Primerjava simpleksnega algoritma in simuliranega ohlajanja pri optimizaciji funkcij z zveznimi spremenljivkami

Jernej Olenšek, Janez Puhan, Árpád Bűrmen, Iztok Fajfar in Tadej Tuma
Fakulteta za elektrotehniko
Univerza v Ljubljani
Tržaška 25, 1001 Ljubljana, Slovenia
jernej.olensek@fe.uni-lj.si

Abstract

This paper presents a method for optimization of functions of continuous variables. The method is based on the simulated annealing algorithm and uses orthogonal experimental design to choose a good candidate point for the next iteration. This method is compared with a variant of simplex algorithm which was already proven to be quite successful for optimization of integrated electronic circuits.

1 Uvod

Optimizacijski problemi se pojavljajo na skoraj vseh področjih znanosti, tehnike in tudi v poslovnem svetu. Ker so ti problemi analitično rešljivi le za majhno skupino najenostavnejših primerov, se vedno bolj uveljavljajo različni algoritmi za numerično optimizacijo dane kriterijske funkcije. Namen optimizacije je v čim krajšem času poiskati globalen ekstrem kriterijske funkcije na danem območju. V praksi imajo funkcije običajno precej obsežno definicijsko območje, še večjo težavo pa predstavlja veliko število lokalnih ekstremov, ki so prav tako lastnost večine praktičnih problemov. Z večanjem števila parametrov se obe lastnosti zelo hitro stopnjujeta, kar predstavlja velik izziv za vsak optimizacijski algoritem. Glavna težava numerične optimizacije je zagotovo čas, ki ga algoritem potrebuje, da najde dobro rešitev. To je še posebej pomembno, ko je izračun kriterijske funkcije odvisen od rezultatov simulacije. Ta običajno traja precej časa, poleg tega pa vnaša v kriterijsko funkcijo še numeričen šum, ki dodatno oteži reševanje že v osnovi težkih problemov. Takšne težave se na primer pojavljajo pri optimizaciji elektronskih vezij, kjer je potrebno za vsak izračun kriterijske funkcije izvesti simulacijo vezja. V tem primeru gradientne metode ne pridejo v poštev, saj bi jih numerični šum takoj ustavil. Na voljo je več različnih vrst algoritmov, ki so namenjeni prav reševanju takih problemov. Med njimi so na primer simulirano ohlajanje, genetski algoritmi, iskanje s tabuji ali pa direktne metode kot je optimizacija s simpleksi. V tem članku primerjamo ortogonalno simulirano ohlajanje [1] in Box-ovo različico simpleksnega algoritma [2], ki je že implementirana v simulacijskem orodju za elektronska vezja SPICE OPUS in je sposobna hitro najti zmerno dobro rešitev optimizacijskega problema.

2 Simulirano ohlajanje

Simulirano ohlajanje je metoda, ki izhaja iz procesa ohlajanja kovin. Cilj procesa je razporediti atome v čimbolj urejeno strukturo, ki bo imela najnižjo energijo in s tem največjo trdnost. Zato je potrebno kovino najprej segreti na visoko začetno temperaturo, nato pa sistem dovolj počasi ohlajati, da se atomi lahko uredijo v strukturo z najnižjo energijo. Pri visoki temperaturi imajo atomi dovolj energije, da lahko brez težav prehajajo tudi v stanja z višjo energijo, medtem ko pri nižji temperaturi vedno bolj težijo v nižja energijska stanja. Če je bilo ohlajanje izvedeno dovolj počasi, se pri temperaturi nič atomi umirijo v stanju z najnižjo energijo in urejeno strukturo, drugače pa se lahko v kovini pojavijo napake, ki ji zmanjšujejo kakovost.

Postopek simuliranega ohlajanja ta proces posnema z namenom doseči čim nižjo vrednost kriterijske funkcije, ki je analogna energiji sistema. Prehodi med stanji so podobno kot prehodi med energijskimi stanji v kovini odvisni od vrednosti kriterijske funkcije in od posebnega parametra, ki igra vlogo temperature. Glavna lastnost algoritma je, da z določeno verjetnostjo dopušča tudi prehode v stanja z višjo vrednostjo kriterijske funkcije. To pomeni, da ima algoritem, tudi če zaide v lokalni ekstrem, še vedno možnost, da ga zapusti. Verjetnost prehodov v slabša stanja je višja pri visoki temperaturi nato pa z nižanjem temperature pada proti nič. V limiti, ko gre temperatura proti nič, dopušča algoritem le še prehode v stanja z nižjo vrednostjo kriterijske funkcije. Osnovni koraki postopka simuliranega ohlajanja so naslednji:

1. Inicializacija
2. Generacija nove točke
3. Premik
4. Če je konec temperaturne stopnje, pojdi na 5, drugače pojdi na 2.
5. Ohlajanje
6. Če še ni izpolnjen pogoj za ustavitev, pojdi na 2.

Med inicializacijo je potrebno nastaviti parametre algoritma, kot so meje definicijskega območja, začetna točka, začetna temperatura, število korakov v vsaki temperaturni stopnji, urnik hlajenja in ustavitveni kriterij.

Med postopkom generacije nove točke na osnovi izbrane verjetnostne porazdelitve iz trenutne točke določimo možno točko za naslednjo iteracijo.

Kriterij za premik v generirano točko je največkrat Metropolisov kriterij, ki za naslednjo iteracijo vedno sprejme točke z nižjo vrednostjo kriterijske funkcije, točke z višjo vrednostjo pa sprejme z verjetnostjo

$$P = exp(-(y' - y)/T) \tag{1}$$

kjer je y' vrednost kriterijske funkcije v novi točki, y vrednost v trenutni točki in T trenutna vrednost temperature. V primeru, ko je P v enačbi (1) večji od naključno generirane vrednosti iz intervala $(0, 1)$, se algoritem premakne v predlagano točko, sicer pa ostane v trenutni. Če je nova točka najboljša doslej, jo tudi shranimo.

Korak 5. je poleg generacije nove točke najpomembnejši del algoritma, saj sta od hitrosti ohlajanja odvisna tako kakovost rešitve, kot tudi čas izvajanja algoritma. V primeru hitrega ohlajanja lahko algoritem hitro najde rešitev, vendar je to pogosto le lokalni ekstrem. Če pa je ohlajanje počasno, algoritem običajno najde boljšo rešitev, vendar za to porabi veliko časa.

Glavna težava vseh globalnih optimizacijskih metod je med množico številnih lokalnih ekstremov poiskati globalnega. Simulirano ohlajanje je eden od algoritmov, ki v določenih izvedbah lahko zagotovijo konvergenco v verjetnosti h globalnemu ekstremu. To pomeni, da gre verjetnost, da bo algoritem prehajal v stanja, ki niso globalni ekstrem, proti nič, ko gre čas izvajanja proti neskončnosti [4]. Vendar pa vse izvedbe algoritma v praksi niso vedno uspešne, saj je potrebno za kovergenco temperaturo nižati počasi in je čas, v katerem algoritem najde zadovoljivo rešitev, nesprejemljivo dolg. Zato je v večini praktičnih izvedb algoritem na tak ali drugačen način prilagojen. Tako sicer dokazi o konvergenci izgubijo veljavo, vendar pa je možno še vedno dobiti dobre rezultate in to v sprejemljivem času.

Najenostavnejša in tudi najpogostejša prilagoditev je hitrejše ohlajanje. S tem sicer poskrbimo, da algoritem hitreje najde rešitev, vendar pa tudi tvegamo, da se bo ujel v lokalni ekstrem, ki ga nato zaradi hitrega ohlajanja morda ne bo mogel več zapustiti. Zato hitrejše ohlajanje samo po sebi običajno ni edina potrebna sprememba. Poleg urnika hlajenja je potrebno prilagoditi tudi način generiranja novih točk. Eden od algoritmov, ki so nastali na ta način, je ortogonalno simulirano ohlajanje (OSA) [1].

2.1 Generiranje nove točke

Način generiranja novih točk je pomemben del algoritma. Izbran mora biti tako, da omogoča učinkovito in hkrati dovolj temeljito iskanje po danem prostoru. Ta algoritem uporablja za generiranje sprememb posameznih parametrov Cauchy-jevo porazdelitev [5]. Gostota verjetnosti za to porazdelitev v eni dimenziji je:

$$p(x) = \frac{T_g}{\pi \cdot (x^2 + T_g^2)} \tag{2}$$

pri čemer pada s T_g tudi verjetnost dolgih skokov. S tem je poskrbljeno, da algoritem pri nizkih vrednostih parametra T_g več časa porabi za iskanje v bližnji okolici trenutne točke, še vedno pa obstaja možnost da zapusti morebitni lokalni ekstrem.

Za razliko od [1] tu Cauchy-jeva porazdelitev ni odvisna od trenutne temperature. Pri funkcijah, ki se ne spreminjajo zelo veliko, lahko nastavimo začetno temperaturo na nižjo vrednost, da ne izgubljamo veliko časa pri visokih temperaturah. S tem bi omejili tudi možnost daljših skokov že na začetku optimizacije. Zato uporabimo za generiranje premikov ločen parameter T_g, ki je odvisen od definicijskega območja. Začetno vrednost parametra postavimo na širino intervala tistega optimizacijskega parametra, ki ima največje območje, nato pa T_g nižamo po enakem urniku kot temperaturo. Urnik hlajenja je opisan v naslednjem razdelku.

Za generacijo točke za naslednjo iteracijo uporablja algoritem ortogonalne eksperimente [1], [3]. Ta statistični postopek omogoča ugotavljanje vpliva posameznih faktorjev na nek odziv. V primeru optimizacije je lahko vsak faktor en sam ali pa tudi skupina večih optimizacijskih parametrov, ki jih nato spreminjamo skupaj kot celoto. Odziv, ki ga opazujemo, je kriterijska funkcija.

Vse optimizacijske parametre najprej rezdelimo v več manjših skupin in vsako obravnavamo kot en faktor. Število faktorjev določimo z:

$$N_f = (3^{\lfloor log_3(2 \cdot N_v + 1) \rfloor} - 1)/2 \tag{3}$$

kjer je N_v število optimizacijskih parametrov. Število faktorjev N_f lahko načeloma postavimo poljubno v intervalu od 1 do N_v. Večja vrednost pomeni boljšo učinkovitost algoritma in manj iteracij, vendar le če so medsebojni vplivi med faktorji majhni. Manjša vrednost pomeni boljšo oceno vpliva posameznega faktorja, vendar potrebuje algoritem več iteracij da najde dobro rešitev. Enačba (3) je kompromis, ki se je izkazal za uspešnega. Za uvrščanje parametrov v faktorje naključno generiramo $N_f - 1$ izmed $N_v - 1$ možnih mest, ki ločijo posamezne parametre med seboj in tako združimo parametre v faktorje.

Za vsako komponento vektorja parametrov po Cauchy-jevi verjetnostni porazdelitvi generiramo premik r in nato iz trenutne točke x in dobljenega vektorja premika določimo dva nova vektorja:

$$\begin{aligned} x_1 &= x + r \\ x_2 &= x - r \end{aligned} \tag{4}$$

V primeru, ko pade x_1 oziroma x_2 izven podanega območja, generiramo vrednost za tisti parameter po enakomerni porazdelitvi v celotnem območju tega parametra.

Nato izvedemo vrsto eksperimentov, v katerih posamezne parametre postavimo na vrednosti x (prvi nivo), x_1 (drugi nivo) oziroma x_2 (tretji nivo), glede na to, v kateri faktor parameter sodi in kateri nivo predpisuje ortogonalno polje za ta faktor v trenutnem eksperimentu. Na ta način izmed vseh nožnih kombinacij nivojev faktorjev izberemo majhen vendar reprezentativen vzorec točk v prostoru okoli trenutne točke. Ortogonalno polje zagotavlja, da so vsi nivoji vseh faktorjev enakomerno zastopani, kar omogoča neodvisno oceno njihovega vpliva na vrednost kriterijske funkcije. Celoten algoritem za generiranje ortogonalnih polj za določeno število faktorjev in nivojev je na voljo v [3]. Za vsak eksperiment izračunamo vrednost kriterijske funkcije in nato ocenimo vplive posameznih nivojev vseh faktorjev. Ocena za vpliv k-tega nivoja faktorja j je:

$$S_{j,k} = \sum_t y_t \cdot F_t \tag{5}$$

kjer je y_t vrednost kriterijske funkcije v eksperimentu t, F_t pa ima vrednost 1, če je nivo faktorja j v eksperimentu t enak k, sicer pa ima F_t vrednost 0. Vsota teče preko vseh eksperimentov.

Za najboljši nivo faktorja j izberemo tistega, pri katerem je $S_{j,k}$ minimalen. S kombinacijo teh nivojev faktorjev nato generiramo novo točko, za katero pričakujemo, da bo boljša od vseh eksperimentov. Na ta način smo na osnovi majhnega vzorca izbrali smer upadanja funkcije, vendar pri tem nismo računali gradienta. V primeru, ko tako generirana točka ni boljša od vseh eksperimentov, predlagamo za novo točko kar najboljši eksperiment, v katerem so parameteri različni od trenutne točke algoritma.

2.2 Urnik hlajenja

Algoritem uporablja zelo pogost geometrični urnik hlajenja, pri katerem je temperatura v vsaki temperaturni stopnji podana kot:

$$T(k) = T_0 \cdot \alpha^k;\ k = 0, 1, 2, ... \tag{6}$$

kjer je T_0 začetna temperatura, α konstanta iz intervala $(0, 1)$ in k zaporedna številka temperaturne stopnje. Izbira T_0 vpliva na verjetnost prehoda v slabše točke v začetku optimizacije in mora biti postavljena dovolj visoko, da algoritem sprejme večino točk. V našem primeru postavimo T_0 na velikost razpršenosti vrednosti kriterijske funkcije v 10 naključnih začetnih točkah. Konstanta α pa vpliva na hitrost nižanja temperature in je v našem primeru empirično postavljena na 0.985.

Za urnik hlajenja je pomemben še en parameter, to je število korakov v posamezni temperaturni stopnji. V našem primeru je ta parameter postavljen na začetno vrednost $N_0 = 10$ in se nato manjša skupaj s temperaturo:

$$NT(k) = N_0 \cdot \alpha^k;\ k = 0, 1, 2, ... \tag{7}$$

S tako izbiro vsaj na začetku izvajanja algoritma poskrbimo, da algoritem dovolj časa teče pri višji temperaturi, s čimer poskrbimo, da že v zgodnji fazi ne zaide v lokalni minimum.

Algoritem ustavimo, ko doseže število izračunov kriterijske funkcije 50000 pri $N_v = 30$ oziroma 70000 pri $N_v = 100$.

3 Razultati

Ortogonalno simulirano ohlajanje smo primerjali z Box-ovo različico simpleksnega algoritma [2], ki se je v praksi že izkazala za precej uspešno pri optimizaciji analognih integriranih elektronskih vezij s programom SPICE OPUS. Primerjava s tem algoritmom bo pokazala, ali ima simulirano ohlajanje boljšo sposobnost iskanja globalnega ekstrema. Oba algoritma sta imela na voljo omejeno največje število izračunov kriterijske funkcije.

Funkcije za primerjavo algoritmov so zbrane v [3]. Dimenzije problemov 1-6 in 11-15 so 30, za funkcije 7-10 pa 100. Problemi so zelo obsežni, nekateri pa imajo tudi zelo veliko število lokalnih ekstremov in so dober test za vsak algoritem (ker so koeficienti

funkcije f_8 naključni in v [3] niso podani, je tu ne upoštevamo). Vsako funkcijo smo optimizirali 50-krat, vsakič iz naključno izbrane začetne točcke. Rezultati so zbrani v tabeli 1.

	Box-ov simpleks	OSA	globalni minimum
f_1	-9596.2 -3012.3	-12569.5 -12569.5	-12569.5
f_2	6.9709 224.67	$3.5527 \cdot 10^{-14}$ 1.9899	0
f_3	$1.6714 \cdot 10^{-2}$ 3.93447	$1.5234 \cdot 10^{-7}$ $2.7337 \cdot 10^{-7}$	0
f_4	$2.2362 \cdot 10^{-5}$ 3.04840	$1.7375 \cdot 10^{-13}$ $1.6971 \cdot 10^{-1}$	0
f_5	$2.1584 \cdot 10^{-5}$ 1.59433	$1.6672 \cdot 10^{-16}$ $2.0732 \cdot 10^{-1}$	0
f_6	$1.9551 \cdot 10^{-5}$ 6.99148	$3.1601 \cdot 10^{-15}$ $1.0987 \cdot 10^{-2}$	0
f_7	-61.124 -29.361	-98.861 -97.860	-99.2784
f_9	-70.408 -63.311	-78.332 -78.331	-78.33236
f_{10}	113.13 196.95	283.59 795.04	0
f_{11}	$2.5981 \cdot 10^{-8}$ $5.3587 \cdot 10^{-5}$	$5.4955 \cdot 10^{-14}$ $8.2124 \cdot 10^{-13}$	0
f_{12}	$4.1618 \cdot 10^{-3}$ $2.0489 \cdot 10^{-2}$	$2.4360 \cdot 10^{-2}$ $1.5660 \cdot 10^{-1}$	0
f_{13}	$1.9705 \cdot 10^{-1}$ 3.6034	$1.8169 \cdot 10^{-7}$ $8.2303 \cdot 10^{-7}$	0
f_{14}	$8.9943 \cdot 10^{-2}$ 39.777	102.83 1928.0	0
f_{15}	1.8026 9.4910	$1.2809 \cdot 10^{-5}$ $8.4080 \cdot 10^{-1}$	0

Tabela 1: Prva vrednost je najboljša, druga pa najslabša vrednost funkcije v 50 optimizacijskih tekih

Rezultati optimizacije kažejo, da je simulirano ohlajanje obetaven algoritem. V primerjavi s simpleksnim algoritmom izkazuje boljšo sposobnost iskanja globalnega ekstrema v večini testiranih primerov. Simpleksni algoritem sicer najde svojo rešitev hitreje (pri funkcijah f_{13} in f_{15} je v povprečju končal še preden je porabil polovico dovoljenih izračunov funkcije), vendar je pri večini funkcij z več minimumi obtičal le v lokalnem ekstremu. Pri nekaterih funkcijah (predvsem pri f_{10} in f_{14}) pa je imel algoritem simuliranega ohlajanja zelo velike težave in ga je simpleksni algoritem prekašal za več velikostnih razredov. To so funkcije, ki imajo zelo neizrazit minimum in zelo močan šum. V teh primerih pride do izraza dejstvo, da dela simpleksni algoritem s populacijo točk, medtem ko se pri simuliranem ohlajanju v naslednjo iteracijo prenaša ena sama točka. Pokaže se tudi zelo počasna konvergenca simuiranega ohlajanja proti koncu optimizacije. Vendar pa je treba poudariti, da pri ortogonalnem simuliranem ohlajanju ni bilo nikakršne uglasitve parametrov na posamezen problem. Vse nastavitve algoritma so bile za vse funkcije take, kot so opisane zgoraj. Uglasitev algoritma na posamezen problem lahko občutno izboljša hitrost delovanja in kakovost rešitve, vendar pa za uglasitev žal ni nobenega splošnega pravila.

4 Zaključek

V članku je predstavljena verzija algoritma simuliranega ohlajanja, ki jo primerjamo z Box-ovim simpleksnim algoritmom. Za primerjavo je uporabljena skupina testnih matematičnih funkcij, med katerimi so funkcije z enim samim ekstremom in tudi težji problemi z velikim številom lokalnih ekstremov. Primerjava z Box-ovim algoritmom je pokazala, da simulirano ohlajanje za večino testiranih funkcij najde boljšo rešitev. Uporaba ortogonalnih eksperimentov pri simuliranem ohlajanju občutno izboljša delovanje osnovnega algoritma, saj zmanjša verjetnost generiranja točk, ki bi jih Metropolisov kriterij kasneje zavrnil, zaradi česar lahko temperaturo tudi hitreje nižamo. Vendar pa algoritem kaže tudi na pomankljivost vseh strategij, ki pohitrijo optimizacijo. Hitrejše delovanje algoritma dosežemo vedno na račun kakovosti rešitve, saj s prehitrim ohlajanjem vedno tvegamo, da bo algoritem obtičal v lokalnem ekstremu.

Pri nadaljnjih raziskavah bi se izplačalo razširiti simulirano ohlajanje na populacijo točk.

5 Zahvala

Raziskave je sofinancirala Agencija za raziskovalno dejavnost v okviru programa P2-0246 - Algoritmi in optimizacijski postopki v telekomunikacijah.

Literatura

[1] Li-Sun Shu, Shinn-Ying Ho. A Novel Orthogonal Simulated Annealing Algorithm for Optimization of Electromagnetic Problems. *IEEE transactions on magnetics*, vol.40, No.4, pp. 1790-1795, July 2004.

[2] M.J. Box. A new method of constrained optimization and a comparison with other methods. *Computer Journal*, vol.8, pp. 42-52, 1965.

[3] Yiu-Wing Leung. An orthogonal Genetic Algorithm with Quantization for Global Numerical Optimizaion. *IEEE transactions on evolutionary computation*, vol.5, No.1, pp. 41-53, 2001.

[4] R.L. Yang Convergence of simulated annealing algorithm for continuous global optimization. *Journal of optimization theory and applications*, vol.104, No.3, pp. 691-716, 2000.

[5] Harold Szu, Ralph Hartley. Fast simulated annealing. *Physics Letters A*, vol.122, No.3,4, pp. 157-162, 1987.