Optimizacija - Minimum K-VERTEX CONNECTED SUBGRAPH

Stefan Nešković
Dimitrije Marković

Matematički fakultet, Univerzitet u Beogradu

26. januar 2024.

Sadržaj

1 Uvod

U ovom radu bavićemo se problemom pronalaženja minimalnog k-vezanog podgrafa, koji ima značajnu primenu u teoriji grafova i optimizaciji mreža.

Variable Neighborhood Search (VNS) je metaheuristički pristup za rešavanje problema optimizacije koji se oslanja na sistematsku promenu okoline u potrazi za boljim rešenjem. U radu ćemo detaljno opisati kako VNS može biti primenjen na problem minimalnog k-vezanog podgrafa.

Dalje, prikazaćemo implementaciju ključnih delova VNS algoritma, uključujući funkciju za 'shaking', lokalnu pretragu i funkciju za ocenu kvaliteta rešenja(fitness funkciju), kao i rezultate dobijene korišćenjem VNS algoritma u poređenju sa drugim pristupima, kao što su algoritam grube sile i naivni algoritam.

2 Opis problema

U ovoj sekciji biće detaljno opisan problem minimalnog k-vezanog podgrafa. Cilj je pronaći podgraf koji je k-vezan, što znači da podgraf ostaje povezan čak i nakon uklanjanja do $k-1$ čvora(bilo kojih). Problem možemo formalizovati na sledeći način:

- **Instanca:** Graf $G = (V, E)$, gde je k konstanta, $k \geq 2$.
- **Rešenje:** Podgraf $G' = (V', E')$ takav da je G' k-vezan.
- **Mera:** Kardinalnost podgrafa, tj. $|E'|$.

Osnovni zahtev je da svaki podgraf koji zadovoljava uslove mora imati najmanje k disjunktnih puteva između bilo koja dva čvora, osiguravajući povezanost čak i nakon potencijalnog uklanjanja čvorova.

2.1 Variable Neighborhood Search (VNS)

Variable Neighborhood Search (VNS) je pristup koji koristi koncept promenljivih okolina za pretragu prostora rešenja. Ključni koraci VNS algoritma uključuju:

- **Shaking:** Faza u kojoj se trenutno rešenje sistematski modifikuje kako bi se izašlo iz lokalnog optimuma.
- **Lokalna pretraga:** Nakon faze 'shaking', primenjuje se lokalna pretraga za poboljšanje rešenja.
- **Fitness funkcija:** Koristi se za ocenu kvaliteta rešenja, obično se bazira na meri definisanoj za problem.

Na slici ispod mozete videti pseudokod VNS algoritma, a u daljem tekstu će biti diskutovane specifičnosti njegove implementacije za problem minimalnog kvezanog podgrafa.

Input: *a set of neighbourhood structures* $N_l, l = 1,2, \dots, l_{max}$
$S = Initial\ solution\ ()$;
Repeat
    $l = 1$;
    While $(l \leq l_{max})$
    {
        $S' = Shaking\ (S, N_l)$
        $S'^{*} = local\ search\ (S')$
        if $f(S'^{*}) < f(S)$
            $S \leftarrow S'^{*}$
            $l = 1$;
        else
            $l = l + 1$;
    }
Until *Stopping criterion is met*;
Report: *The obtained solution with the lowest* $f(S)$

Slika 1: Pseudokod VNS algoritma

3 Implementacija algoritama

3.1 Algoritam grube sile i Greedy algoritam

Algoritam grube sile je implementiran tako da iterativno proverava svaki mogući podskup grana grafa, krenuvši od manjih podskupova, koristeći funkciju 'is_k_connected' koja utvrđuje da li je podgraf k-vezan. Ovaj pristup obezbeđuje da ne moramo proveravati sve podskupove jer ako nadjemo k-vezan podgraf sa N grana, ne moramo proveravati podskupove sa više grana. Algoritam je vremenski zahtevan zbog eksponencijalne složenosti. Primera radi, već za graf od 60 grana, algoritam nije uspeo naći rešenje za manje od 20 minuta, a za graf od 240 grana rešenje nije nadjeno za 4 sata.

Greedy algoritam poziva lokalnu pretragu nad početnim grafom 50 puta kako bi brzo došao do rešenja. Ovaj pohlepni pristup omogućava algoritmu da brzo konvergira (što se može i videti na osnovu rezultata), iako možda neće uvek naći globalno optimalno rešenje algoritam daje solidna rešenja.

3.2 Variable Neighborhood Search (VNS)

VNS je detaljno implementiran sa više komponenti koje zajedno unapređuju proces optimizacije.

Shaking faza je dizajnirana da dodaje ili uklanja grane, omogućavajući algoritmu da istraži različite regione prostora pretrage i potencijalno izbegne lokalne optimume. To je urađeno na dva načina tako da se održi k-vezanost, dok se istovremeno pokušava smanjiti ukupan broj grana. Prvi shaking temelji se na nasumičnom odabiru između dodavanja i uklanjanja grana, pri čemu se te odluke donose na temelju značaja grana koje povezuju čvorove s najvećim zbirom stepena. Drugi shaking temelji se na nasumičnom dodavanju grana koje su izbačene iz početnog grafa i nasumičnom uklanjanju par grana koje su prethodno bile u našem lokalnom rešenju. Ovo može pomoći u "otvaranju"novih puteva i potencijalno vodi do drugačije strukture podgrafa nakon ponovne lokalne pretrage

Lokalna pretraga koristi nasumičan odabir grana za izbacivanje (jedini uslov je da susedni čvorovi koje spaja grana moraju da imaju stepen veći ili jednak od K i nakon njenog uklanjanja graf ostaje povezan), što znatno smanjuje šanse da će nam stalno vraćati isti lokalni optimum. Pretraga se pokazala brzom iz razloga

sto se kompleksna funkcija ‘is_k_connected‘ zove samo jednom, za proveru da li smo dobili rešenje koje je k povezano. Da smo, na primer, funkciju ‘is_k_connected‘ zvali nakon uklanjanja svake grane pretraga bi bila znatno sporija, verovatno i neupotrebljiva.

Fitness funkcija direktno meri kvalitet podgrafa u kontekstu problema, ocenjujući rešenje na osnovu broja njegovih grana, što je u skladu sa ciljem pronalaženja minimalnog k-vezanog podgrafa.

Ove komponente reflektuju sofisticiran pristup rešavanju problema minimalnog k-vezanog podgrafa. Kroz inteligentnu integraciju različitih faza VNS-a, implementacija demonstrira kako se standardni algoritam može prilagoditi za rešavanje specifičnih i zahtevnih problema u teoriji grafova.

3.3 Skup grafova za testiranje

Svi grafovi za testiranje su generisani uz pomoc python biblioteke za grafove, networkx. Grafovi su regularni i ispunjavaju uslov k-povezanosti. Razlog zašto su grafovi k-povezani je taj što početni graf mora ispunjavati uslov k-povezanosti da bi uopšte mogli da trazimo njegov k-povezan podgraf.

4 Rezultati

U ovoj sekciji predstavljamo rezultate testiranja algoritama grube sile, pohlepnog algoritma i VNS-a na setu grafova. Za VNS algoritam, vreme izvršavanja je unapred postavljeno na 20 sekundi za svaki graf.

Tabela 1: Rezultati

Graf	V	E	Brute Force	Greedy		VNS	
			Rezultat	Rezultat	Vreme (s)	VNS1	VNS2
1	10	30	20	20	0.146	20	20
2	20	60	-	40	0.259	40	40
3	30	90	-	60	0.498	60	60
4	40	120	-	81	0.929	80	81
5	50	150	-	101	1.159	100	100
6	60	180	-	121	1.536	120	121
7	70	210	-	142	2.067	141	142
8	80	240	-	163	2.589	162	161
9	90	270	-	183	3.069	182	182
10	100	300	-	204	3.661	203	203

Rezultati prikazani u tabeli iznad ukazuju na različite performanse algoritama u pogledu kvaliteta rešenja i vremena izvršavanja. Primetno je da algoritam grube sile za male grafove pronalazi optimalno rešenje(ali i greedy i VNS takodje pronalaze), ali njegova upotreba postaje nepraktična već za grafove od 60 grana zbog eksponencijalne složenosti(Osim za prvi graf, ni za jedan od preostalih grafova algoritam grube sile nije uspeo da nadje resenje za ispod 20min, za veće grafove smo

čekali i po par sati ali nije nadjeno rešenje). Pohlepni algoritam i VNS pokazuju bolje vreme izvršavanja uz prihvatljiv kvalitet rešenja (za male instance problema nalaze optimalna rešenja), što ih čini primenljivim.

5 Zaključak

Analizom rezultata prikazanih u prethodnoj sekciji možemo uočiti nekoliko ključnih aspekata u performansama algoritama koji su testirani. Algoritam grube sile pokazao je da je njegova upotrebljivost ograničena na grafove manje veličine zbog visokog vremenskog zahteva. Ovo je očekivano, s obzirom na eksponencijalnu složenost koju ovaj metod ima.

S druge strane, pohlepni algoritam i VNS su se pokazali kao efikasni u smislu vremena izvršavanja, ali i upotrebljivim jer daju rešenja približna optimalnom, što ih čini pogodnim za primenu u situacijama koje zahtevaju brza rešenja.