Fakulta informačních technologií VUT v Brně

Maximální nezávislá množina

Grafové algoritmy – projekt
Téma 12, porovnání

Bc. Ondřej Ondryáš
Bc. Filip Stupka

3. října 2023

1 Terminologie

Nechť $G = (V, E)$ je neorientovaný graf, kde V je množina uzlů (vrcholů) a $E \subseteq V \times V$ je množina hran.

$N(u) = \{v \mid \{u, v\} \in E\}$ označuje množinu sousedných uzlů pro uzel u; obdobně pro $I \subseteq V$ pak $N(I) = \bigcup_{v \in I} N(v)$ označuje množinu sousedných uzlů celé podmnožiny I.

$\delta(u) = |N(u)|$ označuje stupeň uzlu u, tj. počet sousedných uzlů.

Nezávislá množina $I_G \subseteq V$ je podmnožina uzlů, které v G nejsou sousedné (není mezi nimi hrana), tj. $\forall u, v \in I_G : \{u, v\} \notin E$.

2 Maximální nezávislé množiny

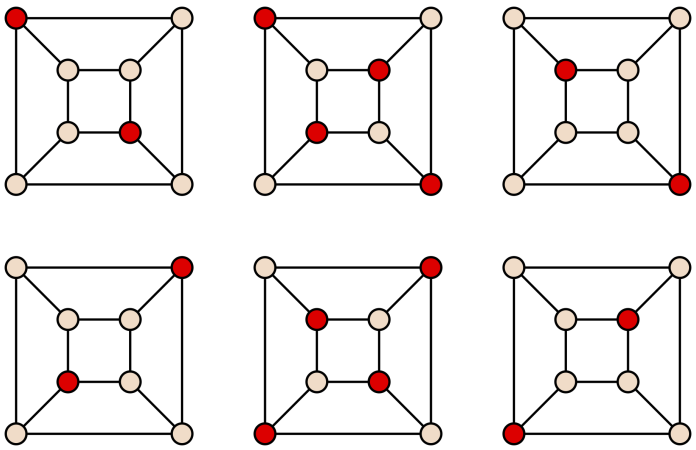
Český termín „maximální nezávislá množina“ může označovat dvě skutečnosti. Prvním možným pohledem je, že maximální nezávislá množina I_G^m je taková nezávislá množina, která není podmnožinou žádné jiné nezávislé množiny. Alternativně, každý uzel grafu buď je prvkem I_G^m, nebo jím není, ale pak je prvkem I_G^m alespoň jeden z jeho sousedních uzlů:

$$\forall u \in V : u \in I_G^m \ \oplus \ N(u) \cap I_G^m \neq \emptyset$$

V angličtině se taková množina označuje jako *maximal independent set*.

Druhým možným pohledem je, že maximální nezávislá množina I_G^n je taková nezávislá množina, která pro graf G obsahuje největší množství uzlů. V angličtině se taková I_G^n označuje jako *maximum independent set*. Pro odlišení od předchozího případu se zde nabízí poněkud nepraktické pojmenování „největší maximální nezávislá množina“. Zřejmě platí, že I_G^n je vždy zároveň I_G^m, ale ne každá I_G^m je zároveň I_G^n. Také zřejmě pro graf může existovat mnoho různých I_G^n i I_G^m.

Rozdíl mezi oběma pohledy znázorňuje obrázek 1. Protože se tato práce zabývá především hledáním množin I_G^n, dále v tomto textu budou pojmem „maximální nezávislá množina“ označeny právě I_G^n, tedy nezávislé množiny s největším množstvím prvků, pokud nebude určeno jinak.



Obrázek 1: Graf s šesti maximálními nezávislými množinami (označeny červenou barvou uzlu) I_G^m, z nichž dvě (v prostředním sloupci) jsou „největší“ I_G^n. Zdroj: [1]

3 Problém maximální nezávislé množiny

Hledání maximálních nezávislých množin patří do kategorie *NP-úplných* problémů. Za předpokladu $NP \neq P$ tedy neexistuje algoritmus s polynomiální časovou složitostí, který by tento problém obecně rozhodoval, a je tedy nutné hledat alternativní přístupy [2].

Jedním z přístupů k řešení NP-úplných problémů jsou aproximační algoritmy, které efektivně (v polynomiálním čase) hledají řešení, která sama o sobě nemusí být optimální, ale jsou prokazatelně v určité vzdálenosti od optimálních řešení [2].

V případě problému maximální nezávislé množiny je možné kvalitu takovýchto aproximačních algoritmů určit jako poměr počtu uzlů skutečné maximální nezávislé množiny v daném grafu a počtu uzlů výsledku aproximačního algoritmu [2].

Další výzkum v oblasti řešení tohoto problému se často zabývá buď optimalizací hledání přesných řešení, nebo hledáním přesných řešení v různě omezených grafech [3].

Praktický význam problému

Problém největší nezávislé množiny má značnou souvislost s problémem hledání největší kliky. Klika je podmnožina vrcholů grafu, které jsou všechny sousedné. Je tedy zřejmé, že klika v grafu $G = (V, E)$ je nezávislou množinou v komplementárním grafu $G' = (V, (V \times V) \setminus E)$. Proto i problém maximální nezávislé množiny je možné převést na problém maximální kliky v komplementárním grafu a naopak. Problém je tak možné použít pro odvozování klasifikace efektivity podobných problémů [4].

Problém maximální nezávislé množiny se objevuje v různých podobách v mnoha oblastech života a vědy. Řeší se například při navrhování genetických systémů (hledání stabilních genetických komponent) [5], v plánování paketů, současném sledování několika objektů [3], pro přidělování zdrojů v sítích [6] nebo ve finančním sektoru [7].

4 Zvolené algoritmy

Pro porovnání bylo zvoleno pět algoritmů založených na hladovém (*greedy*) přístupu a jeden na principu relaxace.

Hladové (*greedy*) aproximační algoritmy obecně pracují tak, že v každém kroku vybírají lokální optima podle určité heuristiky, a snaží se tak dosáhnout globálního optima. Nalezená řešení se mohou od optima výrazně lišit, tyto algoritmy mívají tendence uvíznout v nějakém lokálním optimu, avšak typicky bývají velmi rychlé. Zásadní vlastností těchto algoritmů je, že „nepředpokládají budoucnost“: iterativně redukují problém pouze podle aktuálního, případně minulého stavu, a zpětně svá rozhodnutí nemohou měnit.

Relaxace v tomto případě značí techniku postupného zlepšování odhadu, dokud to je možné.

Hladové algoritmy *Greedy*$_k$ založené na minimalizaci stupně

V případě MIS problému je možné využít pro definování heuristiky hladového algoritmu předpoklad, že do nezávislé množiny budou patřit spíše ty uzly, které mají malý počet sousedů. Hladový algoritmus tedy pracuje v principu takto:

- Vyber uzel $v \in V$ takový, že jeho stupeň $\delta(v)$ je minimální.
- Přidej v do I_G^n.

- Odstraň v z V.
- Odstraň z V všechny uzly v $N(v)$.
- Opakuj, dokud ve V zbývají nějaké uzly.

[8] popisuje úpravu tohoto algoritmu, která spočívá v rozšíření heuristiky z jednoho uzlu s minimálním stupněm na n-prvkové množiny uzlů s minimální *hustotou stupňů*. Hustota stupňů $\delta(I)$ množiny $I \subseteq V$ je definována jako $\delta(I) = \frac{|N(I)|}{|I|}$. Algoritmus *Greedy*$_k$ pro danou konstantu k a vstupní graf $G(V, E)$ pak pracuje takto:

- Vyber podmnožinu $I \subseteq V$ takovou, že $|I| \leq k$ a $\delta(I)$ je minimální.
- Přidej prvky I do I_G^n.
- Odstraň prvky I z V.
- Odstraň z V všechny uzly z $N(I)$.
- Opakuj, dokud ve V zbývají nějaké uzly.

Výstupem je aproximace maximální nezávislé množiny I_G^n. Povšimněme si, že klasický hladový algoritmus odpovídá algoritmu *Greedy*$_1$.

Je zřejmé, že na výsledek bude mít vliv způsob (pořadí) výběru konkrétního uzlu s minimálním stupněm nebo konkrétní n-tice s minimální hustotou stupně.

V této práci proběhly experimenty s variantou *Greedy*$_2$, kterou také popisuje [8]. Autoři zde dále uvádí, že je algoritmus možné naimplementovat s časovou složitostí $\mathcal{O}(\Delta^3 n)$, kde $\Delta = \max_{v \in V} \delta(v)$.

Algoritmus Ramsey a odstraňování podgrafů

Ještě jednodušší hladový algoritmus je možné realizovat rekurzivním způsobem s využitím doplňku množiny sousedů $\overline{N}$:

Algoritmus 1: RandGreedy

Vstup: $G(V, E)$
Výstup: I_G^n

```
1: def RandGreedy(V):
2:     if V == ∅ then return ∅
3:     u = náhodný uzel z V
4:     return {u} ∪ RandGreedy(N̄(u))
```

V tomto případě v podstatě není využita žádná heuristika, algoritmus pouze v každém kroku rekurze splňuje podmínku nezávislé množiny. Je zřejmé, že zde má na výsledek extrémní vliv pořadí výběru uzlů, při výběru špatného uzlu může vést algoritmus na značně neoptimální řešení.

Vylepšení spočívá v úpravě heuristiky tak, že funkce volá sama sebe jak pro okolí, tak pro doplněk okolí, poté vrací větší z výsledků. Algoritmus tedy preferuje řešení, která vedou na větší množiny.

Algoritmus 2: Ramsey

Vstup: $G(V, E)$
Výstup: I_G^n

```
1: def Ramsey(V):
2:     if V == ∅ then return ∅
3:     u = náhodný uzel z V(G)
4:     I_1 = Ramsey(N(u))
5:     I_2 = Ramsey(N̄(u)) ∪{u}
6:     return if |I_1| > |I_2| then I_1 else I_2
```

Algoritmus BMA na bázi Bellman-Fordova algoritmu

Algoritmus BMA (*Bellman-Ford based MIS Algorithm*) představený v [3] problém převádí na variaci problému nejkratší cesty, který ohodnocuje uzly (ne hrany), a to tak, že cena uzlu u představuje počet uzlů, které budou pro další iteraci vyloučeny, pokud se u do cesty přidá. Algoritmus zároveň počítá v každém kroku současně cesty začínající v každém z uzlů. Nalezené cesty pak odpovídají nezávislým množinám, nejdelší cesty odpovídají maximálním nezávislým množinám.

Algoritmus 3 (v příloze A) využívá tři pole množin indexovaná uzlem, která jsou samostatná pro každou iteraci: $Exclude$, $Cost$ a $Previous$. Ve fázi inicializace se do $Exclude$ pro nultou iteraci pro každý uzel vytvoří množina obsahující pouze tento uzel. V dalších iteracích i se prochází všechny dvojice $u \in V : Cost_i[u] < \infty$ a $v \in (V \setminus Exclude_i[u])$, tedy všechny dvojice uzlů, kde jeden uzel už je zařazen v nějaké max. nezávislé množině a druhý uzel pro něj nebyl vyloučen. Pro všechny takové uzly v se pak kontroluje, zda přidání v do MIS povede na nižší cenu v další iteraci. Pokud ano, u je nastaven jako předchůdce v a aktualizují se pole $Exclude_{i+1}[v]$ a $Cost_{i+1}[v]$.

Algoritmus by měl pracovat s časovou složitostí $\mathcal{O}(|V|(|V|^2 - |E|))$ [3]. Z toho také vyplývá, že algoritmus by měl být rychlejší pro grafy s vyšším počtem hran. Teoretická prostorová složitost závisí na zvolené reprezentaci grafu a polí algoritmu, v zásadě by však měla spadat do $\mathcal{O}(n^3)$ [3].

5 Datová sada

Pro experimenty byly zvoleny úlohy ze druhého ročníku soutěže *DIMACS Implementation Challenges*[1]. Tyto grafy byly navrženy pro úlohu hledání maximální kliky, před aplikací algoritmů byl tedy vytvořen jejich komplement. Součástí sady jsou grafy navržené podle různých kritérií, které mají přesah i do jiných aplikací: některé například reprezentují binární slova, která mají určitou Hammingovu vzdálenost [3].

6 Výsledky

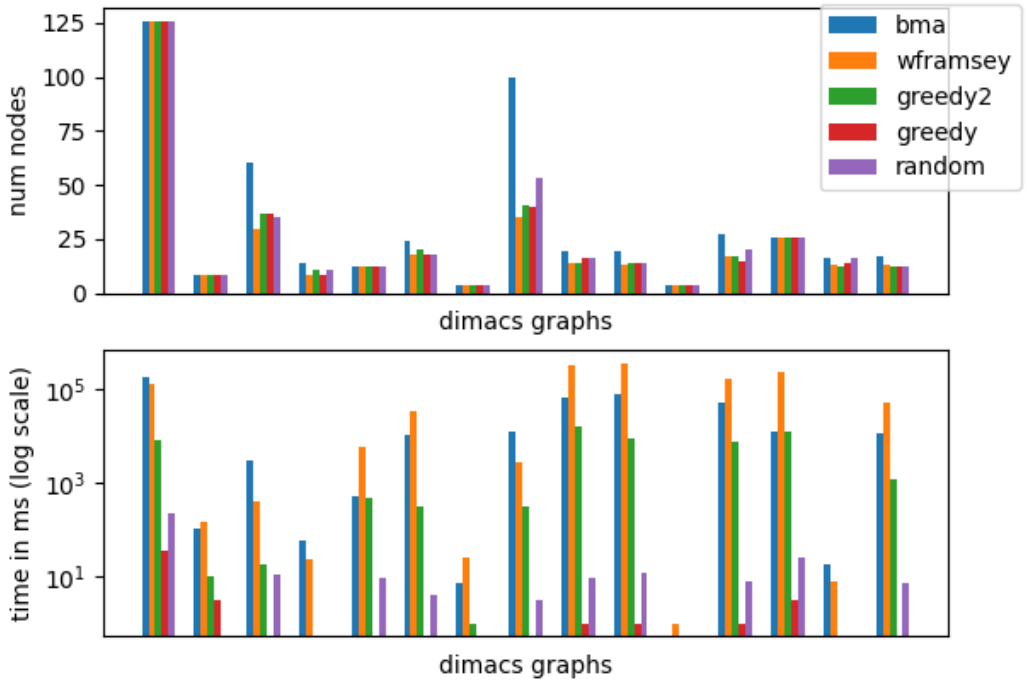
Celkem bylo testováno 5 různých algoritmů, přičemž každý z nich byl spuštěn na všech testovacích grafech ze sady *DIMACS*. U každého algoritmu byl změřen čas a počet uzlů při jeho aproximaci maximální nezávislé množiny a poté byl vypočítán poměr se skutečnou největší nezávislou množinou.

[1] http://archive.dimacs.rutgers.edu/Challenges/

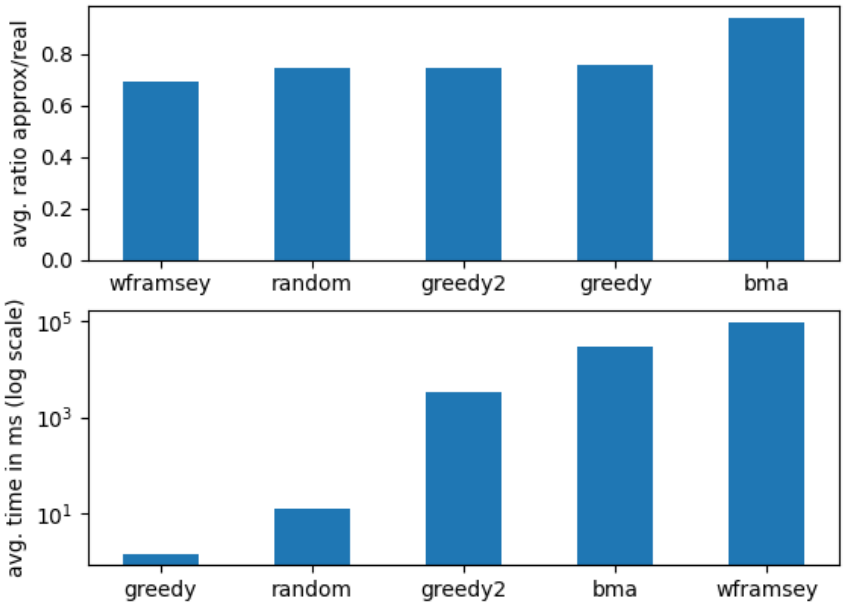
$\overline{G}$	$\lvert V\rvert$	$\lvert E\rvert$	$\alpha(G)$							Čas		
			opt.	BMA	Gr1	Gr2	Ram	Rnd	WFR	BMA	Gr1	Gr2
brock200-1	200	10 332	21	**20**	16	16	16	15	15	2,12	0,000	0,080
brock200-2	200	20 248	12	**11**	7	7	9	7	7	1,27	0,010	0,216
brock200-3	200	15 904	15	**14**	10	10	11	11	10	1,13	0,007	0,237
brock200-4	200	13 822	17	**16**	11	13	12	12	10	2,05	0,000	0,166
brock400-1	400	40 554	27	**24**	19	19	19	19	16	12,74	0,000	1,566
brock400-2	400	40 428	29	**24**	18	20	18	19	16	12,83	0,000	0,788
brock400-3	400	40 638	31	**24**	20	20	18	17	17	12,38	0,000	0,681
brock400-4	400	40 470	33	**24**	18	18	18	18	16	11,41	0,000	0,697
brock800-1	800	224 990	23	**20**	15	16	17	14	14	89,98	0,001	12,997
brock800-2	800	223 668	24	**19**	16	14	16	14	15	70,17	0,001	8,540
brock800-3	800	225 334	25	**19**	14	15	15	15	14	63,50	0,001	21,853
brock800-4	800	224 714	26	**19**	14	14	16	15	14	68,13	0,001	14,170
c-fat200-1	200	36 932	12	**12**	**12**	**12**	**12**	**12**	**12**	0,52	0,000	0,234
c-fat200-2	200	33 530	24	**24**	**24**	**24**	23	**24**	**24**	1,12	0,000	0,219
c-fat200-5	200	23 054	58	**58**	**58**	**58**	**58**	**58**	**58**	2,43	0,001	0,077
c-fat500-1	500	241 082	14	**14**	**14**	**14**	**14**	**12**	**14**	6,06	0,001	7,991
c-fat500-10	500	156 746	126	**126**	**126**	**126**	**126**	**126**	**126**	171,85	0,043	5,319
c-fat500-2	500	231 722	26	**26**	**26**	**26**	**26**	**26**	**26**	15,24	0,003	8,694
c-fat500-5	500	203 618	64	**64**	**64**	**64**	**64**	**64**	**64**	113,31	0,006	6,763
hamming10-2	1024	11 264	512	**512**	**512**	446	323	263	263	308,72	0,006	0,516
hamming10-4	1024	180 224	40	36	29	22	26	19	20	208,48	0,001	20,260
hamming6-2	64	448	32	32	32	22	21	27	17	0,09	0,000	0,000
hamming6-4	64	2688	4	**4**	**4**	**4**	**4**	**4**	**4**	0,01	0,000	0,001
hamming8-2	256	2 304	128	**128**	**128**	104	105	88	66	5,03	0,000	0,023
hamming8-4	256	23 808	16	**16**	13	10	11	11	9	0,88	0,000	0,219
johnson16-2-4	120	3 480	8	**8**	**8**	**8**	**8**	**8**	**8**	0,04	0,000	0,004
johnson32-2-4	496	30 256	16	**16**	**16**	**16**	**16**	**16**	**16**	5,95	0,000	0,602
johnson8-2-4	28	364	4	**4**	**4**	**4**	**4**	**4**	**4**	0,00	0,000	0,000
johnson8-4-4	70	1 190	14	**14**	8	11	10	**14**	8	0,07	0,000	0,000
keller4	171	10 371	11	**11**	7	8	9	8	8	0,68	0,000	0,069
keller5	776	150 196	27	**27**	15	17	18	22	15	69,66	0,001	6,232

MANN-a27	378	1 782	126	**125**	119	117		121	118	8,66	0,000	0,051
MANN-a45	1035	4 995	345	**344**	330	330		333	334	125,56	0,000	0,390
MANN-a9	45	189	16	**16**	14	12	**16**	**16**	15	0,01	0,000	0,000
san1000	1000	499 000	15	**10**	8	8	9	9	8	157,53	0,002	27,550
san200-0-7-1	200	12 140	30	**30**	16	16	15	16	15	2,63	0,000	0,200
san200-0-7-2	200	12 140	18	**16**	12	12	13	12	12	1,98	0,000	0,144
san200-0-9-1	200	4 180	70	**70**	45	45	47	47	45	2,95	0,000	0,031
san200-0-9-2	200	4 180	60	**60**	37	37	35	33	37	3,16	0,000	0,034
san200-0-9-3	200	4 180	44	**44**	30	30	32	29	26	2,45	0,000	0,033
san400-0-5-1	400	80 200	13	**13**	7	7	8	8	7	3,42	0,000	0,595
san400-0-7-1	400	48 280	40	**40**	21	21	21	21	20	19,84	0,006	1,334
san400-0-7-2	400	48 280	30	**30**	15	15	17	16	15	5,90	0,000	0,354
san400-0-7-3	400	48 280	22	**17**	12	12	13	14	14	14,25	0,000	0,864
san400-0-9-1	400	16 360	100	**100**	40	41	54	52	50	21,76	0,000	0,245
sanr200-0-7	200	12 264	18	**18**	14	14	13	14	13	1,73	0,000	0,159
sanr200-0-9	200	4 274	42	**41**	35	35	31	32	30	2,42	0,000	0,023
sanr400-0-5	400	80 032	13	**12**	10	10	10	9	9	9,68	0,000	3,433
sanr400-0-7	400	48 262	21	**20**	16	16	17	16	16	13,64	0,000	0,999

Tabulka 1: Naměřená data pro algoritmy BMA, Greedy, Greedy_2, Ramsey, RandGreedy, WheelFreeRamsey. Sloupec *opt.* obsahuje skutečnou velikost MIS (zdroj: [3]). Tučně vyznačená hodnota značí nejlepší odhad velikosti MIS.



Obrázek 2: Výsledky testování všech 5 algoritmů. Pro přehlednost byly vybrány náhodné grafy ze sady *DIMACS*, přičemž horní graf ukazuje počet uzlů a dolní graf ukazuje časové náklady.



Obrázek 3: Porovnání úspěšnosti jednotlivých grafů. Horní graf obsahuje průměr kvalitativního poměru, a dolní graf ukazuje průměrný čas algoritmů.

Ačkoliv *Wheel Free Ramsey* algoritmus má největší časovou náročnost, z hlediska kvality byl ze všech nejhorší. *Bellman-Ford* algoritmus je naopak ze všech nejúspěšnější, ovšem za vysokou cenovou přirážku.

Hladové algoritmy, které jsou ve svém přístupu k tomuto problému naivní v implementaci, ukazují jako výhodu především rychlost, se zachováním relativně dobrého odhadu. Algoritmus, který generoval maximální nezávislou množinu náhodně, je z hlediska kvality předposlední, nicméně časovou náročnost má srovnatelnou s hladovým algoritmem.

Z naměřených dat je také zřejmé, že každý testovaný algoritmus pracuje lépe v řídkém grafu, neboť čím méně hran, tím se každý algoritmus dopracuje k výsledku rychleji.

Výsledky měření tedy ukazují, že hladové algoritmy jsou nejrychlejší a pokud chceme kvalitnější výsledky, je nutné počítat s vysokým nárůstem časové složitosti.

Reference

1. WIKIPEDIA CONTRIBUTORS.
Maximal independent set — Wikipedia, The Free Encyclopedia [online].
2022. [cit. 2022-12-13]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Maximal_independent_set&oldid=1097443944.
2. BOPPANA, Ravi; HALLDÓRSSON, Magnús M.
Approximating Maximum Independent Sets by Excluding Subgraphs.
BIT Numerical Mathematics. 1992, roč. 32, s. 180–196.
Dostupné z doi: https://doi.org/10.1007/BF01994876.
3. DAHSHAN, Mostafa H.
Maximum Independent Set Approximation Based on Bellman-Ford Algorithm.
Arab J Sci Eng. 2014, roč. 39, č. 10, s. 7003–7011.
Dostupné z doi: https://doi.org/10.1007/s13369-014-1159-7.
4. SKIENA, Steven S. *The algorithm design manual*. Springer, 2012.
isbn 978-1-84800-069-8.
5. HOSSAIN, Ayaan; LOPEZ, Eriberto; HALPER, Sean M.; CETNAR, Daniel P.;
REIS, Alexander C.; STRICKLAND, Devin; KLAVINS, Eric; SALIS, Howard M. Automated design of thousands of nonrepetitive parts for engineering stable genetic systems.
Nature Biotechnology. 2020, roč. 38, č. 12, s. 1466–1475. issn 1546-1696.
Dostupné z doi: 10.1038/s41587-020-0584-2.
6. ZHOU, Jian; WANG, Lusheng; WANG, Weidong; ZHOU, Qingfeng.
Efficient graph-based resource allocation scheme using maximal independent set for randomly-deployed small star networks. *Sensors (Basel, Switzerland)*.
2017, roč. 17, č. 11, s. 2553. issn 1424-8220.
7. BUTENKO, Sergiy; PARDALOS, Panagote M.
Maximum Independent Set and Related Problems, with Applications.
USA: University of Florida, 2003. Dis. pr.
8. LAU, H. Y.; TING, H. F. The Greedier the Better: An Efficient Algorithm for Approximating Maximum Independent Set. *Journal of Combinatorial Optimization*.
2001, roč. 5, s. 411–420. Dostupné z doi: https://doi.org/10.1023/A:1011672624624.

Přílohy

A Algoritmus BMA

Algoritmus 3: BMA

Vstup: $G(V, E)$
Výstup: I_G^n

forall $u \in V$ **do**
    $Exclude[u, 0] = u$
    $Cost[u, 0] = 1$
    $Previous[u, 0] = \emptyset$
    forall $v \in N(u)$ **do**
        $Exclude[u, 0] \leftarrow Exclude[u, 0] \cup \{v\}$
        $Cost[u, 0] \mathrel{+}= 1$
    for $round = 1$ **to** $|V|$ **do**
        $Previous[u, round] = \emptyset$
        $Cost[u, round] = \infty$
for $round = 0$ **to** $|V| - 1$ **do**
    forall $(u, v) \in \{(u, v) \mid u \in V \wedge v \in (V \setminus Exclude[u, round])\}$ **do**
        if $Cost[u, round] < \infty$ **then**
            $newExclude = \{v\}$
            $newCost = Cost[u, round] + 1$
            forall $j \in N(v)$ **do**
                if $j \notin Exclude[u, round]$ **then**
                    $newExclude = newExclude \cup \{j\}$
                    $newCost \mathrel{+}= 1$
            if $Cost[v, round + 1] > newCost$ **then**
                $Previous[v, round + 1] = u$
                $Exclude[v, round + 1] = Exclude[u, round] \cup newExclude$
                $Cost[v, round + 1] = newCost$
    if *v této iteraci nenastala žádná změna v* $Cost$ **then**
        $lastRound = round$
        break
$topU =$ vyber první uzel u, pro který $Cost[u, lastRound] \neq \infty$
while $lastRound \geq 0$ **do**
    $I_G^n = I_G^n \cup \{topU\}$
    $topU = Previous[topU, lastRound]$
    $lastRound \mathrel{-}= 1$
return I_G^n