Daniel Silađi

Genetski algoritmi i Problem trgovačkog putnika

Rad se bavi primenom genetskih algoritama (GA) na problem trgovačkog putnika (TSP). U prvom delu rada data je definicija problema trgovačkog putnika, istorijski pregled problema i algoritama za njegovo rešavanje. Takođe su navedeni osnovni pojmovi vezani za genetske algoritme. Originalni doprinos rada je program koji rešava problem trgovačkog putnika pomoću genetskog algoritma. Algoritam je implementiran u programskom jeziku C++, uz pomoć biblioteka SDL, OpenGL i GALib. U ovom radu je dat i tehnički opis, način korišćenja programa, kao i poređenje sa rezultatima koji se dobijaju sa drugim algoritmima iz literature.

Uvod

Šta je problem trgovačkog putnika?

Neformalno, problem trgovačkog putnika (engl. Travelling Salesman Problem – TSP) možemo formulisati na sledeći način: Dato je n gradova, poznate su sve udaljenosti među njima; trgovački putnik treba da obiđe sve te gradove i da se na kraju vrati u grad odakle je krenuo, a da pri tome pređe najkraću razdaljinu.

Formalna definicija problema trgovačkog putnika glasi:

Definicija 1. Problem trgovačkog putnika – TSP. Ako je dat kompletan neusmeren težinski graf, sa nenegativnim celobrojnim težinama naći Hamiltonovu putanju sa najmanjom težinom.

Pored optimizacijske varijante ovog problema, postoji i tzv. forma problema odlučivanja:

Definicija 2. TSP-odlučivanje. Ako je dat kompletan težinski graf i pozitivan realan broj L, treba odrediti da li postoji Hamiltonov put kraći od L?

Istorija. TSP prvi put spominju irski i britanski matematičari Sir William Rowan Hamilton i Thomas Penyngton Kirkman sredinom XIX veka. Hamilton je 1857. godine izmislio Ikozijansku igru, u kojoj je cilj da se nađe zatvorena putanja koja prolazi kroz (neke) ivice dodekaedra i svako njegovo teme tačno jednom (slika 1). U današnjoj terminologiji, to bi značilo naći Hamiltonovu putanju u grafu kome su čvorovi temena, a ivice ivice dodekaedra.

Nakon njih, sledeća osoba koja se ozbiljno bavila TSP-om je bio Karl Menger, 1930-ih godina. On je u svom radu spomenuo trivijalni brute-force algoritam, konstatovao suboptimalnost heuristike „najbližeg komšije“ i definisao TSP onako kako se danas definiše.

U narednim decenijama, problem je postajao sve više popularan i izučavan od strane mnogih matematičara, informatičara, fizičara itd.

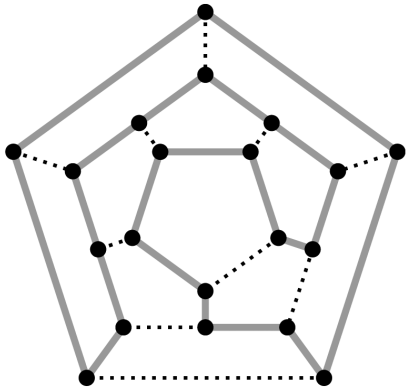
Algoritmi za rešavanje TSP

Očigledno, TSP se može rešiti brute-force algoritmom (složenosti $\mathcal{O}(n!)$) isprobavajući sve moguće permutacije gradova. Danas nije poznat ni jedan determinističko polinomni algoritam za njegovo rešavanje – TSP spada u klasu NP tvrdih problema. Međutim, Datzig, Fulkerson i Johnson (Dantzig *et al*. 1954) su u svom radu dali metodu koja se zasniva na celobrojnom programiranju, i iako je još uvek eksponencijalne složenosti, znatno je brža od svih ostalih do sada poznatih metoda (Applegate *et al*. 2003). Postoje takođe i aproksimativni algoritmi, npr. genetski algoritmi, Lin-Keringan potezi, tabu pretraživanje, i slično.

Daniel Silađi (1995), Novi Sad, Fruškogorska 15, učenik 1. razreda Gimnazije „Jovan Jovanović Zmaj“ u Novom Sadu

MENTORI:
Miloš Savić, Matematički fakultet Univerziteta u Novom Sadu

Milan Gornik, Wildbit – USA, Novi Sad



Slika 1. Ikozijanska igra

Figure 1. Icosian game

Šta su genetski algoritmi i čemu služe?

Genetski algoritmi (u daljem tekstu GA) su porodica algoritama inspirisanih Darvinovom teorijom evolucije. Prvi radovi iz ove oblasti su nastali 60-tih godina prošlog veka, ali se u većini izvora kao tvorac ove oblasti uzima John Holland. On je 1975. godine napisao knjigu „Adaptation in natural and artificial systems" (Holland 1992). GA imaju za cilj rešavanje problema kombinatorne optimizacije, tj. problema u kojima se traži minimum ili maksimum neke funkcije. Pošto je prostor pretraživanja (skup rešenja) ponekad prevelik (i njegovo kompletno pretraživanje se ne može izvršiti u nekom doglednom vremenu), a optimalno rešenje nije neophodno (prihvata se i neko približno, suboptimalno rešenje), mogu se koristiti genetski algoritmi.

U GA svako pojedinačno rešenje je predstavljeno jednom jedinkom, koja sadrži gene, tj. delove rešenja. Nad njima se vrše operatori mutacije i ukrštanja (kao mehanizam pretrage) i selekcije (usmerava algoritam ka perspektivnim delovima pretraživačkog prostora).

Osnovni operatori u genetskim algoritmima

Definicija 3. **Fitness funkcija** je genetski operator koji svakoj jedinki dodeljuje vrednost f_i koja oslikava kvalitet te jedinke. Kod TSP, to je ukupna dužina puta predstavljenog tom jedinkom.

Sledeći operator je operator selekcije. U osnovnim crtama, princip rada selekcije je sličan kao u stvarnom životu: cilj je da se odabere genetski materijal koji će se preneti u narednu generaciju. Kod ovog operatora je bitno sačuvati raznovrsnost, kao i kvalitet genetskog materijala, inače će se dobra rešenja možda zauvek izgubiti.

Definicija 4. **Selekcija** je genetski operator koji bira jedinke koje će se ukrštati i/ili preneti u sledeću generaciju.

Definicija 5. **Ukrštanje** je genetski operator koji na osnovu dve date jedinke (roditelja) konstruiše dve nove jedinke (decu), koje su nastale kombinovanjem genetskog materijala roditelja.

Cilj ovog operatora je da se ukrštanjem dva postojeća rešenja dobiju nova, obično kvalitetnija rešenja.

Definicija 6. **Mutacija** je genetski operator koji (uglavnom) nasumično mutira genetski materijal date jedinke.

Mutacijom se vraća raznovrsnost genetskog materijala u populaciju. Ipak, prekomerna mutacija može svesti algoritam na nasumičnu pretragu, pa se zato uvodi verovatnoća mutacije, koja je uglavnom manja od 5%.

Korišćene metode

Konstrukcija početnog rešenja

Za konstrukciju početnog rešenja korišćen je greedy algoritam koji od zadatog početnog grada gradi put tako što uvek bira grad najbliži poslednjem dodatom gradu. Iako je ovaj pristup suboptimalan, u praksi najčešće daje prilično dobra rešenja koja se od optimalnog retko razlikuju za više od 10 procenata.

Selekcija

U ovom radu su korišćena dva algoritma za selekciju:

Prosta, rulet selekcija. Svaka jedinka ima verovatnoću selekcije direktno proporcionalnu njenoj fitness-vrednosti, tj:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

Selekcija zasnovana na rangu. Svaka jedinka ima verovatnoću selekcije direktno proporcionalnu njenom rangu, tj:

$$p_i = \frac{r_i}{\sum_{j=1}^{n} r_j}$$

Napomena. Promenljiva f_i predstavlja vrednost fitness-funkcije i-te jedinke u populaciji, p_i verovatnoću da baš i-ta jedinka bude odabrana prilikom selekcije, a r_i rang i-te jedinke, tako da najkvalitetnija jedinka (gledajući fitness) ima rang n, a najnekvalitetnija 1. Takođe, važi pretpostavka da u populaciji ima tačno n jedinki.

Ukrštanje

Korišćena su dva algoritma za ukrštanje:

Ukrštanje rekombinacijom grana (edge recombination). Ovaj algoritam od dva roditelja pravi jedno dete, i to na sledeći način:

Algoritam:

1. Konstruiše matrice povezanosti:
$$\begin{matrix} g_1: & g_n & g_2 \\ g_2: & g_1 & g_3 \\ \vdots & \vdots & \vdots \\ g_n: & g_{n-1} & g_1 \end{matrix}, \quad \begin{matrix} g'_1: & g'_n & g'_2 \\ g'_2: & g'_1 & g'_3 \\ \vdots & \vdots & \vdots \\ g'_n: & g'_{n-1} & g'_1 \end{matrix}$$
2. Napravi njihovu uniju:
$$\begin{matrix} g_1 = g'_{i1}: & g_n & g_2 & g'_{i1-1} & g'_{i1+1} \\ g_2 = g'_{i2}: & g_1 & g_3 & g'_{i2-1} & g'_{i2+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g_n = g'_{in}: & g_{n-1} & g_1 & g'_{in-1} & g'_{in+1} \end{matrix}$$
3. Napravi novu praznu listu L
4. Odabere početni čvor L_1
5. Odabere čvorove $L_2,\ldots L_{n-1},L_n$ po sledećem principu: L_{i+1} je sused čvora L_i, takav da L_{i+1} ima najmanji mogući broj suseda, i ne nalazi se već u L
6. Lista L je nova jedinka.

Ukrštanje parcijalnim mapiranjem. Ovaj algoritam proizvodi dva potomka od dva roditelja.

Algoritam:

1. Odaberu se dva čvora po slučajnom izboru, a i b
2. Iz roditelja 1 se kopira put od a do b u dete 1, a iz roditelja 2 u dete 2.
3. Dokle god to može (tj. dok se svaki čvor javlja samo jednom u detetu), algoritam popunjava pozicije u detetu 1 sa putevima iz roditelja 2 i u detetu 2 sa putevima iz roditelja 2
4. Ostatak čvorova se u decu dodaje na slučajan način.

Mutacija

Korišćena su dva algoritma za mutaciju:

Greedy mutacija. Biraju se dva slučajna grada, i gleda se da li se njihovom zamenom dobija kraći put.

2opt mutacija.

Algoritam:

1. Biraju se dva para susednih gradova, (a_1,b_1) i (a_2,b_2)
2. Ako je $d(a_1,b_2)+d(a_2,b_1) < d(a_1,b_1)+d(a_2,b_2)$, ivice (a_1,b_1) i (a_2,b_2) se brišu i dodaju se ivice (a_1,b_2) i (a_2,b_1).

Rezultati

Tehnički detalji. Program je napisan u programskom jeziku C++. Kao osnova za genetski algoritam, korišćena je biblioteka GALib, verzija 2.4.7, autora Mathew Wall-a sa MIT-a. Za grafički prikaz su korišćene biblioteke SDL i OpenGL.

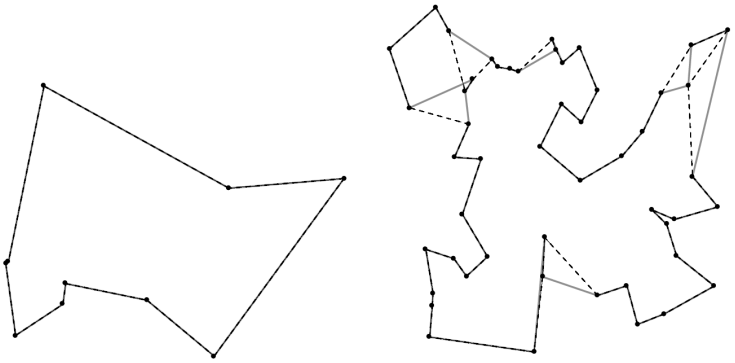
U planu je izrada korisničkog interfejsa, gde korisnik može da bira između grafičkog i tekstualnog unosa podataka (koordinata gradova). U rar fajlu se nalazi sve što je potrebno za pokretanje na 64-bitnom Windows-u. Makefile je pravljen takođe za Windows, i potrebno je imati SDL i OpenGL header-e negde gde kompajler može da ih nađe.

Način upotrebe programa. Glavni algoritam se nalazi u programu pod imenom tsp.exe. Program se koristi unošenjem koordinata gradova u fajl, i to u sledećem formatu:

$$\begin{matrix} ID_1 & X_1 & Y_1 \\ ID_2 & X_2 & Y_2 \\ \vdots & \vdots & \vdots \\ ID_n & X_n & Y_n \end{matrix}$$

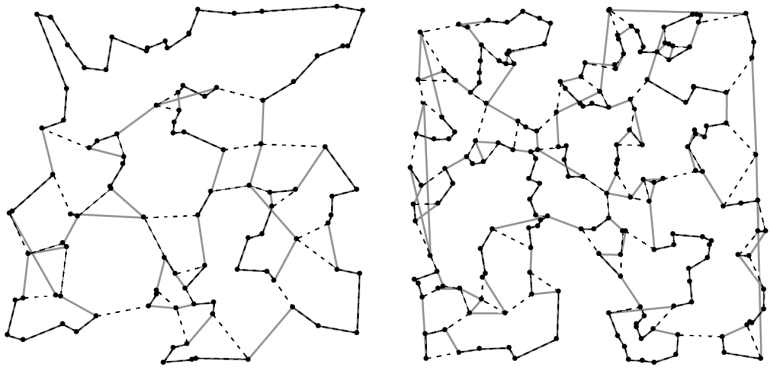
pri čemu su koordinate bilo koji realni brojevi, a ID_i ceo broj. Tako snimljeni fajl se samo prevuče na tsp.exe. Tog trenutka će započeti inicijalizacija, i za par trenutaka pojaviće se grafički prikaz optimalnog rešenja. Program prestaje sa radom kada populacija potpuno konvergira (za 250 gradova za par sekundi, a za 1000, to je otprilike 2 min). U konzoli se vidi dužina najboljeg i najgoreg puta u trenutnoj generaciji, kao i njihov odnos.

Primeri. Na slikama 2 i 3 prikazane su putanje generisane algoritmom datim u ovom radu (siva puna linija), kao i putanje generisane programom LKH (isprekidane crne linije) (Lin i Kernighan 1973).



Slika 2. Primeri sa 10 i 50 gradova

Figure 2. Examples for 10 and 50 cities



Slika 3. Primeri sa 100 i 200 gradova

Figure 3. Examples for 100 and 200 cities

Slike su dobijene pomoću perl skripte koja je na osnovu tekstualnog izlaza navedenih programa generisala SVG slike, koje su dalje konvertovane u .png format programom Inkscape.

Tabela 1. Uporedni prikaz rezultata (dužine puteva) opisanog algoritma (GA) i Lin-Keringhan heuristike (LKH)

Broj gradova	Rezultat GA	Rezultat LKH	$\frac{\text{GA}}{\text{LKH}}$
10	93420.1	93420.09	100%
20	126105	123182.65	102.37%
50	183528	180723.38	100.01%
100	259950	247460.57	105.05%
200	426510	356127.22	119.76%

U tabeli 1 prikazani su uporedni rezultati opisanog algoritma i Lin-Keringhan heuristike (Lin i Kernighan 1973).

Zaključak

Na osnovu testiranih primera i algoritama, vidi se da GA za manji broj gradova daje optimalno rešenje, i to u realnom vremenu. Za veći broj gradova (par stotina), algoritam radi ispod 3 minuta i daje rešenja u proseku 20% gora nego Lin Keringhan heuristika. Razlog za to je da se algoritam prekida kada sve jedinke konvergiraju ka najboljoj, odakle je dalji razvoj moguć samo mutacijom.

Prema tome, jedna od mogućnosti za poboljšanje ovog algoritma je inkorporacija drugih heuristika, kao sto je LKH. Međutim, opšti utisak autora (nastao tokom izrade i testiranja programa) je da GA možda i nisu najbolja metoda za rešavanje TSP. Za dalji rad verovatno najviše smisla ima okrenuti se celobrojnom programiranju i metodama opisanim u Applegate *et al*. (2003).

Literatura

Applegate D., Bixby R., Chvátal V., Cook W. 2003. Implementing the dantzigfulkerson-johnson algorithm for large traveling salesman problems. *Mathematical programming*, **97**: 91.

Cormen T. H., Stein S., Rivest R. L., Leiserson C. E. 2001. *Introduction to Algorithms*. McGraw-Hill Higher Education

Dantzig G., Fulkerson R., Johnson S. 1954. Solution of a large-scale travelingsalesman problem. *Journal of the operations research society of America*, **2**: 393.

Holland. J. 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press

Lin S., Kernighan. B. 1973. An effective heuristic algorithm for the travelingsalesman problem. *Operations research*, **21** (2): 498.

Daniel Silađi

Genetic Algorithms and the Travelling-Salesman Problem

This paper explores possible applications of Genetic algorithms (GA) to solve the Traveling-Salesman Problem (TSP). The referent solution for comparison is Lin-Keringhan heuristics (LKH). In this work a program solving traveling-salesman problem using a genetic algorithm is presented. The program is developed using C++ and is primarily based on Galib library which was used to implement the GA. Graphical user interface was done using sdl and openGL libraries. The results show that ga is giving optimal solutions for a small number of cities, in real time. For larger numbers of cities (several hundreds), the algorithm is working under three minutes and proposes solutions which are in average 20% worse than those from LKH. The reason is the fact that the GA is stopped when all cells are convergating towards the best solution, and from that point only mutations can be performed.