări, se returnează cel mai bun  $c$  (hilltop).

## 8 REFERINȚE

---

J.D. Schaffer and L.J. Eshelman. "On crossover as an evolutionary viable strategy". In R.K. Belew and L.B. Booker, editors. *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 61-68, Morgan Kaufmann, 1991.

Melanie Mitchell and Stephanie Forrest. Fitness Landscapes: Royal Road Functions in., Back T., Fogel D. and Michalewicz, Z. (Eds.), *Handbook of Evolutionary Computation*. Oxford: Oxford University Press., 1997.