УДК 512.97

Кубайчук О.О., Теренчук С.А.,
Єременко Б.М.

ЗАСТОСУВАННЯ ТОПОЛОГІЧНОГО СОРТУВАННЯ ОРІЄНТОВАНИХ АЦИКЛІЧНИХ ГРАФІВ У ПЛАНУВАННІ БУДІВЕЛЬНИХ РОБІТ

Орієнтовані ациклічні графи (directed acyclic graph, скорочено DAG) застосовують для відображення послідовності подій в системах планування і управління розробками (СПУ).

Задачу планування виконання будівельних робіт можна розв'язувати як задачу оптимізації. Однак за такого підходу, мусимо на змінні накласти додаткову умову, а саме, умову їх цілочисленості. В результаті отримуємо задачу цілочисельного лінійного програмування для якої навіть пошук допустимого розв'язку є NP-повною задачею. Оскільки, на сьогодні, жодна NP-повна задача не має поліноміального алгоритму розв'язання, то і для задач цілочисельного лінійного програмування поліноміальні алгоритми також невідомі.

Отже, є сенс розглянути проблему з позиції теорії графів. Нехай, $G=(V,E)$ – орієнтований ациклічний граф. Ребра – роботи, ваги ребер – проміжки часу, необхідні для виконання робіт. Причому, якщо ребро (u,v) входить у вершину v, а ребро (v,w) залишає її, то це означає, що робота (u,v) мусить бути завершеною до роботи (v,w). Шлях по такому орієнтованому ациклічному графу – послідовність робіт, які потрібно виконати в певному порядку, а критичний шлях – найдовший (той, що вимагає найбільше часу) з цих шляхів.

Для визначення критичного шляху застосуємо процедуру знаходження найкоротших шляхів із однієї вершини в орієнтованих ациклічних графах DAG_SHORTEST_PATHS в [1] з такими модифікаціями:

1. в процедурі INITIALIZE_SINGLE_SOURCE значення $+\infty$ змінимо на $-\infty$;
2. в процедурі RELAX знак "більше" змінимо на знак "менше".

Слід відмітити, що ключовим моментом DAG_SHORTEST_PATHS є процедура топологічного сортування орієнтованого ациклічного графа за лінійний час $\Theta(V+E)$, розроблена Кнутом [2]. У свою чергу ця процедура використовує алгоритм пошуку в глибину DFS в [1]. Коректність алгоритму топологічного сортування випливає з леми 22.11 і теореми 22.12 в [1].

Можна запропонувати реалізацію модифікованого алгоритму DAG_SH_PATHS знаходження найдовших шляхів із однієї вершини в

орієнтованих ациклічних графах і розв'язання поставленої задачі в середовищі MATHCAD v.11.

Розглянемо деякий реальний процес підготовки і проведення будівельних робіт , навединих в таблиці:.

Назва операцій (робота)	**Початок**	**Кінець**	**Трив.**	**Подія**	№ Вузла
Розробка документації	1	2	3	Старт	1
Підготовка території	2	3	8	Дозвіл на будівництво	2
Замовлення матеріалів	2	4	5	Замовлено матеріали	3
Підготовка кадрів	3	4	5	Початок будівельних робіт	4
Доставка матеріалів	3	6	6	Закладення фундаменту	5
Закладення фундаменту	4	5	9	Підготовка до будівництва стін	6
Огородження території	5	6	3	Закінчення будівництва стін і основних елементів будинку	7
Будівництво основних елементів будинку	5	7	2	Закінчення зовнішніх робіт	8
Оформлення благоустрою	5	9	5	Готовність до здачі	9
Будівництво стін	6	7	4		
Будівництво підвалу	6	8	7		
Встановлення вікон	7	8	6		
Будівництво покрівлі	7	9	5		
Проведення внутрішніх робіт	8	9	10		

Такий процес можна представити у вигляді орієнтованого зваженого ациклічного графа (Рис.1).

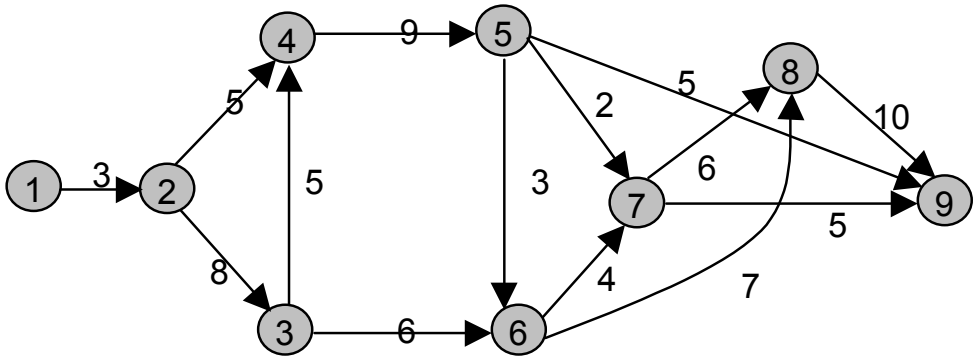

Рис. 1

Знаходження критичного шляху (максимального шляху)

ORIGIN:= 1

1). Задати граф списком суміжностей.

$$T1_1 := 2 \quad T2 := \begin{pmatrix} 3 \\ 4 \end{pmatrix} \quad T3 := \begin{pmatrix} 4 \\ 6 \end{pmatrix} \quad T4_1 := 5 \quad T5 := \begin{pmatrix} 6 \\ 7 \\ 9 \end{pmatrix} \quad T6 := \begin{pmatrix} 7 \\ 8 \end{pmatrix} \quad T7 := \begin{pmatrix} 8 \\ 9 \end{pmatrix} \quad T8_1 := 9 \quad T9_1 := 9$$

$$W1_1 := 3 \quad W2 := \begin{pmatrix} 8 \\ 5 \end{pmatrix} \quad W3 := \begin{pmatrix} 5 \\ 6 \end{pmatrix} \quad W4_1 := 9 \quad W5 := \begin{pmatrix} 3 \\ 2 \\ 5 \end{pmatrix} \quad W6 := \begin{pmatrix} 4 \\ 7 \end{pmatrix} \quad W7 := \begin{pmatrix} 6 \\ 5 \end{pmatrix} \quad W8_1 := 10 \quad W9_1 := 0$$

$$GRAF := \begin{pmatrix} 1 & T1 \\ 2 & T2 \\ 3 & T3 \\ 4 & T4 \\ 5 & T5 \\ 6 & T6 \\ 7 & T7 \\ 8 & T8 \\ 9 & T9 \end{pmatrix} \qquad WGT := \begin{pmatrix} W1 \\ W2 \\ W3 \\ W4 \\ W5 \\ W6 \\ W7 \\ W8 \\ W9 \end{pmatrix}$$

2). Операції зі стеком.

$S\_max := 10$

$STACK\_EMPTY(S) := S_1 = 0$

$STACK\_OVER(S) := S_1 = S\_max$

$INI\_STACK :=$
- $head \leftarrow 0$
- for $i \in 1, S\_max$
  - $S_i \leftarrow 0$
- $\begin{pmatrix} head \\ S \end{pmatrix}$

$POP(S) :=$
- $error("STACK\ UNDERFLOW")\ \text{if}\ STACK\_EMPTY(S)$
- $S_1 \leftarrow S_1 - 1$
- $\begin{bmatrix} (S_2)_{S_1+1} \\ S \end{bmatrix}$

$PUSH(S, x) :=$
- $error("STACK\ OVERFLOW")\ \text{if}\ STACK\_OVER(S)$
- $S_1 \leftarrow S_1 + 1$
- $(S_2)_{S_1} \leftarrow x$
- S

3). Процедури, які використовує процедура пошуку в глибину DFS.

$FIND\_MARK(v, T, B) :=$
- return -1 if $IsScalar(T)$
- for $i \in 1..length(T)$
  - $w \leftarrow T_i$
  - return w if $B_{v,w} = 0$
- -1

$IS\_WHITE(color) := color = "WHITE"$

$fm(i, j) := 0$

$T\_num := rows(GRAF)$

4). Пошук в глибину.

```
DFS(G) := for s ∈ 1.. T_num
            color_s ← "WHITE"
            d_s ← 0
            f_s ← 0
            pi_s ← "NIL"
          M ← matrix(T_num, T_num, fm)
          time ← 0
          S ← INI_STACK
          for k ∈ 1.. T_num
            if IS_WHITE(color_k)
              color_k ← "GRAY"
              d_k ← time ← time + 1
              S ← PUSH(S, k)
              v ← k
              while ¬STACK_EMPTY(S)
                w ← FIND_MARK(v, G_{v,2}, M)
                if w ≠ −1
                  if color_w ≠ "WHITE"
                    M_{v,w} ← "B"  if color_w = "GRAY"
                    M_{v,w} ← "FC"  otherwise
                  otherwise
                    M_{v,w} ← "T"
                    color_w ← "GRAY"
                    d_w ← time ← time + 1
                    pi_w ← v
                    S ← PUSH(S, w)
                    v ← w
                otherwise
                  v ← POP(S)_1
                  S ← POP(S)_2
                  color_v ← "BLACK"
                  f_v ← time ← time + 1
                  v ← pi_v
          augment(G^⟨1⟩, color, d, f, pi, M)
```

5). Процедура ініціалізації оцінок найкоротших шляхів і попередників. Процедура ослаблення ребра.

```
INI_SINGLE_SOURCE(G, s) :=
    for i ∈ 1.. rows(G)
        d_i ← −∞
        pi_i ← "NIL"
    d_s ← 0
    augment(d, pi)
```

```
RELAX(u, v, w, d, pi) :=
    if d_v < d_u + w
        d_v ← d_u + w
        pi_v ← u
    augment(d, pi)
```

6). Знаходження найкоротших шляхів з однієї вершини в орієнтованому ациклічному графі. Попередньо виконується топологічне сортування вершин вхідного графа.

```
TOPOL_SH_PATHS(G, s) :=
    U ← reverse(csort(DFS(G), 4))^⟨1⟩
    A ← INI_SINGLE_SOURCE(G, s)
    d ← A^⟨1⟩
    pi ← A^⟨2⟩
    for i ∈ 1.. length(U)
        u ← U_i
        continue if G_{u,2} = 0
        for j ∈ 1.. length(G_{u,2})
            B ← RELAX[u, (G_{u,2})_j, (WGT_u)_j, d, pi]
            d ← B^⟨1⟩
            pi ← B^⟨2⟩
    augment(pi, G^⟨1⟩, d)
```

7). Формат результату: x_{i1} *– початок,* x_{i2} *– кінець,* x_{i3} *– "вага" шляху із джерела до вершини* x_{i2} *. Якщо* $x_{i3} = \infty$ *, то не існує шляху з джерела до цієї вершини.*

$$\text{TOPOL\_SH\_PATHS}(\text{GRAF}, 1) = \begin{pmatrix} \text{"NIL"} & 1 & 0 \\ 1 & 2 & 3 \\ 2 & 3 & 11 \\ 3 & 4 & 16 \\ 4 & 5 & 25 \\ 5 & 6 & 28 \\ 6 & 7 & 32 \\ 7 & 8 & 38 \\ 8 & 9 & 48 \end{pmatrix}$$

Розрахований програмою критичний шлях показано на Рис.2.

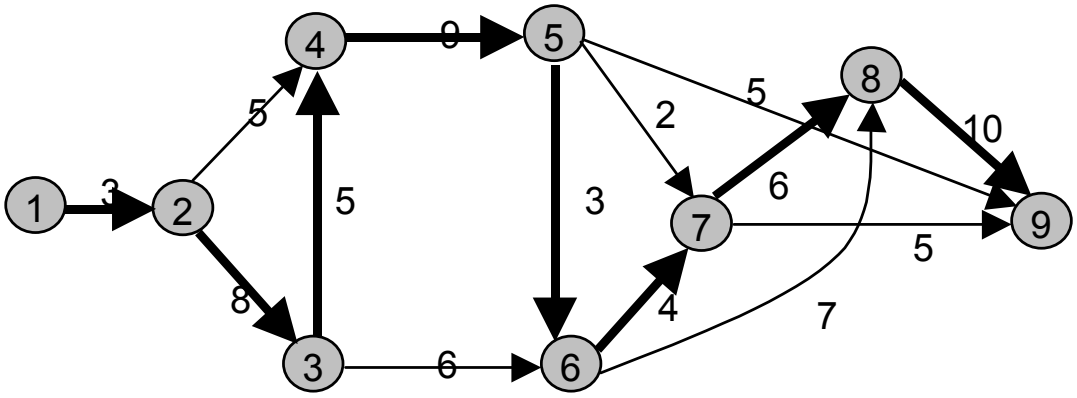

Рис. 2 Критичний шлях

Окремо слід відмітити, що процедура TOPOL_SH_PATHS тільки частково використовує результати процедури пошуку в глибину DFS. Програма DFS модифікована таким чином, що проводить класифікацію ребер вихідного графа, а саме, виділяє ребра: а) ребра дерева пошуку в глибину; б) обернені ребра; в) прямі, або перехресні ребра. Цю (надлишкову з точки зору знаходження критичного шляху) інформацію можна використати для розв'язання інших задач.

Література

1. *Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн.* Алгоритмы. Построение и анализ: Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 1290 с.
2. *Д. Кнут.* Искусство программирования, Т.1. Основные алгоритмы, 3-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2000.- 723 с.

Анотація

Розроблено ефективну програму в середовищі MATHCAD, яка вирішує проблему визначення критичного шляху у плануванні будівельних робіт методами теорії графів.

Аннотация

Разработана эффективная программа в среде MATHCAD, которая решает проблему определения критического пути в планировании строительных работ методами теории графов.