VNS metoda za nalaženje maksimalne klike

Todor Todorović 241/2019, Nevena Radulović 407/2016

Opis problema i matematička definicija

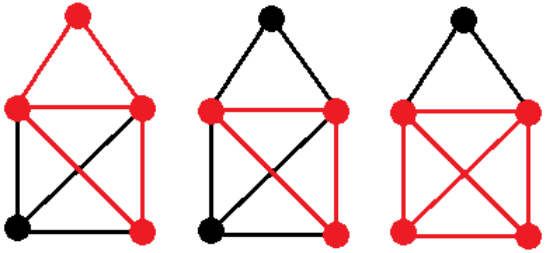
- Dat je neusmeren graf $G = (V, E)$

Opis problema i matematička definicija

- Dat je neusmeren graf $G = (V, E)$
- Tražimo maksimalnu kliku C

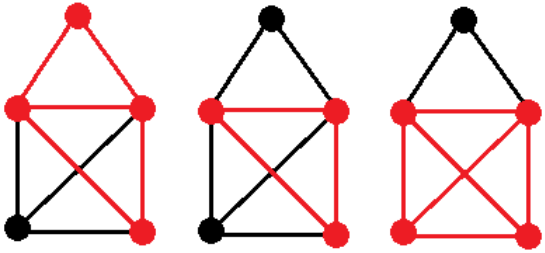
Opis problema i matematička definicija

- Dat je neusmeren graf $G = (V, E)$
- Tražimo maksimalnu kliku C
- Klika je skup čvorova u kome je svaki par čvorova spojen granom
- Za kliku C kažemo da je maksimalna, ukoliko je njena dimenzija veća od dimenzije svih drugih klika



Opis problema i matematička definicija

- Dat je neusmeren graf $G = (V, E)$
- Tražimo maksimalnu kliku C
- Klika je skup čvorova u kome je svaki par čvorova spojen granom
- Za kliku C kažemo da je maksimalna, ukoliko je njena dimenzija veća od dimenzije svih drugih klika



- Ovaj problem je NP-težak!

Matematička formulacija

- Koristimo binarne promenljive:

$$x_i = \begin{cases} 1, & \text{ako čvor } i \in C, \\ 0, & \text{inače.} \end{cases}$$

Matematička formulacija

- Koristimo binarne promenljive:

$$x_i = \begin{cases} 1, & \text{ako čvor } i \in C, \\ 0, & \text{inače.} \end{cases}$$

-

$$\max \sum_{i=1}^{n} x_i \tag{1}$$

$$x_i + x_j \leq 1, \quad (i,j) \in E^C \tag{2}$$

$$x_i \in \{0,1\}, \quad i = 1,2,...,n. \tag{3}$$

Matematička formulacija

- Koristimo binarne promenljive:

$$x_i = \begin{cases} 1, & \text{ako čvor } i \in C, \\ 0, & \text{inače.} \end{cases}$$

-

$$\max \sum_{i=1}^{n} x_i \tag{1}$$

$$x_i + x_j \leq 1, \quad (i,j) \in E^C \tag{2}$$

$$x_i \in \{0,1\}, \quad i = 1,2,...,n. \tag{3}$$

- (1) -veličina klike, (2)-obezbedjuju da svaki par čvorova skupa C zaista jeste susedan

Bruteforce algoritam

- Rekurzivni algortiam
- poziv: početni čvor $startNode$=0, i početnu veličinu klike $size$=1

Require: Matrica susedstva grafa G, polazni čvor, veličina trenutne maksimalne klike
Ensure: Veličina maksimalne klike grafa G

```
n = broj čvorova grafa G
maxClique = 0
for node = startNode + 1 to n do
    clique[size] = node
    if isClique(clique) then
        maxClique = max(maxClique, size)
        maxClique = max(maxClique, brute_force(node, size + 1))
    end if
end for
return maxClique
```

Bruteforce algoritam (2)

- Pomoćna funkcija isClique

Require: Skup čvorova $clique$
Ensure: True ako je skup $clique$ klika, False u suprotnom
  n = broj čvorova u clique
  for $i = 1$ to n **do**
    for $j = i + 1$ to n **do**
      if ne postoji grana izmedju čvorova $clique[i]$ i $clique[j]$ **then**
        return False
      end if
    end for
  end for
  return True

VNS metoda

- Metaheuristička metoda, predložili Mladenović i Hansen

VNS metoda

- Metaheuristička metoda, predložili Mladenović i Hansen
- Ideja: promena okoline u kojoj se vrši lokalna pretraga

VNS metoda

- Metaheuristička metoda, predložili Mladenović i Hansen
- Ideja: promena okoline u kojoj se vrši lokalna pretraga
- Lokalna pretraga poprvlja tekuće rešenje

VNS metoda

- Metaheuristička metoda, predložili Mladenović i Hansen
- Ideja: promena okoline u kojoj se vrši lokalna pretraga
- Lokalna pretraga poprvlja tekuće rešenje
- BVNS: Razmrdavanje-dodatni korak za izbegavanje lokalnih optimuma
- Nasumično se bira rešenje u okolini trenutnog optimuma od koga se započinje lokalna pretraga

VNS za MCP

- Dopustivo rešenje predstavljeno je skupom prirodnih brojeva koji predstavljaju čvorove koji čine kliku

VNS za MCP

- Dopustivo rešenje predstavljeno je skupom prirodnih brojeva koji predstavljaju čvorove koji čine kliku
- Početno rešenje sadrži jedan nasumično odabrani čvor $v \in V$

VNS za MCP

- Dopustivo rešenje predstavljeno je skupom prirodnih brojeva koji predstavljaju čvorove koji čine kliku
- Početno rešenje sadrži jedan nasumično odabrani čvor $v \in V$
- Koliko trenutno rešenje odstupa od dopustivog?

$$fit(S) = \begin{cases} \frac{n(n-1)}{2} - m, & \text{ako } \frac{n(n-1)}{2} - m > 0 \\ \frac{1}{n}, & \text{ako } \frac{n(n-1)}{2} - m = 0. \end{cases}$$

VNS za MCP

- Dopustivo rešenje predstavljeno je skupom prirodnih brojeva koji predstavljaju čvorove koji čine kliku
- Početno rešenje sadrži jedan nasumično odabrani čvor $v \in V$
- Koliko trenutno rešenje odstupa od dopustivog?

$$fit(S) = \begin{cases} \frac{n(n-1)}{2} - m, & \text{ako } \frac{n(n-1)}{2} - m > 0 \\ \frac{1}{n}, & \text{ako } \frac{n(n-1)}{2} - m = 0. \end{cases}$$

- Cilj je usmeriti pretragu ka rešenju sa što manjom vrednosti funkcije prilagodjenosti

Razmrdavanje

Biramo rešenje S_{new} u k-toj okolini tekućeg rešenja S na sledeći način:

Require: Skup S veličine n, Broj k
Ensure: Skup S_{new} veličine $n+1$
$S_{new} = k\_swap(S, S^c)$
$S_{new} = S_{new} + 1\_add(S^c_{new})$

S_{new} se koristi kao početno rešenje za lokalnu pretragu.

Lokalna pretraga

Pretražuje se okolina N_1 tekućeg rešenja S i koristi best improvement strategija

Inicijalizacija: $best\_fit = fitenss(S), improvement = 1$
while $improvement$ **do**
  $improvement = 0$
  for $x \in S$ **do**
    for $y \in S^C$ **do**
      $S_{new} = (S \cup y) \setminus x$
      $new\_fit = fitness(S_{new})$
      if $new\_fit < best\_fit$ **then**
        $best\_fit = new\_fit$
        $S_{best} = S_{new}$
        $improvement = 1$
      end if
    end for
  end for
end while

VNS za MCP

Inicijalizacija: generisati inicijalno rešenje S, definisati okoline $N_1, N_2, ...N_{k_{max}}$ za fazu razmrdavanja, kao i vrednost parametra k_{max}. Definisati okolinu za fazu lokalne pretrage. Postaviti vrednost parametra $iter$ na 1.

$k = 1$

while $iter < iter\_max$ **do**

  $S_{new} = shaking(S, k)$, $iter = iter + 1$

  $S'' = local\_search(S_{new})$

  if $fitness(S'') \leq fitness(S)$ **then**

    $S = S'', k = 1$

  else

    $k = k + 1$

    if $k > n/2$ **then**

      $k = 1$

    end if

  end if

end while

Eksperimentalni rezultati

- Algoritmi su implementirani u programskom jeziku C++, a testiranje je radjeno na računaru sa intel procesorom I5-3320M, 4 jezgra, taktom 3.3ghz, ram memorijom od 16GB, pod operativnim sistemom Ubuntu.
- Instance su preuzete sa `https://networkrepository.com/dimacs.php`.

Eksperimentalni rezultati

- Algoritmi su implementirani u programskom jeziku C++, a testiranje je radjeno na računaru sa intel procesorom I5-3320M, 4 jezgra, taktom 3.3ghz, ram memorijom od 16GB, pod operativnim sistemom Ubuntu.
- Instance su preuzete sa `https://networkrepository.com/dimacs.php`.
- Bruteforce ni nakon 10 sati nije uspeo da pronadje tražene klike, pa je algoritam prekinut

Eksperimentalni rezultati

- Algoritmi su implementirani u programskom jeziku C++, a testiranje je radjeno na računaru sa intel procesorom I5-3320M, 4 jezgra, taktom 3.3ghz, ram memorijom od 16GB, pod operativnim sistemom Ubuntu.
- Instance su preuzete sa `https://networkrepository.com/dimacs.php`.
- Bruteforce ni nakon 10 sati nije uspeo da pronadje tražene klike, pa je algoritam prekinut
- VNS je pokretan 5 puta za svaku instancu sa različitim početnim rešenjima.

VNS rezultati

		VNS					VNS HMU	
Instanca	|V|	*best*	*worst*	*average*	t	*iter_max*	|clique|	t
C125_9	125	34	34	34	133.36	200	34	0.02
brock200_2	200	11	11	11	5000	5000	11-12	1.05
gen400_p0_9_55	400	55	55	55	62459	4000	54.8	34.28
hamming8_4	256	16	16	16	369	200	16	0.01
C250_9	250	40	40	40	17404	3000	44	0.22
keller4	171	11	11	11	66	200	11	0.01
gen200_p0_9_44	200	40	38	39.6	10209	3000	44	0.91
brock200_4	200	15	15	15	9036	2000	16-17	6.8
gen200_p0_9_55	200	55	55	55	18500	3000	55	0.24
keller5	776	25	25	25	111218	1000	27	0.38
p-hat300-3	300	36	33	34.6	22291	2000	36	0.07

Table: Poredjenje predložene VNS metode i VNS metode iz literature

Zaključak

- predložen je VNS algoritam za problem nalaženja maksimalne klike u neusmerenom grafu, kao i Bruteforce algoritam

Zaključak

- predložen je VNS algoritam za problem nalaženja maksimalne klike u neusmerenom grafu, kao i Bruteforce algoritam
- Bruteforce algoritam nije uspeo da se izbori sa zadatim instancama, dok su VNS algoritmom dobijena kvalitetna rešenja.

Zaključak

- predložen je VNS algoritam za problem nalaženja maksimalne klike u neusmerenom grafu, kao i Bruteforce algoritam
- Bruteforce algoritam nije uspeo da se izbori sa zadatim instancama, dok su VNS algoritmom dobijena kvalitetna rešenja.
- Izvršeno je poredjenje sa rešenjima iz literature.

Zaključak

- predložen je VNS algoritam za problem nalaženja maksimalne klike u neusmerenom grafu, kao i Bruteforce algoritam
- Bruteforce algoritam nije uspeo da se izbori sa zadatim instancama, dok su VNS algoritmom dobijena kvalitetna rešenja.
- Izvršeno je poredjenje sa rešenjima iz literature.
- VNS metoda se pokazala kao pogodna za rešavanje problema maksimalne klike

Zaključak

- predložen je VNS algoritam za problem nalaženja maksimalne klike u neusmerenom grafu, kao i Bruteforce algoritam
- Bruteforce algoritam nije uspeo da se izbori sa zadatim instancama, dok su VNS algoritmom dobijena kvalitetna rešenja.
- Izvršeno je poredjenje sa rešenjima iz literature.
- VNS metoda se pokazala kao pogodna za rešavanje problema maksimalne klike
- Dalje?

Zaključak

- predložen je VNS algoritam za problem nalaženja maksimalne klike u neusmerenom grafu, kao i Bruteforce algoritam
- Bruteforce algoritam nije uspeo da se izbori sa zadatim instancama, dok su VNS algoritmom dobijena kvalitetna rešenja.
- Izvršeno je poredjenje sa rešenjima iz literature.
- VNS metoda se pokazala kao pogodna za rešavanje problema maksimalne klike
- Dalje?
- Modifikacija (Iskošeni VNS, Redukovani VNS...), hibridizacija sa nekom drugom metodom

Hvala na pažnji!