Laborator 5 (SPER)

Modelare cu variabile mixte

15 martie 2022

Cuprins

compilat la: 15/03/2022, 00:14

Scopul lucrării

Recapitulăm notiunea de obstacol poliedral și adăugăm în descriere variabile binare pentru a caracteriza o condiție de excluziune de forma $x \notin P$.
Integrăm o astfel de condiție într-o problemă de generare a traiectoriei "din mers", ca rezultat al rezolvării repetate a unei probleme de optimizare cu constrângeri.

1 Noțiuni teoretice

Pentru a genera "on-the-fly" o traiectorie într-un mediu complex (cu obstacole fixe sau mobile) avem nevoie de:

- o modalitate eficientă de a descrie o regiune de interes (fie ea obstacol sau zonă de siguranță);
- un mecanism care să returneze comanda pentru sistem în așa fel încât să se îndeplinească cerințele misiunii (ocolire obstacole, minimizare traiectorie).

O descriere des întâlnită este acea de regiune poliedrală:

$$P = \{x \in \mathbb{R}^n : \ F_i^\top x \leq \theta_i, \forall i = 1 : N_h\}, \tag{1}$$

unde spațiul acoperit de obstacol este dat ca o intersecție de inegalități (amintiți-va de definițiile din Laboratorul 2). Definiția din (1) definește toate acele puncte ce respectă *incluziunea* $x \in P$ pe când noi suntem interesați de *excluziunea* $x \notin P$. Pentru a modela o astfel de relație, introducem variabile binare $z_i \in \{0, 1\}$, ceea ce ne permite să scriem o formulare "big-M":

$$x \notin P \iff \begin{cases} -F_i^\top x & \leq -\theta_i + M(1 - z_i), \ \forall i = 1 : N_h \\ \sum\limits_{i=1}^{N_h} z_i & = 1. \end{cases} \tag{2}$$

Relația (2) se interpretează astfel:

- o singură variabilă binară, cea de index i este egală cu '1', toate celelalte ($\forall j = 1 : N_h$ pentru $j \neq i$) sunt zero;
- prin urmare pentru toți indicii $j \neq i$ avem că inegalitățile ce implică x se reduc la $-F_j^\top x \leq -\theta_j + M \cdot 1$, ceea ce înseamnă că, pentru valori suficient de mari ale lui M, se poate ignora;
- rămânem doar cu inegalitatea $-F_i^\top x \leq -\theta_i + M \cdot 0$ ce ne asigură că suntem în exteriorul obstacolului P.

Ecuațiile din (2) combină variabile reale (x) și binare (z_i), așa că numim problema de optimizare în care sunt integrate *problemă de optimizare cu variabile mixte*.

O astfel de problemă, deși relativ ușor de formulat, este dificilă (NP-hard, fără garanții de optim global) deoarece numărul de cazuri crește exponențial (în exemplul din relația (2) sunt 2^{N_h} posibilități de a scrie inegalitățile).

O modalitate (descrisă în Laboratorul 3) de planifica mișcarea este să precalculăm o traiectorie pe care apoi să o urmărim în timpul simulării/experimentului. Cealaltă abordare uzuală este să calculăm "din mers" traiectoria, rezolvând la fiecare pas o problemă de optimizare.

O metodă foarte populară este "Model Predictiv Control – MPC" unde o formulare tipică este dată de problema de optimizare:

$$\min_{u_k \dots u_{k+N-1}} \sum_{i=1}^{N} (x_{k+i} - \bar{x})^\top (x_{k+i} - \bar{x}) + \sum_{i=0}^{N-1} u_{k+i}^\top u_{k+i} \tag{3a}$$

$$\text{s.t.} \quad x_{k+i+1} = Ax_{k+i} + Bu_{k+i}, \tag{3b}$$

$$|u_{k+i}| \leq \bar{u}, \tag{3c}$$

$$|x_{k+i+1}| \leq \bar{x}, \tag{3d}$$

$$x_{k+i+1} \notin P, \qquad \forall i = 1 : N. \tag{3e}$$

Fără a intra în detalii, o astfel de metodă este preferată datorită versatilității ei (putem schimba/adăuga după cum dorim elemente din cadrul (3)) și datorită faptului că ia în calcul în mod explicit constrângeri:

- se definesc costuri ce se doresc minimizate (efortul de-a lungul traiectoriei: $u_{k+i}^\top u_{k+i}$, distanța față de destinație $(x_{k+i} - \bar{x})^\top (x_{k+i} - \bar{x})$, etc.);
- se forțează respectarea constrângerilor (pe intrare: $|u_{k+i}| \leq \bar{u}$, pe stare: $|x_{k+i+1}| \leq \bar{x}$, de evitarea a obstacolelor: $x_{k+i+1} \notin P$);
- se aplică constrângerile și costul pe un orizont de predicție finit (de lungime N) iar din secvența de comenzi obținute $\{u_k, \dots, u_{n+N-1}\}$ se aplică doar prima, u_k;
- se repetă pași anteriori incrementând indexul $k \mapsto k + 1$.

Parametrul N merită câteva cuvinte adiționale. Tentația este bineînțeles să alegem un orizont de predicție cât mai lung pentru a "vedea" cât mai departe în viitor comportamentul sistemului. Dificultatea este că problema (3) trebuie rezolvată la fiecare pas k. La un moment dat (ce depinde de natura și dimensiunea problemei) timpul de calcul devine semnificativ (nu putem pierde secunde de calcul pentru o comandă ce trebuie trimisă la fiecare 50 sau 100ms, așa cum se întâmplă pentru o dronă).

În particular, pentru cazul în care adăugăm constrângeri pseudo-liniare precum cea din (2), problema devine semnificativ mai dificil de rezolvat ((3) este de fapt un MIQP-MPC). În general, cu cât un model este mai complex și ia în considerare restricții mai dificile, cu atât se ajunge la o problemă de optimizare mai dificilă.

Tool-urile Yalmip (disponibil doar în Matlab/Octave) sau CasADi (disponibil în Matlab/Octave/Python/C++) permit implementări eficiente ce, în ultimă instanță, apelează solvere specializate precum CPLEX, GUROBI, MOSEK sau IPOPT.

2 Ocolirea unui obstacol în plan prin programare cu variabile mixte și MPC

Ca prim pas, definim ecuațiile ce impun ocolirea unui pătrat:

$$\begin{cases} x_1 & \leq 5 + M(1 - z_1), \\ -x_1 & \leq 5 + M(1 - z_2), \\ x_2 & \leq 5 + M(1 - z_3), \\ -x_2 & \leq 5 + M(1 - z_4), \\ 1 & = z_1 + z_2 + z_3 + z_4. \end{cases} \tag{4}$$

Considerăm un model simplificat, “dublu integrator”, discretizat cu pasul 0.1,

$$\begin{cases} x_1^+ & = x_1 + 0.1 x_3, \\ x_2^+ & = x_2 + 0.1 x_4, \\ x_3^+ & = x_3 + 0.1 u_1, \\ x_4^+ & = x_4 + 0.1 u_2, \end{cases} \tag{5}$$

unde (x_1, x_2) definesc poziția, (x_3, x_4) viteza iar (u_1, u_2) accelerațiile sistemului. De asemenea, impunem constrângeri pe accelerații

$$-1 \leq u_1 \leq 1, \qquad -1 \leq u_2 \leq 1. \tag{6}$$

Integrând aceste elemente ca în (3) conduce la o problemă de optimizare cu constrângeri precum în fragmentul de cod următor:

```
objective = 0
solver.subject_to(X[:, 0] == X_init)

for k in range(controller['Npred']):

    # @ does matrix multiplication in Casadi
    solver.subject_to(X[0, k + 1] == (X[0, k] + plant['Te']*X[2,k]))
    solver.subject_to(X[1, k + 1] == (X[1, k] + plant['Te']*X[3,k]))
    solver.subject_to(X[2, k + 1] == (X[2, k] + plant['Te']*U[0,k]))
    solver.subject_to(X[3, k + 1] == (X[3, k] + plant['Te']*U[1,k]))

    solver.subject_to(plant['umin'] <= U[:, k])
    solver.subject_to(U[:, k] <= plant['umax'])

    objective = objective + \
                casadi.transpose(X[:, k] - X_target) @ (X[:, k] - X_target) + \
                casadi.transpose(U[:, k]) @ (U[:, k])

solver.minimize(objective)
```

Odată obținut obiectul ‘solver’, acesta este apelat în mod repetat, schimbând doar valorile inițiale, într-o buclă de simulare:

```
solver.set_value(X_target, [2,2,0,0])

for i in range(controller['Nsim']):
    solver.set_value(X_init, x0)

    sol = solver.solve()
    u0 = sol.value(U[:, 0])

    x0[0]=x0[0]+plant['Te']*x0[2]
    x0[1]=x0[1]+plant['Te']*x0[3]
    x0[2]=x0[2]+plant['Te']*u0[0]
    x0[3]=x0[3]+plant['Te']*u0[1]
```

3 Exerciții propuse

Exerciţiul 1. Codul asociat acestui laborator implementează doar cazul cu un singur obstacol. Cum ați proceda pentru a generaliza pentru cazul cu mai multe obtacole?

Exerciţiul 2. Am considerat un model extrem de simplu (și prin urmare nerealist). Modificați codul pentru a folosi modelul mașinii Dubins:

i) Testați traiectoriile obținute în cazul fără obstacole;

ii) Verificați comportamentul variind restricțiile pe comandă și lungimea orizontului de predicție. Ilustrați timpul de calcul.