Optimizacija parametrov regulatorja z algoritmom na osnovi obnašanja netopirjev

Dušan Fister[1]

[1] *Fakulteta za strojništvo, Smetanova ulica 17, 2000 Maribor*
E-pošta: dusan.fister@student.um.si

Abstract

Nowadays, nature inspired algorithms offer new approaches how to solve hard optimization problems. Bat Algorithm (BA) is one of the newest algorithms from this family. In this article, we use BA to optimize the SCARA robot positional regulator. Simulations and laboratory tests had shown the efficiency of the used algorithm.

1 Uvod

Natančnost pozicioniranja robota zaradi dinamičnih lastnosti okolja predstavlja v praksi zahtevnejše načrtovanje regulatorjev. V pričujočem povzetku smo se osredotočili na optimizacijo parametrov položajnega regulatorja dvoosnega mehanizma, imenovanega SCARA (angl. Selective Compliance Assembly Robotic Arm). V nasprotju z običajno metodo načrtovanja parametrov nelinearnega sistema, tukaj opisujemo načrtovanje parametrov s pomočjo algoritmov na osnovi obnašanja netopirjev.

Struktura članka je naslednja: v drugem poglavju opišemo problem, v tretjem pa podrobneje predstavimo možno rešitev problema ter algoritem na osnovi obnašanja netopirjev. V četrtem poglavju opišemo potek simulacij. V zadnjem poglavju povzamemo opravljeno delo in začrtamo smernice za nadaljni razvoj.

2 Opis problema

SCARA je robotski mehanizem, namenjen za natančnejša opravila v proizvodni liniji. Njegovo gibanje v splošnem določajo štiri prostostne stopnje, zato spominja na zgradbo človeške roke. Uporabljen SCARA robot je preprostejše izvedbe in sestoji iz dveh prostnostnih stopenj, ki zajemata rotacijo okrog "rame"in "komolca" [2]. Poganjata ga dva DC motorja z vgrajenima inkrementalnima dajalnikoma. Vodenje motorjev po navoru in hitrosti ter branje relativnega položaja v realnem času opravlja DSP-2-Roby vhodno/izhodna kartica. Mehanski sistem SCARA robota lahko s pomočjo nelinearne diferencialne enačbe modeliramo ter na modelu opravljamo poljubna testiranja. Naša naloga je optimizacija parametrov položajnega regulatorja, ki so predstavljeni v nadaljevanju. Parametre optimiramo glede na zahtevano vrednost prenihaja, časa vzpona ali statičnega pogreška. Podobna rešitev je bila implementirana že v [3].

3 Rešitev problema

Algoritem na osnovi obnašanja netopirjev (angl. Bat Algorithm, krajše BA) je razvil leta 2010 kitajski matematik Xin-She Yang [4, 5, 6, 8]. Njegov algoritem spada v področje inteligence rojev [7], ki se dandanes v splošnem uporablja za namene optimizacije. Glavni vzor BA algoritma predstavlja netopirjeva sposobnost določitve razdalje na podlagi ultrazvočnega merjenja odmeva. Avtor je v algoritmu želel opisati njihovo obnašanje z matematično enačbo ter v nadaljevanju razvil optimizacijski algoritem za širšo uporabo. Psevdo-koda algoritma je prikazana v Algoritmu 1.

Algorithm 1 Algoritem na osnovi obnašanja netopirjev

Vpis: Populacija netopirjev $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ za $i = 1 \ldots Np$, MAX_FE.
Izpis: Najboljša rešitev $\mathbf{x}_{best}$ in njena vrednost $f_{min} = \min(f(\mathbf{x}))$.

```
1: init_bat();
2: eval = vrednoti_novo_populacijo;
3: f_min = išči_najboljšo_rešitev(x_best);
4: while termination_condition_not_meet do
5:    for i = 1 to Np do
6:       y = generiraj_novo_rešitev(x_i);
7:       if rand(0,1) > r_i then
8:          y = izboljšaj_najboljšo_rešitev(x_best)
9:       end if{ lokalno iskanje }
10:      f_new = vrednoti_novo_rešitev(y);
11:      eval = eval + 1;
12:      if f_new ≤ f_i and N(0,1) < A_i then
13:         x_i = y; f_i = f_new;
14:      end if{ shrani najboljšo rešitev po pogojem }
15:      f_min=išči_najboljšo_rešitev(x_best);
16:   end for
17: end while
```

BA je populacijski algoritem, kjer vsak delec (netopir) populacije predstavlja možno rešitev problema. Rešitev problema optimizacije položajnega regulatorja v splošnem predstavimo kot vektor

$$\mathbf{x}_i = \{Kp_1, Kv_1, \ldots, Kp_n, Kv_n\} \quad (1)$$

kjer parametra $\langle Kp_i, Kv_i \rangle$ predstavljata položaj regulatorja i-te osi in n določa število osi. Vsaka rešitev določa pozicijo delca v prostoru preiskovanja. To pozicijo ocenimo z ocenitveno funkcijo. Boljša, kot je vrednost ocenitvene funkcije, boljša (kakovostnejša) je rešitev. V našem

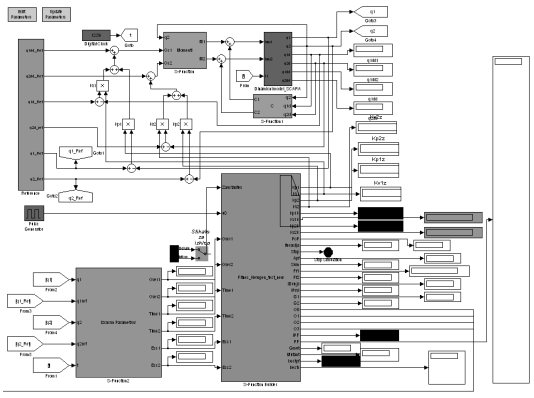
primeru ocenitveno funkcijo, ki jo minimiziramo , izrazimo z enačbo:

$$f(\mathbf{x}_i) = \frac{1}{n}\sum_{i=1}^{n} w_{i1}(1 - |P_i - Over_i|) + \\ w_{i2}(1 - Time_i) + w_{i3}(1 - Ess_i) \quad (2)$$

kjer P_i predstavlja zahtevan prenihaj, $Over_i$ dejanski prenihaj, $Time_i$ čas zakasnitve, in Ess_i statični pogrešek i-te osi. Uteži w_{ij} določajo pomembnost posameznega člena v enačbi 2. Pri tem velja, da je vsota uteži pri regulaciji posamezne osi enaka , t.j. $\sum_{j=1}^{3} w_{ij}$ za $i = 1 \ldots n$. Parametre, ki nastopajo v enačbi (2), delimo na vhodne (npr. P_i) in izhodne (npr. $Over_i$, $Time_i$, Ess_i). Vhodne parametre določa uporabnik. Izhodni podatki predstavljajo dejanske vrednosti, ki jih pridobimo morebiti s simulacijskim izračunom ali morebiti z realno meritvijo. V praksi vrednosti uteži običajno uravnotežimo, t.j. $w_{i1} = w_{i2} = w_{i3} = \frac{1}{3}$.

4 Simulacije

V simulacijah uporabimo položajni regulator z dvema osema ($n = 2$). Pri tem operiramo z dvema paroma parametrov $\langle Kp_1, Kv_1 \rangle$ in $\langle Kp_2, Kv_2 \rangle$. Rezultati optimizacije z algoritmom BA predstavljajo vrednosti parametrov, ki jih vpišemo neposredno v regulator. Odziv regulatorja testiramo s stopnično funkcijo. Simulacijski model je izdelan v MATLAB/Simulink okolju, primernem za matematično računanje, modeliranje, simuliranje ter analiziranje dinamičnih sistemov [1]. Uporabnik sestavlja simulacijski model z zlaganjem *S-funkcij* blokov. Ob vsakem izračunu ocenitvene funkcije se na izhod postavijo vrednosti koeficientov. Izhod predstavlja položajni regulator, ki niz podatkov posreduje do bloka dinamični model SCARA. Ta sodeluje z blokom reference za izračun trenutnega položaja, vztrajnostnega momenta in Coriollisovih ter centrifugalnih sil. Blok za izračun parametrov izračuna dejanske vrednosti prenihajev, vzponske čase in statične pogreške, ki jih v naslednjem koraku BA algoritem selekcionira. Zadnji korak predstavlja izpis elitne četverice parametrov, ki je pripravljena za neposredni vpis v regulator.



Slika 1: Posnetek MATLAB/Simulink modela.

5 Zaključek

Rezultati simulacij so pokazali, da je bila predlagana rešitev uspešno realizirana. Nastavljanje parametrov regulatorja poteka enostavno, vendar je iskanje njihovih optimalnih vrednosti s pomočjo genetskih algoritmov oz. nevronskih mrež časovno zahtevno in neprimerno za delo v realnem času. Začetna testiranja z algoritmom BA so pokazala, da je ta veliko manj časovno zahteven, njegovi rezultati pa so dovolj natančni za uporabo v praksi. V prihodnje želimo omenjeni algoritem uporabiti tudi na realnem sistemu.

Zahvala

Zahvaljujem se prof. Riku Šafariču za pomoč pri testiranjih in izgradnji simulacijskega modela. Prav tako se zahvaljujem Iztoku Fistru Jr. za pomoč pri izdelavi optimizacijskega algoritma.

Literatura

[1] T. Slanič: Genetski regulator za dvoosnega robota SCARO, Diplomsko delo, September 2006, Maribor

[2] J. Čas: Izdelava zveznega nevronskega Sliding-Mode regulatorja za teleoperiranje SCARA robota, Diplomsko delo, September 2006, Maribor

[3] R. Šafarič, K. Jezernik, M. Rodič, A. Sabanovic, S. Uran: Sliding-mode neural network robot controller, Advanced Motion Control, 18. - 21. Marec 1996, 4th International Workshop

[4] X.S. Yang, A.H. Gandomi: Bat algorithm: a novel approach for global engineering optimization, 2012, Engineering Computations, 29.5, 464-483.

[5] I. Fister Jr., D. Fister, X.S. Yang: A Hybrid Bat Algorithm, 2013, Elektrotehniški vestnik 80.1-2: 1-7

[6] X.S. Yang: A new metaheuristic bat-inspired algorithm, Nature inspired cooperative strategies for optimization, Springer Berlin Heidelberg, 2010, NICSO 65-74

[7] I. Fister Jr., D. Fister, I. Fister: A comprehensive review of cuckoo search: variants and hybrids, 2013, International Journal of Mathematical Modelling and Numerical Optimisation 4.4, 387-409

[8] I. Fister Jr, D. Fister, I. Fister: Differential evolution strategies with random forest regression in the bat algorithm, ACM 2013, Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion