SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4834

Algoritmi za skupu optimizaciju

Lucija Ulaga

Zagreb, lipanj 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 5. ožujka 2017.

ZAVRŠNI ZADATAK br. 4834

Pristupnik: **Lucija Ulaga (0036483630)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Algoritmi za skupu optimizaciju**

Opis zadatka:

Opisati algoritme za skupu optimizaciju koji se koriste na problemima čija evaluacija traje netrivijalno dugo. Istražiti i implementirati algoritme za skupu optimizaciju u programski sustav ECF. Usredotočiti se na Powellove algoritme optimizacije i njihove inačice. Ispitivanje i usporedbu provesti nad postojećim problemima već prilagođenim postojećem programskom sustavu. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 10. ožujka 2017.
Rok za predaju rada: 9. lipnja 2017.

Mentor:
Izv. prof. dr. sc. Domagoj Jakobović

Djelovođa:
Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Siniša Srbljić

SADRŽAJ

1. Uvod

Razvoj računala i računarstva omogućili su ljudima rješavanje do tada nerješivih, kompleksnih problema. Glavno svojstvo takvih problema je da je potrebno napraviti puno evaluacija što ručno nema smisla raditi stoga današnja računala, koja mogu u jednoj sekundi pretražiti milijune potencijalnih rješenja kako bi pronašli ono najbolje, stvaraju priliku za proučavanje i rješavanje velikog skupa teških problema.

Prije nego što su spomenuti različiti pristupi i algoritmi, potrebno je uvesti pojam optimizacije. Optimizacija se pojavljuje u matematici i računarskoj znanosti u kontekstu rješavanja kompleksnih problema. Optimizacija znači pronalaženje najboljeg elementa po nekom kriteriju iz niza dostupnih alternativa. Najčešće to znači pronalaženje minimuma ili maksimuma realne funkcije tako da se sistematski biraju ulazni podatci odnosno ulazne vrijednosti i računa izlazna vrijednost funkcije. Generalno, optimizacija je pronalaženje najbolje dostupne vrijednosti funkcije nad definiranom domenom.

Optimizacijski problem može se najjednostavnije predstaviti na sljedeći način:

Zadana je funkcija $f : A \rightarrow R$ iz skupa A u skup realnih brojeva. U slučaju minimizacije, potrebno je pronaći element x_0 iz skupa A tako da vrijedi:

$$f(x_0) \leq f(x) \text{ za svaki x iz A.}$$

U slučaju maksimizacije, potrebno je pronaći element x_0 iz skupa A tako da vrijedi:

$$f(x_0) \geq f(x) \text{ za svaki x iz A.}$$

Skup A je podskup Euklidskog prostora, određen nizom ograničenja koja elementi skupa moraju zadovoljavati. Domena skupa A naziva se prostor pretraživanja, dok su elementi skupa moguća rješenja. Funkcija f ima više naziva, funkcija cilja, funkcija cijene itd. Bitna prepreka na koju algoritmi optimizacije nailaze je razlikovanje lokalnog ekstrema, na primjer lokalnog minimuma od globalnog minimuma. No nešto više o tome kasnije.

Ovisno o prirodi problema i njegovim karakteristikama, osmišljeni su različiti pristupi, metode i algoritmi. Neki od načina za rješavanje teških problema podrazumijevaju korištenje algoritama koji završavaju nakon konačnog broj koraka, zatim iterativne metode koje se približavaju rješenju ili heuristike koje daju aproksimirana rješenja.

Većina kompleksnih problema svodi se zapravo na pronalaženje optimalnog rješenja, što može biti već spomenuti pronalazak minimuma ili maksimuma funkcije, najkraći ili najbrži put do ciljnog stanja, pronalaženje funkcije ili modela koji najbolje opisuju neko ponašanje i mnogo drugih.

Ukratko, optimizacijski algoritmi i metode dijele se na nekoliko kategorija:

- Kombinatorička optimizacija – skup mogućih rješenja je diskretan ili ga je moguće reducirati na diskretan skup, cilj je pronaći najbolje rješenje (na primjer problem trgovačkog putnika)
- Dinamičko programiranje – raščlanjivanje problema na skup manjih i jednostavnijih pod problema
- Evolucijski algoritmi – optimizacijski algoritam temeljen na populaciji, podskup evolucijskog računarstva
- Gradijentne metode – koriste gradijent funkcije za usmjeravanje pretrage
- Metaheuristike – skup koncepata za definiranje heurističkih metoda
- Stohastička optimizacija – korištenje slučajno generiranih varijabli

Od ovih 6 kategorija, svaka sadrži još mnogo pod kategorija. Detalji svih ovih metoda i algoritama nisu ovdje u fokusu, već one služe za dočaravanje brojnosti različitih pristupa koji su do danas osmišljeni. Netko se može pitati koji je optimizacijski algoritam najbolji? No to pitanje nije potpuno, trebamo se pitati koji je algoritam najbolji za naš problem. Kada dobro definiramo problem, njegova svojstva i karakteristike, tek tada trebamo krenuti u potragu za primjenjivom metodom i algoritmom.

Tako na primjer, za jednostavnije probleme, rješenje se može pronaći pretraživanjem cijelog prostora stanja algoritmima slijepog pretraživanja ili nešto naprednijim algoritmima usmjerenog pretraživanja koji koriste dodatne informacije kako bi usmjerili pretraživanje. Takvi problemi koji se mogu riješiti iscrpnom pretragom su samo jedna kategorija problema.

Jedna od značajnijih grana u računarskoj znanosti je evolucijsko računarstvo. Ono obuhvaća familiju algoritama za globalnu optimizaciju koji su inspirirani biološkom evolucijom. Drugim riječima, taj podskup algoritama i tehnika za rješavanje teških problema nastao je proučavanjem procesa u prirodi. U evolucijskom računarstvu, generira se inicijalni set kandidata za rješenje i iterativno se ažurira. Ovdje se javljaju pojmovi također inspirirani evolucijom odnosno preuzeti iz biologije, poput generacije, populacije, selekcije, mutacije i sl. Tako svaka nova generacija potencijalnih rješenja nastaje odabirom (selekcijom) boljih odnosno odbacivanjem lošijih rješenja uz uvođenje sitnih promjena nad jedinkama (mutacijom, križanjem), odnosno pojedinim rješenjima. Rezultat ovog procesa je evolucija populacije.

Generalno, evolucijski algoritam predstavlja podvrgavanje populacije optimizaciji sa ciljem evoluiranja ka boljim rješenjima. Svako rješenje ima određena svojstva odnosno genotip koji je moguće mijenjati odnosno mutirati. Funkcija fitnesa radi evaluaciju odnosno vraća informaciju o dobroti rješenja što se onda koristi za selekciju jedinki iz populacije za sljedeću generaciju.

Ovakav kratak opis obilježava generalan pristup evolucijskog računarstva rješavanju problema, ali nam i daje uvid u moguća ograničenja ovakvog pristupa. Funkcija fitnesa kod kompleksnih problema je najčešći ograničavajući faktor kod evolucijskih algoritama. Za probleme koji imaju kompleksno, višedimenzijsko rješenje, pronalazak optimalnog rješenja iziskuje skupe evaluacije u funkciji fitnesa. Funkcija kojoj tražimo optimum može imati svojstva da nije kontinuirana ili derivabilna, osim globalnog ima dosta lokalnih optimuma i sl. Zbog svojih ograničenja, metode koje koriste računanje derivacija ili gradijentnog spusta ovdje nisu primjenjive. Evolucijski algoritmi ne uzimaju u obzir navedene karakteristike funkcije već iterativno provjeravaju točke prostora. Zbog takvog

pristupa, samo jedna evaluacija potencijalnog rješenja može trajati od nekoliko sati do nekoliko dana. Stoga je potrebno koristiti drugačije pristupe optimizaciji za probleme kojima evaluacija traje netrivijalno dugo. Takvi problemi nazivaju se skupi problemi, a algoritmi koji ih rješavaju zovu se algoritmi za skupu optimizaciju i oni su u fokusu ovog rada.

2. Optimizacija

2.1. Metode optimizacije

Klasične metode optimizacije koriste se za pronalazak optimalnog rješenja, minimuma ili maksimuma kontinuirane i diferencijabilne funkcije. One koriste računanje derivacija u određivanju optimalnog rješenja, točnije pretpostavljaju da funkcija ima prvu i drugu derivaciju te da su one kontinuirane. Iz tog razloga njihova je primjena ograničena na primjerima koji sadrže funkcije cilja koje nemaju svojstva kontinuiranosti ili diferencijabilnosti. Međutim, klasične metode su podloga razvijanju naprednijih numeričkih metoda koje se koriste za današnje praktične probleme, nerješive prijašnjim metodama.

2.2. Numerička optimizacija

Optimizacija je važan alat u znanosti odlučivanja kao i analizi fizičkih sustava. Za optimizaciju, prvo je potrebno odrediti funkciju cilja koja opisuje kvantitativnu mjeru sustava kojeg proučavamo. Ta funkcija cilja određena je značajkama sustava koje nazivamo varijablama. Varijable su često na neki način ograničene, te je cilj optimizacije pronaći vrijednosti tih varijabli tako da one daju optimalnu vrijednost funkcije cilja.

Prvi korak procesa optimizacije je izgradnja modela koji opisuje zadani problem. Kako bi dobili zadovoljavajuće rješenje, model ne smije biti niti prejednostavan niti prekompleksan. Nad modelom se zatim primjenjuje algoritam optimizacije za pronalazak rješenja.

Model je zapravo pojednostavljena verzija odnosno aproksimacija stvarne funkcije. Njega možemo dobiti na primjer interpolacijom stvarne funkcije kako bi smanjili broj evaluacija. Algoritam dalje radi sa modelom čije evaluacije su znatno brže, uz evaluiranje stvarne funkcije samo kada je to nužno. Stoga vidimo da kompleksnost modela, odnosno koliko točno on predstavlja funkciju, određuje kvalitetu rezultata optimizacije. Iz tog razloga, osim brojnih algoritama postoje i različiti načini za izgradnju modela. Nakon što je nad modelom primijenjen algoritam, potrebno je provjeriti uspješnost pronalaska rješenja i primijeniti potrebne modifikacije.

Kao što je već spomenuto, ne postoji univerzalni algoritam za optimizaciju. Različit algoritmi kreirani su za različite vrste problema stoga je zadaća korisnika da izabere onaj algoritam koji je najprikladniji za njegov specifičan problem. Ovaj odabir određuje koliko brzo se problem može riješiti i zapravo može li se uopće doći do rješenja.

Definicija: Nešto kompletnija definicija od one u uvodu za optimizacijski problem je [1]:

Ako je zadano:

- **x** je vektor varijabli odnosno parametara
- f je funkcija cilja od **x** koju želimo maksimizirati ili minimizirati
- c je vektor ograničenja koje varijable trebaju zadovoljiti, to je vektorska funkcija od **x**

Tada pišemo:

$$\min f(\mathbf{x}) \quad \text{uz} \quad c_i(\mathbf{x}) = 0, \quad i \in \xi$$
$$c_i(\mathbf{x}) \geq 0, \quad i \in I$$

gdje su f i svaki c_i skalarne funkcije od **x**, a ξ i I skupovi indeksa.

U općenitoj definiciji, funkcija cilja f i ograničenja c mogu biti linearna, nelinearna ili konveksna. Zatim broj varijabli može biti velik ili malen te funkcije mogu biti diferencijabilne ili nediferencijabilne. Najveća razlika je u tome imaju li problemi ograničenja nad varijablama. Prema tome, optimizacijski problemi dijele se na neograničene i ograničene.

Neki algoritmi pronalaze lokalni optimum, tj. ne pronalaze uvijek najbolje rješenje, globalni optimum. Naravno, uglavnom težimo pronalasku globalnog optimuma, no to nije tako jednostavan zadatak. Ako pri minimizaciji funkcija cilja i područje pretraživanja oboje nisu konveksni, moguće je da postoji više lokalnih minimuma. Točka **x** je lokalni minimum funkcije f ako su na nekom području oko te točke sve vrijednosti funkcije f veće ili jednake vrijednosti u točki **x**. Na analogni način definira se i lokalni maksimum. Globalni minimum mora vrijediti na cijelom području pretraživanja. U slučaju konveksnog programiranja i linearnog programiranja koje zapravo spada u klasu konveksnog programiranja, svi lokalni optimumi su ujedno i globalni.

Nelinearni problemi mogu sadržavati lokalna rješenja koja nisu ujedno i globalna. Velik broj algoritama za rješavanje nelinearnih ili generalno nekonveksnih problema nemaju sposobnost razlikovanja lokalnog optimuma od globalnog. Globalna optimizacija je grana primijenjene matematike i numeričke analize koja se bavi upravo razvojem deterministčkih algoritama koji mogu u konačnom vremenu pronaći stvarno optimalno rješenje nekonveksnog ili nelinearnog problema. Jedna od karakteristika numeričke optimizacije je da se primjenjuje upravo na takvim nelinearnim funkcijama sa kontinuiranim domenama.

Optimizacije se mogu podijeliti na stohastičke i determinističke. Stohastička optimizacija koristi se kada je potrebno predvidjeti ili procijeniti neke nepoznanice kako bi dobili rješenja koja optimiziraju očekivano ponašanje modela. Drugim riječima, koriste se slučajni koraci pretrage. Deterministička optimizacija koristi modele koji su u potpunosti specificirani.

2.3. Optimizacijski algoritmi

Optimizacijski algoritmi su iterativni te započinju s jednom pretpostavkom o optimalnom rješenju te iz nje generiraju niz poboljšanih predviđanja dok ne dođu do rješenja. Prelasci između iteracija karakteriziraju različite strategije odnosno algoritme. Algoritmi koriste vrijednosti funkcije cilja i ograničenja, derivacije tih funkcija, dok neki spremaju informacije dobivene u prethodnim iteracijama. Bez obzira o strategiji koju koristi, svaki optimizacijski algoritam trebao bi imati sljedeća svojstva [1]:

- robusnost – dobro izvođenje na širokom spektru problema iz njima specifične klase problema, za sve razumne izbore početnih varijabli
- efikasnost – bez potrebe za puno procesnog vremena i memorije
- točnost – precizno identificiranje rješenja, bez prevelike osjetljivosti na pogreške u podatcima ili aritmetičkih pogrešaka pri zaokruživanju koje izvodi algoritam.

Ponekad gore navedena svojstva dolaze u konflikt odnosno za ostvarenje jednog svojstva, drugo mora biti uskraćeno. Na primjer, ako je metoda robusna onda može biti dosta spora.

Postoji čitav niz optimizacijskih algoritama, no u ovom radu detaljnije ću opisati Powellove algoritme.

2.4. Powellovi algoritmi

Michael J. D. Powell je poznat po svojem doprinosu u području numeričke analize, točnije nelinearne optimizacije i aproksimacije. Autor je nekoliko algoritama za optimizaciju poput:

- TOLMIN – tolerantan algoritam za linearno ograničenu optimizaciju
- COBYLA – algoritam za ograničenu optimizaciju linearnom aproksimacijom
- UOBYQA – algoritam za neograničenu optimizaciju kvadratnom aproksimacijom, bez derivacija
- NEWUOA – iznimno efikasan algoritam za neograničenu optimizaciju koji probleme rješava bez korištenja derivacija kvadratnom aproksimacijom
- BOBYQA – ograničena optimizacija kvadratnim modelom, također ne koristi derivacije.

Od navedenih algoritama koji koriste kvadratnu aproksimaciju, prvo je nastao UOBYQA. Algoritam je iterativan, koristi tehniku radijusa pouzdanosti i u svakoj iteraciji stvara kvadratni model interpolacijom određenog broja točaka funkcije cilja te minimizira stvoreni model unutar zadanog radijusa pouzdanosti. Idući algoritam kojeg je Powell razvio je NEWUOA.

NEWUOA predstavlja napredak u odnosu na prethodni algoritam zbog toga što za konstrukciju kvadratnog modela koristi manje točaka interpolacije. Zbog toga, NEWUOA se predlaže koristiti umjesto UOBYQA, ne samo zbog veće efikasnosti već i zbog mogućnosti rješavanja puno većih problema sa i do nekoliko stotina varijabli odnosno dimenzija.

Posljednji algoritam je BOBYQA koji također uvodi određena poboljšanja vezana za granice prostora problema.

U ovom radu zanimljiva su nam novija dva algoritma, NEWUOA i BOBYQA.

Prije nego što krenemo na same algoritme, potrebno je ukratko navesti svojstva algoritama koji ne koriste derivacije te pojasniti pojmove kvadratnog modela aproksimacije i radijusa pouzdanosti.

2.5. Optimizacija bez derivacija

Postoji klasa algoritama za optimizaciju koji ne koriste derivacije funkcije cilja već se oslanjaju samo na dostupnost vrijednosti funkcije. Ta kategorija algoritama se dugo proučavala i razvijala zbog sve većeg broja primjena u znanosti i inženjerstvu.

Cilj je optimizirati funkciju $f : R^n \rightarrow R$ na određenoj domeni, uz moguće donje i gornje granice varijabli. Pretpostavka je da derivacije funkcije cilja f nisu dostupne, nisu pouzdane ili ih je nepraktično za izračunavati, bilo zbog skupe evaluacije ili šuma. Takva optimizacija naziva se optimizacija bez derivata. Potrebno je naglasiti da za algoritam optimizacije bez derivata možemo reći da ne koristi derivacije funkcije cilja, no to ne znači da ne koristi računanje derivacija drugih funkcija poput zamjenske funkcije modela aproksimacije.

Za probleme koji spadaju u ovu kategoriju postoje algoritmi koji su direktni i oni koji su temeljeni na modelu. Iz samih naziva je jasno da direktni modeli tijekom potrage za optimumom koriste funkciju cilja direktno dok oni temeljeni na modelu konstruiraju zamjenski model za funkciju cilja te njega koriste za pretraživanje rješenja. U algoritmima koji su u fokusu ovog rada gradi se zamjenski kvadratni model odnosno kvadratna aproksimacija.

2.6. Kvadratna aproksimacija

Postojanje vjerodostojnog zamjenskog modela omogućuje pametnu provedbu pretraživanja pri čemu se koriste svojstva poput gradijenta, derivacija i određenih vjerojatnosnih funkcija modela. Za zadani problem uglavnom nije dostupan takav model već metode koje se koriste pri optimizaciji na početku moraju uzeti uzorke prostora pretraživanja odnosno funkcije cilja i izgraditi početni model aproksimacije. Tijekom daljnjih koraka algoritma optimizacije, rješenje se evaluira i model se iterativno optimizira i nadograđuje.

Powell je prvo predložio linearan model funkcije cilja unutar radijusa pouzdanosti koji se gradi interpolacijom stvarne funkcije u n točaka. Takav model je praktičan zbog svoje linearne složenosti, no cijena koju plaćamo je nemogućnost predstavljanja zakrivljenosti funkcije koju aproksimiramo. Zbog toga, Powell je predložio kvadratni model za kojeg trebamo najmanje $\frac{(n+1)(n+2)}{2}$ točaka za interpolaciju. Konkretni kvadratni model opisan je u kontekstu samog algoritma NEWUOA.

Metode radijusa pouzdanosti su jedan primjer korištenja zamjenskog modela koji je gladak, jednostavan za evaluaciju i točan na području određenom radijusom pouzdanosti

2.7. Metoda radijusa pouzdanosti

Metode radijusa pouzdanosti definiraju područje na dijelu funkcije koji je u fokusu trenutne iteracije (trenutna točka i područje oko nje). Radijus područja ovisi o adekvatnosti modela. Drugim riječima, radijus obuhvaća onoliko područje za koje se može reći da na njemu model predstavlja dobru aproksimaciju funkcije cilja. Na tom području se radi korak u cilju minimizacije modela. Korak minimizacije odnosno minimizator je definiran smjerom i duljinom. Ako pritom ne dođe do minimizacije tj. ne postignemo zadovoljavajuće smanjenje funkcije cilja, promjeni se veličina područja i bira se drugačiji minimizator.

Sama veličina radijusa pouzdanosti odnosno veličina područja bitno utječe na efektivnost svakog koraka, bilo ono preveliko ili premalo. Ako je područje preveliko, minimizator modela može bit dosta različit od minimizatora funkcije cilja. U tom slučaju potrebno je smanjiti radijus i ponovo provesti korak minimizacije. Ako je područje premalo, algoritam ne može napraviti veći korak prema minimizatoru funkcije cilja.

Veličina radijusa pouzdanosti može se birati obzirom na uspješnost u prijašnjim iteracijama algoritma. Pronalaskom adekvatnog odnosno pouzdanog kvadratnog modela funkcije cilja, koji točno predviđa ponašanje funkcije po koracima, radijus pouzdanosti se povećava kako bi se omogućilo provođenje većih koraka. U suprotnom ga je potrebno smanjiti. Konkretno, radimo omjer stvarne redukcije na funkciji cilja i predviđene redukcije na modelu te s obzirom na taj omjer mijenjamo veličinu radijusa pouzdanosti. Radijusom pouzdanosti najčešće je definirana kugla no mogu se koristiti i eliptična ili kvadratna područja.

2.8. NEWUOA

NEWUOA je opisan u znanstvenom radu [2] kao algoritam koji izvodi neograničenu optimizaciju pomoću iterativno konstruirane kvadratne aproksimacije funkcije cilja bez korištenja derivacija. Konkretno, algoritam traži najmanju vrijednost funkcije F(**x**) gdje je **x** vektor varijabli iz skupa R^n. Kvadratni model je nužan na početku svake iteracije te se koristi u kombinaciji sa metodom radijusa pouzdanosti kako bi podesio varijable funkcije.

Za rad algoritma i ostvarivanje ranije navedenih dobrih svojstava optimizacijskih algoritama poput točnosti i robusnosti, bitni su odabir inicijalnog kvadratnog modela, održavanje linearne neovisnosti među interpolacijskim uvjetima u prisutnosti grešaka kod zaokruživanja te stabilnost ažuriranja matrica koje se koriste za bržu reviziju modela Q.

Kvadratne aproksimacije su korisne jer, za razliku od linearnih, mogu dočarati zakrivljenost funkcije cilja te omogućavaju brzu konvergenciju iterativnih algoritama za neograničenu optimizaciju. Negativna strana je to što zahtijevaju $\frac{(n+1)(n+2)}{2}$ izračuna vrijednosti funkcije cilja, gdje parametar n predstavlja broj varijabli. U primjenama koje imaju veći n, takav račun je preskup stoga se u ovom algoritmu model Q izrađuje iz manjeg broja točaka interpolacije.

Kvadratni model Q stvara se interpolacijom funkcije cilja F u m točaka. Parametar m se zadaje od strane korisnika. Za vrijednost parametra m mora vrijediti:

$$m \geq n+2 \quad \text{i} \quad m \leq \frac{1}{2}(n+1)(n+2)$$

pri čemu se preporuča korištenje vrijednosti $m = 2n+1$. U svakoj iteraciji mijenja se samo jedna točka interpolacije stoga je prosječna količina posla po iteraciji reda $(m + n)^2$. Iz tog razloga moguće je raditi sa velikim vrijednostima parametra n.

Pri izgradnji novog modela on mora zadovoljavati jednadžbe interpolacije, no ima i slobode nad parametrima. Algoritam implementira tehniku za ažuriranje $\nabla^2 Q$ koja koristi tu slobodu za minimiziranje Forbeiusove norme $||\nabla^2 Q_{new} - \nabla^2 Q_{old}||_F$, gdje je Q_{old} trenutni model, a Q_{new} novi model. Na početku nove iteracije stvara se novi vektor varijabli $x_{new} = x_{opt} + d$, gdje je x_{opt} vektor koji predstavlja do sada najmanju pronađenu vrijednost funkcije F, a d je promjena varijabli unutar radijusa pouzdanosti. Svaka iteracija zapravo rješava podproblem minimizacije $Q(x_{opt} + d)$ pri čemu za d vrijedi $||d|| \leq \Delta$, gdje je Δ oznaka za radijus pouzdanosti.

Osim zadavanja parametra m, od korisnika se očekuje da zada funkciju cilja, inicijalni vektor x_0 i početnu i krajnju veličinu radijusa pouzdanosti, ρ_{beg} i ρ_{end} uz uvjet $\rho_{beg} \leq \rho_{end}$. Početne točke interpolacije uključuju zadani vektor x0. Vrijednost Δ ograničena je zadanim vrijednostima ρ_{beg} i ρ_{end} te se mijenja tijekom svake iteracije.

Algoritam završava kada radijus pouzdanosti padne ispod vrijednosti zadane tolerancije ili kada se dosegne maksimalan broj iteracija.

2.9. BOBYQA

Algoritam BOBYQA izvodi ograničenu optimizaciju pomoću iterativno konstruirane kvadratne aproksimacije funkcije cilja, također bez korištenja operacije deriviranja.

3. Implementacija

3.1. NLopt

NLopt je besplatna knjižnica otvorenog koda za nelinearnu optimizaciju. Sastoji se od sučelja za različite programske jezike, neki od kojih su C, C++, Fortran, Python, Matlab itd. Među brojnim algoritmima implementirani su i Powellovi algoritmi BOBYQA i NEWUOA. U pozadini se koristi izvorni kod u C-u koji se poziva iz C++ sučelja. Korisnik uz zadavanje funkcije cilja samo još mora zadati određene parametre.

3.2. Testovi algoritma NEWUOA

Nad algoritmom NEWUOA provedeni su testovi na problemima sa i do 200 varijabli. Rezultati za problem TRIGSABS uz dvije različite vrijednost m su sljedeći:

n	m = 2n + 1		m = ½ (n+1)(n+2)	
	#F	$\|\|x_{fin} - x^*\|\|_\infty$	#F	$\|\|x_{fin} - x^*\|\|_\infty$
20	1454	$1.0 * 10^{-8}$	4947	$4.8 * 10^{-9}$
40	3447	$1.6 * 10^{-8}$	24039	$5.9 * 10^{-9}$
80	7626	$1.2 * 10^{-8}$	-	-
160	16496	$2.2 * 10^{-8}$	-	-

Tablica 1: Razultati testova algortima NEWUOA na problemu TRIGSABS

Prvi stupac (n) predstavlja broj varijabli funkcije, #F je broj kalkulacija funkcije cilja, a $||x_{fin} - x^*||_\infty$ razlika između stvarnog minimuma i dobivenog rješenja. Iz tablice se vidi da su rezultati bolji za vrijednost m = 2n + 1, #F raste sa povećanjem broja varijabli, a greška u oba slučaja mala.

U idućoj tablici nalaze se rezultati testova za još nekoliko problema (znak * označava da vrijednost prelazi 500 000):

n	ARWHEAD	CHROSEN	PENALTY1	PENALTY2	PENALTY3
20	404	845	7476	2443	3219
40	1497	1876	14370	2455	16589
80	3287	4314	32390	5703	136902
160	8504	9875	72519	*	*

Tablica 2: Vrijednosti #F za 5 problema uz m = 2n+1

Kod problema ARWHEAD, CHROSEN i PENALTY1, najveća pogreška odnosno razlika između stvarnog i dobivenog minimuma je 6.1 * 10-6. Razvoj algoritma NEWUOA trajao je godinama, jer su numerički rezultati ovisni o greškama pri zaokruživanju. No na temelju testova pokazuje se da je točnost ostvarena stoga zaključujemo da su korištene tehnike stabilne.

4. Zaključak

Powellovi algoritmi NEWUOA i BOBYQA spadaju u klasu algoritama za skupu optimizaciju koji ne koriste derivacije. Iz tog razloga, moguće ih je koristiti na problemima koji imaju nederivabilnu funkciju cilja. Algoritmi daju dobre rezultate uz velik broj varijabli funkcije zbog toga što pri izradi zamjenskog modela biraju broj točaka interpolacije koji je dovoljno velik da dobro predstavi funkciju, no nije prevelik da negativno utječe na brzinu izvođenja algoritama. NEWUOA predstavlja zamjenu za prijašnji algoritam UOBYQA, dok Powellov najnoviji algoritam BOBYQA donosi određena poboljšanja, no u usporedbi sa NEWUOA daje bolje rezultate samo u nekim slučajevima. Implementacija ova dva algoritma u ECF omogućuje izvođenje već definiranih problema sustava ovim algoritmima te njihovu usporedbu sa drugačijim pristupima kod evolucijskih algoritama.

LITERATURA

1. Numerical Optimization, Jorge Nocedal, Stephen J, Wright, (Springer), 2000
2. Powell, M. J. D. (November 2004): The NEWUOA software for unconstrained optimization without derivatives
3. Powell, M. J. D. (June 2009): The BOBYQA algorithm for bound constrained optimization without derivatives
4. http://ab-initio.mit.edu/wiki/index.php/NLopt

Algoritmi za skupu optimizaciju

Sažetak

Cilj ovog završnog rada je upoznati se sa numeričkim metodama optimizacije koje se koriste na problemima čija evaluacija traje netrivijano dugo. Takva optimizacija naziva se skupa optimizacija. Kao primjeri algoritama za skupu optimizaciju uzeti su dva Powellova algoritma, NEWUOA i BOBYQA te implementirani u sustav ECF.

Ključne riječi: Optimizacija, numerička optimizacija, radijus pouzdanosti, kvadratna aproksimacija, Powell, NEWUOA, BOBYQA.

Algorithms for expensive optimization

Abstract

The goal of this project is to get to know the numerical optimization methods used when a single evaluation of a problem lasts for a nontrivial period of time. This kind of optimization is called expensive optimization. As examples of algorithms for expensive optimization, this project focuses on two Powell algorithms, NEWUOA and BOBYQA and describes their implementation into ECF.

Keywords: Optimization, numerical optimization, trust region, quadratic approximation, Powell, NEWUOA, BOBYQA.