XXX
Simpozijum o operacionim istraživanjima
SYM-OP-IS 2003
Hotel PLAŽA, Herceg-Novi, 30.09.-03.10.2003.

WEB SERVIS ZA PARALELNE EVOLUTIVNE ALGORITME

PARALLEL EVOLUTIONARY ALGORITHM WEB SERVICE

Vladimir Filipović
MATEMATIČKI FAKULTET, BEOGRAD, Studentski trg 16, tel. 630-151
Dušan Tošić
MATEMATIČKI FAKULTET, BEOGRAD, Studentski trg 16, tel. 630-151
Jozef Kratica
MATEMATIČKI INSTITUT, BEOGRAD, Knez Mihailova 35

Rezime: *U radu je opisan Web servis koji izvršava paralelne evolutivne algoritme. Motivacija za ovakav pristup je korišćenje računarskih resursa dostupnih na Internet-u kako bi se obezbedila računarska snaga neophodna za rešavanje različitih problema. Web servis i aplikacije su razvijene na Microsoft .NET platformi, korišćenjem programskog jezika C#. Ovakav pristup podržava različite tipove evolutivnoh algoritama (sa potpuno različitim vrstama reprezentacije, ukrštanja, selekcije, mutacije, politike zamene jedinki, različitim mrežnim topologijama, itd.).U realizaciji je korišćena objektno orjentisana paradgma, osobine interfejsi i indekseri, pa uvođenje i implementacija novih modifikacija evolutivnog algoritma zahteva minimalne izmene u kodu.*

KLJUČNE REČI: PARALELNI EVOLUTIVNI ALGORITAM, XML, SOAP, WEB SERVIS, .NET, C#

Abstract: This paper presents a Web Service that executes Parallel Evolutionary Algorithm. The motivation for this approach is to utilize computing resources available on the Internet in order to provide the computing power required for solving difficult problems. Web service and applications are developed on Microsoft .NET framework in the C# programming language. This approach supports different types of Evolutionary Algorithms (with completely different ways of representation, crossover, selection, mutation, generation replacement policies, different network topologies, etc.). Object-oriented paradigm, properties, interfaces and indexers are heavily used in this approach, so introduction and implementing new modifications of Evolutionary Algorithms requires minimal changing in code.

KEY WORD: PARALLEL EVOLUTINARY ALGORITHM, XML, SOAP, WEB SERVICE, .NET, C#

1. EVOLUTIVNI ALGORITMI

Prema najčešće korišćenoj klasifikaciji, Evolutivni algoritmi (nadalje označeni sa EA), zajedno sa Fazi logikama i Neuralnim mrežama spadaju u tzv. Soft computing. EA sadrže Genetske algoritme, Evolutivno programiranje, Evolutivne strategije, Genetsko programiranje i još neke slične tehnike.

Oblast primena EA su optimizacioni problemi (job shop scheduling, simple plant location, traveling salesman), razni konkretni problemi (ekonomsko modeliranje, projektovanje gasovoda), mašinsko učenje, teorija igara, itd. (videti [6], [7], [8]).

EA su zasnovani na Darvinovoj teoriji evolucije i Mendelovim zakonima nasleđivanja. EA imitiraju procese u prirodi kako bi rešile određeni problem.

Po Darvinovoj teoriji, u osnovi prirodne selekcije leže sledeća tvrđenja:

- Bolje prilagođene jedinke preživljavaju i one imaju jači uticaj na formiranje novih generacija.
- Jedinke u novoj generaciji se formiraju rekombinacijom genetskog materijala roditelja.
- S vremena na vreme dolazi do mutacije (izmene genetskog materiljala na slučajan način).

Veliki deo popularnosti EA duguju činjenici da proces određivanja novih rešenja ima paralelnu prirodu. Holland [4] je uočio paralenu prirodu reproduktivne paradigme i unutrašnju efikasnost paralelnog procesiranja desetak godina pre formalnog nastanka EA.

EA imaju sledeću strukturu:

- Prostor pretrage – prostor mogućih rešenja.
- Populacija – skup aktuelnih kandidata za rešenje; elementi populacije su jedinke.

- Prostor niski – sadrži reprezentacije (genetski kod) jedinki u populaciji.
- Funkcije za konverziju između elemenata u prostoru pretrage i elemenata u prostoru niski (kodiranje i dekodiranje).
- Skup genetskih operatora za generisanje novih niski, tj. novih jedinki (ukrštanje, mutacija, itd.).
- Funkcija prilagođenosti – ona određuje nivo prilagođenosti za svaku jedinku u populaciji.
- Stohastička kontrola genetskih operatora.

Osnovni koraci kod EA su:

1 Inicijalizacija – generisanje inicijalne populacije (najčešće) slučajnim sampliranjem.
2 Evaluacija – izračunavanje prilagođenosti za sve jednke u populaciji.
3 Selekcija – izbor bolje prilagođenih jednki iz populacije koje će preneti svoj genetski materijal u nove generacije.
4 Rekombinacija (uključuje ukrštanje i mutaciju) – izmena reprezentacije jedinke.
5 Ponavljanje koraka 1 do 4 do ispunjenja kriterijuma završetka.

Sledeće karakteristike razlikuju EA od drugih metoda za rešavanje problema:

- EA rade sa kodovima koji predstavljaju parametre, a ne sa samim parametrima.
- EA simultano pretražuju po više tačaka, pa su stoga veoma robusni.
- EA koriste probabilistička, a ne deterministička pravila prelaska.
- EA (u svojoj čistoj formi) ne eksploatišu dodatne informacije o prirodi problema.

2. PARALELNI EVOLUTIVNI ALGORITMI

Osnovna ideja kod mnogih paralelnih programa je da podele zadatak u podzadatke i da korišćenjem više procesora simultano rešavaju podzadatke. Ovaj pristup može biti iskorišćen na različite načine, što dovodi do različitih metoda za paralelizaciju EA [3]. Neki metodi paralelizacije menjaju ponašanje algoritma, a neki ne. Neko od metoda mogu da eksploatišu paralelne računarske arhitekture sa ogromnim brojem procesora, dok su druge bolje prilagođene paralelnim računarima sa manjim brojem moćnijih procesora.

Prvi metod paralelizacije EA je globalna paralelizacija. Kod ***Globalnih paralelnih EA*** postoji samo jedna populacija (kao i kod klasičnih serijskih EA), ali se određivanje prilagođenosti jednki i izvršavanje genetskih operatora eksplicitno paralelizuju. S obzirom da je populacija jedinstvena, selekcija uzima u obzir sve jedinke i svaka jedinka ima šansu da se rekombinuje sa ma kojom drugom, pa ponašanje algoritma ostaje nepromenjeno. Ovaj metod se relativno lako implementira i daje značajno ubrzanje ukoliko vreme komunikacije nije dominantni deo vremena izvršavanja EA (npr. tamo gde je funkcija prilagođenosti veoma složena, pa izračunavanje prilagođenosti traje jako dugo).

Grubo usitnjen paralelni EA koristi sofisticiraniju ideju. U ovom slučaju, populacija je podeljena na podpopulacije koje najveći deo vremena evoluiraju nezavisno jedna od druge, ali koje s vremena na vreme rezmene jedinke. Ova razmena jedinki se zove migracija i nju kontroliše nekoliko parametara. Grubo usitnjen paralelni EA uvodi fundamentalne promene u način rada EA i njegovo ponašanje se razlikuje od ponašanja prostog EA. Grubo usitnjen paralelni EA je poznat i pod nazivom Distribuisani EA, zato što su ovi algoritmi obično implementirani na MIMD računarima sa distribuisanom memorijom. Ova klase paralenih EA se ponegde još naziva i Ostrvski Paralelni EA, jer u genetici populacija postoji model strukturisanja populacije koji razmatra relativno izolovane podpopulacije i taj model se zove ostrvski model.

Treći pristup u paralelizaciji EA koristi fino usitnjen paralelizam. ***Fino usitnjen paralelni EA*** deli populaciju u veliki broj veoma malih podpopulacija (bilo bi idealno da postoji tačno jedna jedinka za svaki procesni element koji je na raspolaganju). Ovaj model je prilagođen masivnim paralelnim računarima.

Poslednji metod paralelizacije EA koristi neke kombinacije prethodna tri metoda. Ova klasa evolutivnih algoritama se naziva ***Hibridni paralelni EA***. Kombinovanje tehnika paralelizacije dovodi do algoritama koji sadrže pozitivne karakteristike uključenih komponenti.

3. XML i SOAP

XML tj. Proširiv jezik za označavanje (eng. eXxtensible Markup Language) predstavlja način opisivanja strukturisanih podataka.

XML koristi skup tagova za izdvajanje elemenata u okviru podatka. Svaki elemenat enkapsulira deo podatka koji može biti veoma jednostavan ili vrlo složen.

XML je nezavistan od platforme, jednostavan i široko prihvaćen standard. On razdvaja podatke od prezentacije [2], što omogućuje integraciju podataka iz različitih izvora.

Ipak, XML nije odgovarajući za svaku situaciju. XML dokumenti imaju tendenciju da budu mnogo opširniji od binarnih formata koje zamenjuju. Stoga, oni zahtevaju veći propusni opseg mreže i zauzimaju veći memorijski prostor, ili se vrši njihova kompresija, što zahteva veće procesorsko vreme. Nadalje, parsiranje XML-a je obično sporije i memorijski zahtevnije od parsiranja visoko optimizovanih binarnih formata.

Pažljiv dizajn aplikacije i korišćenja XML-a u njoj može zaobići neke od ovih problema.

Što se prethodno uočenog problema tiče, treba istaći da način izvršavanja EA ne generiše veliki mrežni saobraćaj - klijent pošanje parametre algoritma (u XML formatu) servisu, servis izvrši algoritam i pošalje rezultate (opet u XML formatu) nazad klijentu. Dakle, u svim netrivijalnim slučajevima, vreme utrošeno na komunikaciju je skoro zanemarljivo mali deo vremena koje servis troši na izvršavanje algoritma.

SOAP tj. Prosti protokol za pristup ojektima (eng. Simple Object Access Protcol) je industrijski protokol koji je dizajniran za prenos XML poruka kroz veći broj protokola. Specifikacija SOAP-a koristi HTTP protokol kao primer koji demonstrira vezivanje protokola. Ova specifikacija takođe opisuje [2] kako SOAP poruke mogu biti vezane i na druge protokole.

SOAP specificira standardni format, čiji deo predstavljaju podaci koji su enkodirani u XML dokument. Taj XML dokument se sastoji od tag-a-korena ***Envelope***, koji dalje mora sadržati ***Body*** elemenat i eventualno ***Header*** elemenat.

4. .NET okvir, Web servisi i C#

.NET okvir (eng. .NET framework) je nova računarska platforma dizajnirana od strane Microsoft-a, da bi, po rečima kreatora, uprostila razvoj aplikacija u široko ditribuisanom Internet okruženju [1]. Ovaj okvir sadrži dve osnovne komponente: izvršni modul za zajednički jezik (common language runtime - CLR) i .NET biblioteku klasa.

1. CLR je osnova .NET okvira. CLR može biti posmatran kao agent koji upravlja kodom tokom izvršenja, obezbeđujući osnovne servise kao što su upravljanje memorijom, upravljanje nitima, pri čemu se istovremeno obezbeđuje striktna sigurnost koda. Koncept upravljanja kodom je fundamentalni princip modula za izvršavanje. Kod koji se pogađa izvršni modul naziva se upravljani kod, a kod koji zaobiliazi modul je neupravljani kod.
2. Biblioteka klasa .NET okvira obimna objektno-orjentisana kolekcija klasa raspoloživih za ponovno korišćenje, koje se koriste u razvoju različitih tipova aplikacija, počev od klasičnih aplikacija koje se izvršavaju u komandnoj liniji, preko GUI aplikacija, pa sve do aplikacija zasnovanih na poslednjim inovacijama koje obezbeđuju ASP.NET i Web servisi.

Web Servisi, važan korak u razvoju Web-baziranih tehnologija, su distribuisane serverske komponente koje liče na Web sajtove. Oni predstavljaju moćan metod za pružanje servisa kojima se (preko Internet-a) može pristupiti programskim putem. Ove komponente, za razliku od Web sajtova, nemaju UI, već su dizajnirane tako da budu korišćene od strane drugih aplikacija, kao što su tradicionalne klijentske aplikacije, Web-bazirane aplikacije, ili čak drugi Web servisi.

Web servisi mogu biti korišćeni interno (od strane jedne aplikacije) ili izloženi eksterno preko Internet-a za korišćenje od strane većeg broja aplikacija. S obzirom na to da su dostupni preko standardnih interfejsa, Web servisi dopuštaju da heterogeni sistemi rade zajedno kao jedna mreža izračunavanja.

Web servisi koriste XML-bazirane poruke kao fundamentalni način za prenos podataka, kako bi se premostili jazovi između sistema koji koriste neuklapajuće modele komponenti, operativne sisteme i programske jezike. Preciznije rečeno, Web servisi su izgrađeni na standardima kao što su SOAP (protokol za poziv udaljenih metoda), XML (proširiv format podataka), and WSDL (eng. Web Service Description Language – jezik za opis Web servisa).

Jedna od osnovnih karakteristika Web servisa je visok nivo apstrakcije između implementacije servisa i korišćenja servisa. Korišćenjem XML-zasnovanih poruka kao mehanizma za kreiranje i pristup servisu, klijent Web servisa i sam Web servis su oslobođeni potrebe da znaju bilo šta o onom drugom, osim ulaza, izlaza i lokacije (adrese).

.NET okvir poštuje sve komunikacione standarde na kojima se zasnivaju Web servisi, što omogućuje interoperabilnost sistema. Drugim rečima, klase iz skupa koje obezbeđuje .NET okvir poštuju komunikacione standarde (SOAP, XML, WSDL), pa se programer može fokusirati na logiku servisa koji implementira, bez potrebe da brine o komunikacionoj infrastrukturi koju zahteva razvoj distribuisanog softver-a [1].

Web servisi podržavaju i sinhronu i asinhronu komunikaciju između klijenta i servera koji je domaćin Web servisu. Kod sinhrone komunikacije, klijent pošalje zahtev za servis i čeka na odgovor. Kod asinhrone komunikacije klijent po slanju zahteva nastavlja sa radom tj. obradom, a rezultate koje kasnije dobije od serisa obrađuje tek onda kada mu budu na raspolaganju. Korišćenje asinhrone komunikacije poboljšava iskorišćenost sistema i izbegava klijentovo kašnjenje zbog čekanja na odgovor Web servisa. Uspešno korišćenje asinhrone komunikacije je čvrsto povezano sa višenitnim programiranjem.

C# je programski jezik dizajniran za gradnju širokog opsega aplikacija koje se izvršavaju u .NET okviru. Na sam dizajn jezika C# je, pored C++, najviše uticala Java, mada je uočljiv i uticaj Smalltalk-a i Pascal-a. C# is jednostavan, moderan, objektno-orijentisan jezik. C# kod se kompajlira do nivoa upravljanog koda, što znači da koristi servise CLR. Prednost ovakvog pristupa uključuje međuoperabilnost sa drugim programskim

jezicima, garbage collection, poboljšanu sigurnost i pojačanu podršku za upravljanje verzijama [1].

C# ima sledeće karakteristike:
- Podrška za punu interoperabilnost sa COM+ 1.0 i sa servisima .NET okvira, uz blizak pristup zasnovan na bibliotekama.
- XML podrška za interakciju Web-baziranih komponenti.
- Upravljanje verzijama, što olakšava administraciju, implementaciju i isporuku.
- U sam jezik su ugrađeni atributi, osobine, interfejsi, indekseri, refleksija, događaji, a u biblioteku klase i interfejsi za olakšanje višenitnog programiranja.

5. DIZAJN WEB SERVISA I APLIKACIJA

Web servis za Evolucionarni algoritam, nazvan EaWebService je razvijen na programskom jeziku C#, u Microsoft .NET okviru. On, slično kao [5] demonstrira prednosti distibuisanog evolutivnog programiranja na Internet-u. Nadalje, ovakav pristup demonstrira i klijent-server obradu podataka, kao i korišćenje XML-a i SOAP-a u Internet izračunavanjima.

EaWebServis je dizajniran i implementiran tako da istovremeno podržava različite tipove EA (sa potpuno različitim vrstama reprezentacije, ukrštanja, selekcije, mutacije, politike zamene jedinki, različitim mrežnim topologijama, itd.). U ovom pristupu je korišćena objektno-orjentisana paradigma, osobine, interfejsi i indekseri, pa uvođenje i implementacija novih modifikacija EA zahteva minimalnu promenu koda. Ovakvo rešenje dopušta unificiran pogled na performanse i ponašanje različitih EA tehnika i, u isto vreme, postiže efikasno izvršavanje EA.

EaWebService može biti korišćen u svim scenarijima koji uključuju optimizaciju pomoću EA. U isto vreme, servis može biti korišćen i na način kao kod Evolutivnih strategija ili Genetskog programiranja.

Pri izvršavanju EA, aplikacija tj. klijent šalje Web servisu XML nisku sa parametrima EA koji će biti izvršen. Neki od ovih parametara (kao što su zamena populacije, selekcija, kriterijum završetka, generator slučajnih brojeva, izveštaji) ne zavise od reprezentacije jedinke dok neki drugi (npr. tip populacije, inicijalizacija, instanca, ukrštanje i mutacija) zavise. Svaki parametar je XML čvor sa elementima koji određuju vrednost parametara. EaWebService podržava tri tipa reprezentacije jedinki (kao binarna niska, kao broj u pokretnom zarezu i drvoidna reprezentacija). U okviru parametara EA specificira se i problem (u XML reprezentaciji) koji se rešava (npr. zapis funkcije je predstavljen u XML formatu, pa isti kod može optimizovati različite funkcije), inicijalni podaci (kada je to potrebno), te veze među Web servisima iste vrste koji zajednički rade (paralelno) kako bi se odredilo rešenje.

Web servis vraća (u XML formatu) nađeno rešenje i neophodne dodatne podatke. Web servis dopušta aplikaciji da prekine tekuće izvršavanje i vrati dotadašnji rezultat u ma kom momentu izvršavanja. Ovaj servis takođe (korišćenjem refleksije) dopušta aplikaciji da dobije informacije o klasama koje predstavljaju pojedine delove EA i da, na taj način, omogući lakšu i bržu kreaciju EA.

Prethodno pomenuta lakoća nadogradnje Web servisa sledi iz činjenice da je servis čvrsto zasnovan na interfejsima. Tako, kad god se pristupa metodu ili osobini neke klase van te klase, to se realizuje uzimanjem interfejsa i pristupom metodu odnosno osobini interfejsa (a ne direktno klase).

Po kreiranju gore opisanog servisa, istraživač (čak i kad nije ekspert za EA) može eksperimentisati (podešavanjem vrednosti XML elemenata) radi dobijanja najboljih parametara za rešavanje konkretnog problema, ili lako preći sa jednog problemskog domena na drugi. Aplikacija koja koristi servis samo treba da oformi XML niske koje su parametri EA i da prihvati i prikaže rezultat (koji je takođe XML niska).

LITERATURA

[1] Archer T.: Inside C#, *Microsoft Press, Redmond, WA*

[2] Baartse M, Blair R, Bolognese L, Dalvi D, Hahn S, Haines C, Homer A.: Proffesional ASP XML, *Wrox Press, Birmingham*, 2000.

[3] Cantu-Paz E.: Efficient and Accurate Parallel Genetic Alghorithms, *Kluwer Academic Publishers, Boston*, 2000.

[4] Holland J. H.: Adaptation in Natural and Artificial Systems, *MIT Press, Cambridge*, Massachusetts, 1992.

[5] Chong F. S.: A Java based Distributed Approach to Genetic Programming on the Internet, *MS disertation, University of Birmingham, School of Computer Science, UK*, 1998.

[6] Filipović V, Kratica J, Tošić D, Ljubić I.: Fine Grained Tournament Selection for the Simple Plant Location Problem, *Proceedings of the 5th Online World Conference on Soft Computing Methods in Industrial Applications - WSC5*, pp. 152-158, September 2000,.

[7] Kratica J, Tošić D, Filipović V, Ljubić I.: Genetic Algorithm for Designing a Spread-Spectrum Radar Polyphase Code, *Proceedings of the 5th Online World Conference WSC5*, pp. 191-197, september 2000.

[8] Kratica J, Filipović V, Tošić D, Ljubić I.: Solving the Simple Plant Location Problem by Genetic Algorithm, *RAIRO Operations Research, Vol. 35, No. 1*, pp. 127-142, 2001.