Algoritmi Avansați
Laborator 4

Gabriel Majeri

Introducere

Unele dintre cele mai simple probleme de optimizare (convexă) întâlnite în practică sunt cele de programare liniară [1] (în care variabilele sunt numere reale, iar constrângerile și funcția obiectiv sunt liniare/de gradul 1).

În acest laborator, vom încerca să **modelăm** și să **rezolvăm** câteva probleme de programare liniară, dar **nu** vom implementa noi algoritmul care găsește soluția optimă pentru un program liniar (nu este dificil, dar ar fi mult de scris). În schimb, vom învăța să instalăm și să configurăm o bibliotecă existentă.

Dacă vreți să înțelegeți mai bine ce sunt problemele de programare liniară, unde le întâlnim în practică și cum le putem rezolva folosind algoritmul simplex, vă recomand să consultați cursul 4 sau materialele recomandate și în seminarul 3 [2, 3, 4].

Setup

- Pentru **Python**: biblioteca SciPy este inclusă în majoritatea distribuțiilor de calcul științific în Python (de exemplu, Anaconda) și are suport pentru rezolvarea problemelor de optimizare liniară. Puteți urmării instrucțiunile din acest ghid sau cele de pe site-ul SciPy pentru a vedea cum se poate configura și utiliza această bibliotecă. Pe scurt, ce aveți de făcut este să:

    1. Creați un nou *virtual environment* în care să instalați pachetul `scipy`. IDE-urile ca PyCharm vă permit să faceți asta automat când creați un nou proiect, altfel puteți să o faceți din linia de comandă.

       Nu e nevoie să creați un *venv* dacă aveți deja instalat SciPy global (de exemplu, dacă folosiți Anaconda).

2. Activați *virtual environment*-ul (dacă nu ați făcut-o deja).
3. Instalați pachetul `scipy` cu pip:

   ```
   python -m pip install --user scipy
   ```

4. Importați pachetul `scipy.optimize.linprog` și începeți să modelați problema de interes:

   ```
   from scipy.optimize import linprog
   ```

5. Urmăriți pașii din acest ghid pentru a vedea cum puteți modela o problemă de programare liniară în Python.

- Pentru **C++**: puteți folosi biblioteca OR-Tools, dezvoltată de Google.

  1. Descărcați și instalați binarele OR-Tools de pe site-ul oficial.

     Pentru Windows, utilizarea OR-Tools este suportată doar cu Visual Studio. Dacă nu aveți Visual Studio instalat sau nu vreți să-l folosiți, cel mai bine ar fi să încercați să rezolvați laboratorul de astăzi în Python (unde e mult mai simplu să instalezi o bibliotecă externă).
  2. Configurați build system-ul vostru să link-uiască programul vostru cu biblioteca OR-Tools. Pentru Visual Studio pe Windows, puteți urmării acești pași. Pe Mac OS/Linux, puteți urmării pașii din ghidul oficial.
  3. Urmați ghidul introductiv oficial pentru a vedea cum puteți folosi această bibliotecă.

Exerciții

1. Modelați și rezolvați problema tâmplarului, dată ca exemplu în video-ul de *Intro to Linear Programming* [2]. Cu alte cuvinte, rezolvați următorul programul liniar:

$$\begin{gathered}\max\ 180x + 200y \\ 5x + 4y \leq 80 \\ 10x + 20y \leq 200 \\ x \geq 0 \\ y \geq 0\end{gathered}$$

Folosiți biblioteca pe care ați instalat-o.

2. Modelați și rezolvați o instanță a problemei 3-CNF (așa cum a fost descrisă în tema 1 de la curs) folosind programarea liniară.

De exemplu, pentru expresia logică:

$$(x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee x_3 \vee x_5) \wedge (x_2 \vee x_4 \vee x_5)$$

Programul liniar corespunzător ar fi:

$$\begin{aligned} \min\ & x_1 + x_2 + x_3 + x_4 + x_5 \\ & x_1 + x_2 + x_4 \geq 1 \\ & x_1 + x_3 + x_5 \geq 1 \\ & x_2 + x_4 + x_5 \geq 1 \\ & 0 \leq x_1 \leq 1 \\ & 0 \leq x_2 \leq 1 \\ & 0 \leq x_3 \leq 1 \\ & 0 \leq x_4 \leq 1 \\ & 0 \leq x_5 \leq 1 \end{aligned}$$

După ce rezolvați problema, interpretați orice variabilă x_i care are valoarea mai mare decât $\frac{1}{3}$ ca fiind *true*, și orice variabilă care are o valoare mai mică decât $\frac{1}{3}$ ca fiind *false*. Astfel, veți obține o soluție 3-aproximativă la problema inițială 3-CNF.

Referințe

[1] Wikipedia contributors, *Linear programming*, URL: https://en.wikipedia.org/wiki/Linear_programming.

[2] Trefor Bazett, *Intro to Linear Programming and the Simplex Method*, URL: https://www.youtube.com/watch?v=K7TL5NMlKIk.

[3] MIT OpenCourseWare, *15. Linear Programming: LP, reductions, Simplex*, URL: https://www.youtube.com/watch?v=WwMz2fJwUCg.

[4] The Organic Chemistry Tutor, *Linear Programming*, URL: https://www.youtube.com/watch?v=Bzzqx1F23a8.