Пошук максимальних незалежних множин вершин графа для вдосконалення програмних проектів

Ольга Слабоспицька[1], Петро Стецюк[2], Ольга Хом'як[2]

[1]Інститут програмних систем НАН України, Київ
[2]Інститут кібернетики імені В.М. Глушкова НАН України, Київ

Міжнародна науково-практична конференція з програмування
УкрПРОГ'2022
Київ, 12 жовтня 2022

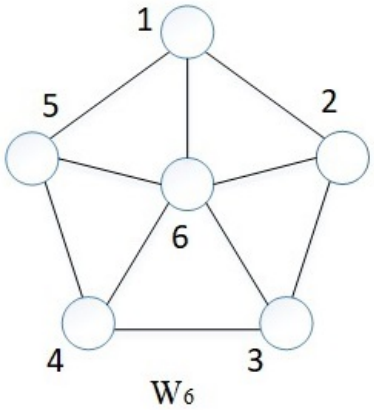
Мета роботи:

вдосконалити процеси керування програмними проектами, використовуючи булеві задачі про максимальну незалежну множину вершин неорієнтованого графа;

уточнити постановки задачі про максимальну незалежну множину і навести ефективні коди програм мовою AMPL для знаходження усіх максимальних незалежних множин і максимальної суми незалежних множин вершин графа.

Що таке $\alpha(G)$?

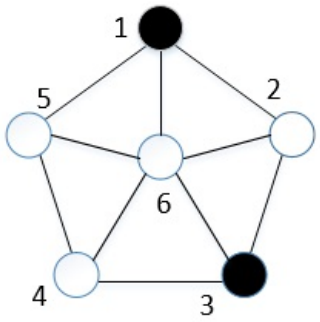
У заданому неорієнтованому графі G без петель потрібно знайти незалежну множину максимального розміру. Її розмір називають числом незалежності і позначають $\alpha(G)$.



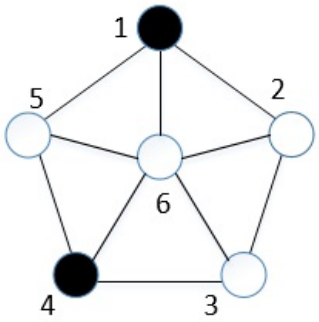
W_6

$E(W_6) = \{(1, 2), (1, 5), (1,6), (2, 3), (2, 6), (3, 4), (3, 6), (4, 5), (4, 6), (5, 6)\}$

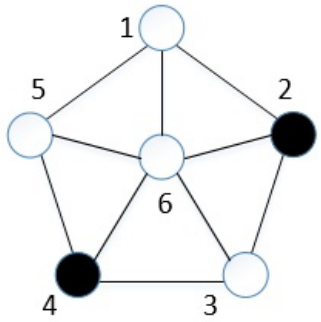
$\Gamma(W_6) :=$
0 1 0 0 1 1
1 0 1 0 0 1
0 1 0 1 0 1
0 0 1 0 1 1
1 0 0 1 0 1
1 1 1 1 1 0



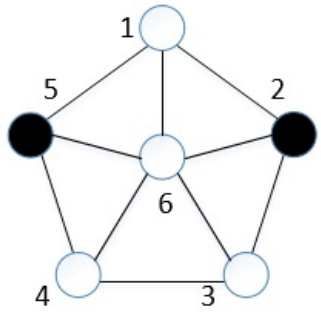
S_1



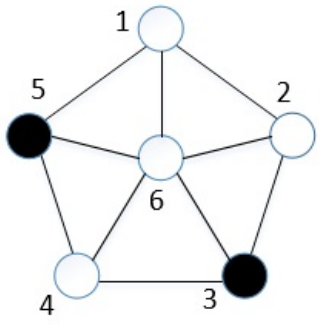
S_2



S_3



S_4



S_5

Зміст

1. Булеві задачі для $\alpha(G)$
2. Булеві задачі для $\alpha_K(G)$
3. $\alpha(G_K)$ та $\alpha_K(G)$ в задачах управління проектами

Перша булева задача

Знайти

$$\alpha(G) = \max \sum_{i \in V(G)} x_i \tag{1a}$$

за обмежень

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

$$x_i \in \{0,1\}, \quad i \in V(G). \tag{3a}$$

Тут булева змінна x_i дорівнює одиниці, якщо вершина i включається в незалежну множину, і нулю – в протилежному випадку.

Друга булева задача

Знайти

$$\alpha(G) = \max \sum_{i \in V(G)} x_i \tag{1b}$$

за обмежень

$$\sum_{j \in \Gamma(i)} x_j \leq d_i(1 - x_i), \quad \forall i \in V(G), \tag{2b}$$

$$x_i \in \{0, 1\}, \quad i \in V(G). \tag{3b}$$

Balansundaram B., Butenko S., Hicks I.V. (2011)

Clique relaxations in social network analysis: the maximum k-plex problem. Operations Research. 2011. Vol. 59, N 1. P. 133–142.

Як знайти усі максимальні незалежні множини?

$$\sum_{i \in V(S_j)} x_i \leq \alpha(G) - 1/2, \quad j = 1, \ldots, m. \tag{4}$$

Лінійні нерівності (4) відсікають уже знайдені множини S_j, $j = 1, \ldots, m$, тому розв'язком задачі (1a)-(3a), (4) або задачі (1b)-(3b), (4) буде нова незалежна множина S_{m+1}.

Якщо розмір множини S_{m+1} дорівнює $\alpha(G)$, то вона буде новою максимальною незалежною множиною, і її можна додати до списку множин S_j, $j = 1, \ldots, m$. Якщо розмір менший ніж $\alpha(G)$, то список множин S_j, $j = 1, \ldots, m$ містить усі максимальні незалежні множини для графа G.

AMPL-код для задачі (1a)-(3a), (4)

```
# Опис графу G для булевої задачі (1a)-(3a), (4)
   set V_G := {1,2,3,4,5,6};  # вершини графа
   set E_G:={(1,2),(1,5),(1,6),(2,3),(2,6),(3,4),
             (3,6),(4,5),(4,6),(5,6)}; # ребра графа
   display V_G,E_G;

# Опис та задання параметрів
   param nCalls:=100; #кількість запусків солвера
   param alphaG1;
   set nS default {}; set S{nS} default {};

# Опис булевої задачі (1a)-(3a), (4)
   var xs{i in 1..card(V_G)} binary;
   maximize alpha_G: sum{i in 1..card(V_G)}xs[i];
   subject to con2 {(i,j) in E_G}: xs[i]+xs[j] <= 1;
   subject to con5 {i in nS}: sum{j in S[i]} xs[j] <= alphaG1;

   option solver gurobi; # вибір солвера

# nCalls разів розв'язуємо задачу (1a)-(3a), (4)
   solve; display alpha_G; let alphaG1:=alpha_G-0.5;
   let nS:={1}; let S[1] := {j in V_G: xs[j]>=0.99};
   for{i in 2..nCalls} {
      solve; display alpha_G;
      if (alpha_G<alphaG1) then break;
      let nS:=nS union {i}; let S[i] := {j in V_G: xs[j]>=0.99};
   }
# друкуємо кількість запусків солвера та отримані максимальні незалежні множини
   display nCalls,S;
```

Зміст

1. Булеві задачі для $\alpha(G)$
2. Булеві задачі для $\alpha_K(G)$
3. $\alpha(G_K)$ та $\alpha_K(G)$ в задачах управління проектами

Перша булева задача

Знайти

$$\alpha_K(G) = \max_{x_{ki}\in\{0,1\}} \sum_{k=1}^{K} \sum_{i\in V(G)} x_{ki} \tag{5a}$$

за обмежень

$$x_{ki} + x_{kj} \leq 1, \quad k = 1, \ldots, K, \quad (i,j) \in E(G), \tag{6a}$$

$$\sum_{k=1}^{K} x_{ki} \leq 1, \quad i \in V(G). \tag{7a}$$

Друга булева задача

Знайти

$$\alpha_K(G) = \max_{x_{ki}\in\{0,1\}} \sum_{k=1}^{K} \sum_{i\in V(G)} x_{ki} \tag{5b}$$

за обмежень

$$\sum_{j\in\Gamma(i)} x_{kj} + \sum_{\substack{t=1,\dots,K \\ t\neq k}} \sum_{j\in V(G)} x_{tj} \leq (d_i + (K-1)|V(G)|)(1-x_{ki}), \tag{6b}$$

$$\forall k \in \{1,\dots,K\}, \forall i \in V(G).$$

Алгоритм Візинга-Плесневича

Задачі (5a) – (7a) та (5b), (6b) можуть бути використані в алгоритмі Візинга-Плесневича для пошуку мінімальної кількості кольорів для розфарбування вершин графа, щоб суміжні вершини були розфарбовані різними кольорами.

Лема 1

$$\alpha_K(G) = \alpha(G \times I_K)$$

Лема 2

$$\chi(G) = \min\{k : \alpha(G \times I_k) = |G(V)|\} = \\ = \min\{k : \alpha_k(G) = |G(V)|\}$$

Опис алгоритму Візинга-Плесневича

Ініціалізація

На ітерації $k = 0$ маємо $\chi_{\min} = 1$, $\chi_{\max} = |V(G)|$.

Ітеративний процес

Нехай на k-й ітерації знайдені $\chi_{\min}$ та $\chi_{\max}$.

1. Якщо $\chi_{\max} - \chi_{\min} < 1.5$, то $\chi(G) = \chi_{\max}$, $itn = k$ і зупиняємося.
2. Обчислимо $K = \lceil (\chi_{\min} + \chi_{\max})/2 \rceil$, де $\lceil \cdot \rceil$ – найближче ціле число. Знайдемо $\alpha = \alpha_K(G)$ або $\alpha = \alpha(G_K)$.
3. Якщо $\alpha = |V(G)|$, то $\chi_{\max} = \alpha$, інакше $\chi_{\min} = \alpha$.
4. Переходимо до $(k+1)$-ї ітерації з новими $\chi_{\min}$ та $\chi_{\max}$.

Зміст

1. Булеві задачі для $\alpha(G)$
2. Булеві задачі для $\alpha_K(G)$
3. $\alpha(G_K)$ та $\alpha_K(G)$ в задачах управління проектами

Задачі управління програмними проектами

- формування прототипних складів згуртованих команд;
- оптимізація розкладу автономного тестування компонентів повторного використання в складі критичної програмної системи;
- формування ядер незалежних команд у критичному програмному проекті.

Приклад 1. Формування згуртованих команд

Нехай 25 фахівців, виконавців портфеля програмних проектів, впорядковані згідно з алфавітним порядком їх прізвищ і занумеровані числами від 1 до 25.

Номери фахівців, які конфліктували в попередніх проектах					
1	21	10	2,6,16,23,25	19	9,13,17,22,24
2	6,10,11,15,22,23,24	11	2,4,14,15,16,23	20	4,5,12,16,18
3	13,17	12	5,20	21	1,4,8,14
4	5,8,11,14,16,20,21	13	3,7,9,17,19	22	2,6,17,19,24
5	4,8,12,20	14	4,11,15,21	23	2,10,11,16
6	2,10,17,22	15	2,11,14	24	2,9,19,22
7	9,13	16	4,10,11,18,20,23	25	10,16,18
8	4,5,21	17	3,6,13,19,22		
9	7,13,19,24	18	16,20,25		

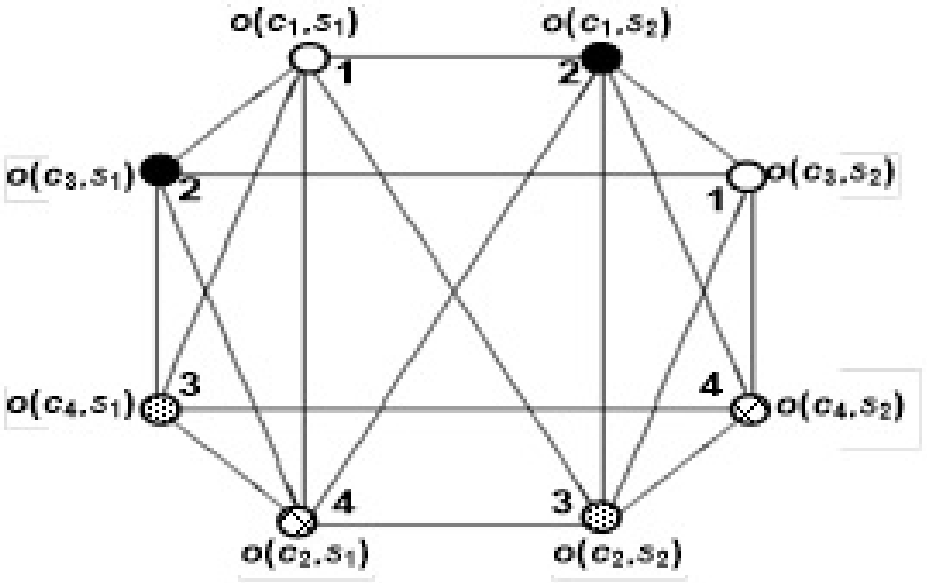
Формування згуртованих команд (слайд 2)

Тут $\alpha(G_{25}) = 10$ та 16 максимальних незалежних множин:

```
set S[1] := 1 3 6 7 8 12 14 18 19 23;
set S[3] := 1 3 4 6 7 12 15 23 24 25;
set S[5] := 1 3 6 7 8 12 14 19 23 25;
set S[7] := 1 3 6 7 8 12 14 23 24 25;
set S[9] := 1 3 6 7 8 15 19 20 23 25;
set S[11] := 1 3 6 7 8 12 15 18 19 23;
set S[13] := 1 3 6 7 8 14 20 23 24 25;
set S[15] := 1 3 4 6 7 12 15 19 23 25;
```

```
set S[2] := 1 3 6 7 8 12 15 23 24 25;
set S[4] := 1 3 6 7 8 12 15 19 23 25;
set S[6] := 1 3 6 7 8 12 14 18 23 24;
set S[8] := 1 3 6 7 8 14 19 20 23 25;
set S[10] := 1 3 4 6 7 12 15 18 23 24;
set S[12] := 1 3 6 7 8 12 15 18 23 24;
set S[14] := 1 3 4 6 7 12 15 18 19 23;
set S[16] := 1 3 6 7 8 15 20 23 24 25;
```

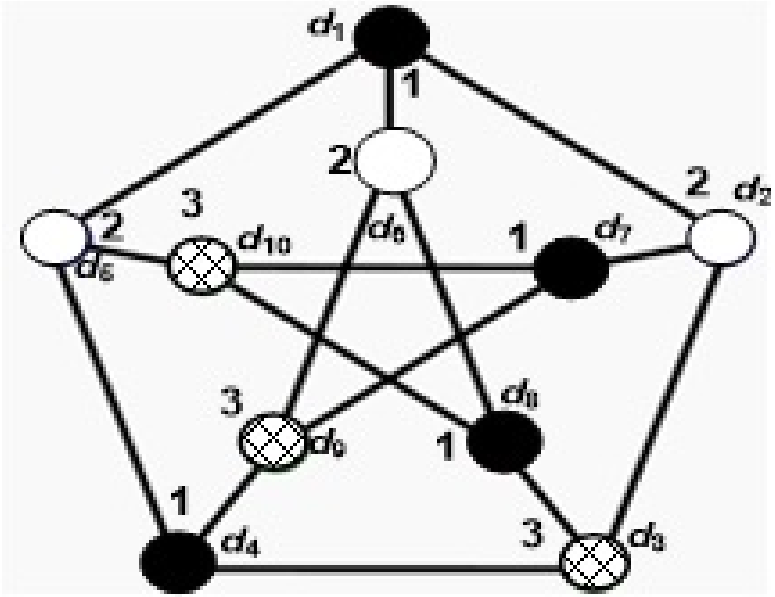
Приклад 2. Оптимізація розкладу тестування



Граф для побудови розкладу тестування, розфарбований фарбами 1-4. Підмножини вершин, розфарбованих одним кольором, відповідають операціям тестування.

Приклад 3. Формування незалежних команд

Потрібно визначити мінімальну кількість та склад ядер згуртованих підкоманд для незалежного розроблення версій модулів критичної програмної системи, члени яких повинні мати досвід успішної спільної роботи, на підставі матриці сумісності фахівців-кандидатів.



Висновки:

Запропоновані постановки задач та AMPL-коди

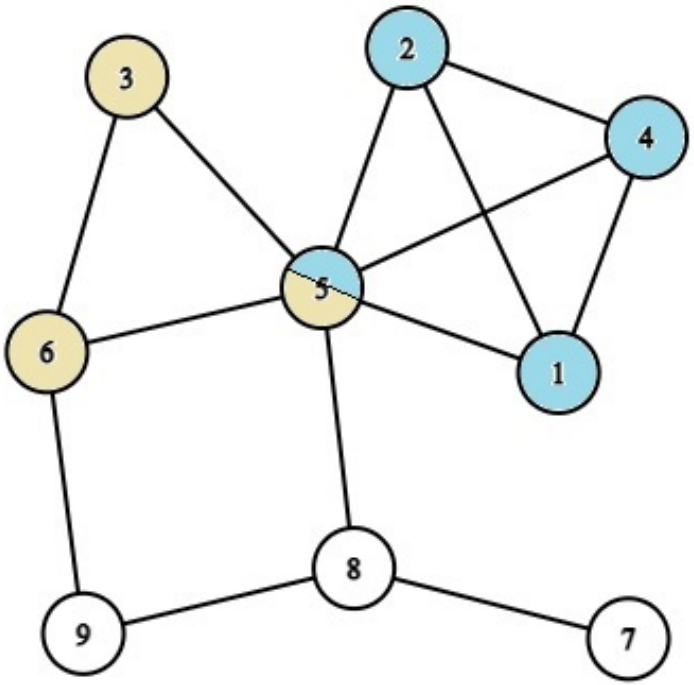
можуть бути використані для автоматизованого отримання розв'язків тих задач керування програмним проектом, розв'язання яких недостатньо регламентовано в руслі технологічно-описового підходу до керування ним.

Досліджені приклади таких задач:

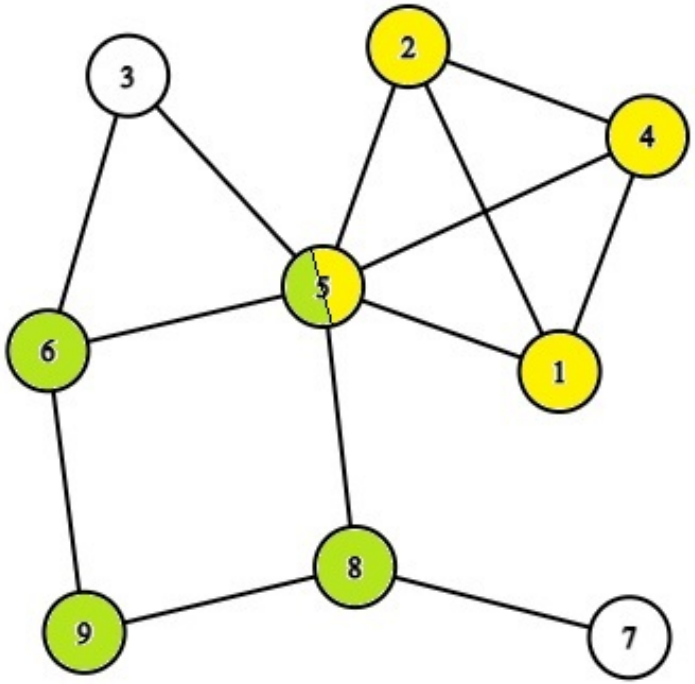
- формування найбільш згуртованої команди програмного проекту;
- оптимізація розкладу тестування компонентів критичної програмної системи;
- формування ядер незалежних підкоманд у критичному програмному проекті.

Напрями подальших досліджень

Наведені алгоритми висвітлюють доцільність узагальнення незалежної множини/кліки до co-k-плексу/k-плексу.



Максимальний 1-плекс (кліка)



Максимальний 2-плекс

Запитання?

ДЯКУЮ ЗА УВАГУ!

e-mail: stetsyukp@gmail.com