Parallel Beam Search za problem trgovačkog putnika

Nikola Markov

215/2018

Sadržaj

1 Uvod

Problem trgovačkog putnika (eng. Travelling salesman problem - TSP) je NP-tezak kombinatorno optimizacioni problem. Predstavlja problem gde za dat skup gradova treba konstruisati najkraći mogući put koji obilazi svaki grad tačno jednom I vraća se u početni grad.

Efikasna rešenja ovog problema su veoma značajna jer imaju praktičnu primenu, ali i služe kao reper za efikasnost i kvalitet drugih algoritama. Naučnici i istraživači se bave traženjem efikasnih rešenja ovog problema pomoću raznih heuristika. [1]

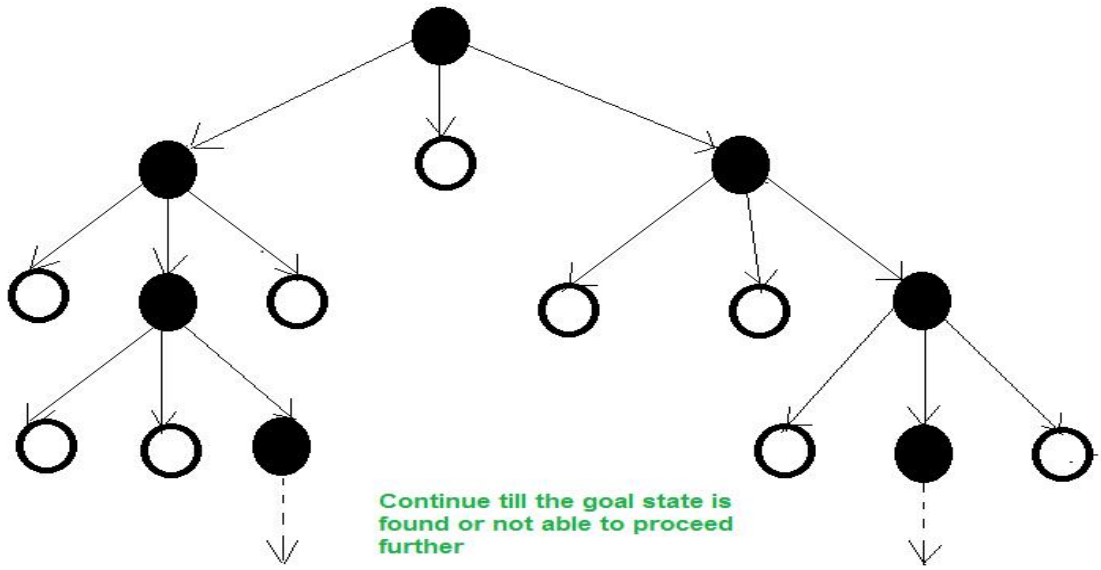
1.1 Parallel Beam Search

Beam search je algoritam pretrage koji na svakom nivou stabla pretrage bira određen broj čvorova iz kojih dalje pokreće pretragu. Skup izabranih čvorova na nekom nivou po nekom kriterijumu se naziva snop (beam) I prenose se u sledeću iteraciju, a ostali ćvorovi se ne razmatraju.

Parallel Beam Search koristi tehnike paralelizacije za ubrzanje izvršavanja osnovnog algoritma beam search radi dobijanja efikasnije implementacije. Usled ekspanzije broja računarskih klastera sa više procesorskih jedinica kao I samih karakteristika hardvera, sve više se radi na paralelizaciji algoritama.[2]

2 Opis algoritma

Glavna ideja Beam Search algoritma je pretraga stabla pretrage nivo po nivo, tako da se na svakom nivou bira β cvorova koji najviše obećavaju dok se ostali čvorovi odsecaju. Postupak se ponavlja dok se ne obiđu svi nivoi stabla pretrage, tj. onoliko koliko imamo čvorova.



Slika 1

U rešavanju problema trgovačkog putnika čvorovi predstavljaju gradove a težine grana su rastojanja između gradova.

Implementacija Parallel Beam Search algoritma se zasniva na osnovnoj implementaciji Beam Search algoritma gde uz pomoć concurrent.futures python biblioteke pravimo niti kojima dodeljujemo poslove u čvorovima.

Pseudokod Parallel Beam Search algoritma prikazan je na slici 1.

```
Input: Set of cities and their distances
Output: Shorthest found path

function parallel_beam_search(distnaces, beam_width, num_processes):
        beam = initial beam with starting node
        for 1 to num_nodes:
                create num_processes thread
                for each thread start search

                wait for all threads
                Sort beam
                Select beam_width nodes for next iteration

        return best beam
```

Slika 1

Pretraga iz čvorova se pokreće u svakoj niti, s obzirom da je broj izabranih čvorova (snop) fiksiran na svakom sloju pretražujemo isti broj čvorova. Sortiranje vršimo po dužini puta u rastućem poretku I prenosimo najbolje u sledecu iteraciju. Kao rezultat vraćamo najkraći put, koji će se nalaziti na početku sortiranog snop.

Usko grlo u paralelizaciji ovog algoritma može biti sortiranje snopa po dužini puta. Da bi se smanjilo vreme izvršavanja sortiranja, umesto celog niza može se sortirati samo β najboljih čvorova.

Određivanje vrednosti β se vrši eksperimentalno. U listu beam se stavlja inicijalni čvor iz kog se kreće pretraga. Obilaze se svi čvorovi u listi beam tako da svaka nit uzima po jedan čvor I računa rastojanje od tog čvora do svih ostalih. Kada sve niti završe sa poslom uzima se β najkraćih puteva I prenose se u sledeću iteraciju. Postupak se ponavlja onoliko puta koliko ima čvorova u grafu.

3 Rezultati

Test primeri nad kojima je primenjen Parallel Beam Search su uzeti iz TSPLIB biblioteke. Takođe rešenja koja su dobijena su upoređena sa rezultatima koji su dobijeni drugim tehnikama iz literature[4]. Korišćeno je 12 niti u PBS.

					Beam width	
att48	Scale	Optimal solution	Best	100	1000	5000
Tabu search	48	33522	34198			
GAs	48	33522	34572			
PSO	48	33522	34759			
ACO	48	33522	34357			
PBS	48	33522	36184	42979	36184	37381

Table 1

					Beam width	
berlin52	Scale	Optimal solution	Best	100	1000	5000
Tabu search	52	7542	7976			
GAs	52	7542	8201			
PSO	52	7542	8197			
ACO	52	7542	7647			
PBS	52	7542	8719	9851	8719	9709

Table 2

					Beam width	
pr76	Scale	Optimal solution	Best	100	1000	5000
Tabu search	76	108159	110941			
GAs	76	108159	115329			
PSO	76	108159	118038			
ACO	76	108159	110517			
PBS	76	108159	145599	151976	145599	150913

Table 3

					Beam width	
eil101	Scale	Optimal solution	Best	100	1000	5000
Tabu search	101	629	667			
GAs	101	629	682			
PSO	101	629	687			
ACO	101	629	649			
PBS	101	629	809	819	809	817

Table 4

Iz priloženih rezultata vidimo da PBS jako varira u odstupanju od optimalnog resenja. Takođe vidimo da je jako tesko odrediti pravu meru za dužinu snopa. Međutim vreme izvrsavanja PBS je vrlo dobro s obzirom da se sečenje čvorova dešava na svakom nivou što dovodi do jako brzog konvergiranja ka rešenju. U sledećim tabelama je dato poređenje u vremenu izvršavanja sa tehnikom Ant System iz literature[5].

att48	Time (s)	Best	Optimal
Ant system	66,425	35250	33522
PBS	2,274	36184	33522

Table 5

berlin52	Time (s)	Best	Optimal
Ant system	72,961	7681	7542
PBS	2,663	8719	7542

Table 6

pr76	Time (s)	Best	Optimal
Ant system	116,907	118693	108159
PBS	6,035	145599	108159

Table 7

st70	Time (s)	Best	Optimal
Ant system	104,988	721	675
PBS	33,403	871	675

Table 8

Rezultati jako variraju u slučaju PBS. Treba napomenuti da nema pravila u odnosu dužine snopa I kvaliteta rešenja, zbog toga je jako teško odrediti najbolju dužinu za konkretnu instancu problema. Međutim, kroz ovaj rad pokazalo se da najbolji odnos efikasnosti I kvaliteta rešenja se dobija za vrednost $\beta = 1000$ na test primerima.

4 Zaključak

U ovom radu rešavan je problem Traveling salesman korišćenjem Parallel Beam Search-a. Rezultati dobijeni izvršavanjem algoritma na test primerima su upoređivani sa rezultatima drugih algoritama koji su preuzeti iz literature.

Iz dobijenih rezulata može se zaključiti da algoritam Beam search nije konkurentan vodećim algortimima koji se bave ovim problemom. Razlog tome je jako brzo konvergiranje ka rešenju, jer se odseca veliki broj mogučnosti koje mogu biti pregledane. Kako bi se povećala efikasnost može se uzeti još neka mera koja će sa nekom određenom verovatnoćom birati čvorove.

Određivanje broja čvorova koji će se birati na svakom nivou pretrage je izazovno, jer ne treba uzimati ni premale a ni prevelike opsege.

Paralelizacija algoritma zavisi od konkretne primene i od progrmaskog jezika, jer treba voditi računa o podacima i nitima. Efikasnije implemantacije se mogu dobiti u jezicima koji podržavaju bolje upravljanje memorijom i nitima, ali ovde je iskorišćen python radi jednostavnosti i ilustrativnosti.

[1] Hoffman, Karla L., Manfred Padberg, and Giovanni Rinaldi. "Traveling salesman problem." Encyclopedia of operations research and management science 1 (2013): 1573-15
[2] Nikolaus Frohner, Jan Gmys, Nouredine Melab, Günther R. Raidl, El-Ghazali Talbi. Parallel Beam Search for Combinatorial Optimization. SoCS 2022 - 15th International Symposium on Combinatorial Search, Jul 2022, Vienne, Austria. ff10.1609/socs.v15i1.21783ff. ffhal-03689638f
[3] Parallel Beam Search for Functionality Partial Matching, Yanran Guan, School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6
[4] Deng, Wu & Chen, Rong & He, Bing & Liu, Yaqing & Yin, Lifeng & Guo, Jinghuan. (2012). A novel two-stage hybrid swarm intelligence optimization algorithm and application. Soft Computing. 16. 10.1007/s00500-012-0855-z.
[5] Experimental Analysis of Ant System on Travelling Salesman Problem Dataset TSPLIB, Kalaipriyan Thirugnanasambandam, Raghav.R.S, Saravanan.D, Prabu.U, Rajeswari.M