Технологія і організація виробництва

УДК 512.97

О.О. Кубайчук, к.ф.- м.н. доцент Європейського університету,
С.А .Теренчук, к.ф.- м.н. доцент КНУБА,
Б.М. Єременко, студент КНУБА

ОПТИМІЗАЦІЯ УПРАВЛІННЯ МЕТОДОМ ВИДІЛЕННЯ СИЛЬНО ЗВ'ЯЗНИХ КОМПОНЕНТІВ НА ГРАФАХ

У статті засобами MATHCAD вирішена проблема знаходження сильно зв'язних компонентів для орієнтованого графа.

В статье средствами MATHCAD решена проблема отыскания сильно связных компонент ориентированного графа.

Постановка задачі. Під оптимальним управлінням підприємством в даній роботі будемо розуміти впорядкування його структурних підрозділів за обраним критерієм.

Розглянемо дану проблему на прикладі будівельної компанії, в структурі якої є окремі фірми, зайняті виробництвом, причому, вхідним продуктом для одних є вихідний продукт інших.

Таку складну систему природно розглянути в теоретико-графовій постановці. Нехай $G=(V,E)$ – орієнтований граф, вершини якого – це фірми, а ребра відображають зв'язки між фірмами. Оскільки граф орієнтований, то порядок запису вершин ребра (u,v) є суттєвим. В даному випадку це означає, що фірма v використовує продукцію фірми u.

Ефективний менеджмент компанії полягає, в тому числі, у впорядкуванні роботи фірм. Ідеальною є ситуація, коли вдається так організувати цей процес, що фірми у виробничому циклі розташовуються в лінійному порядку – вишиковують вздовж горизонтальної лінії так, що всі ребра направлені зліва направо (або всі справа наліво). Іншими словами, фірми сортують, точніше, топологічно сортують.

Граф, який моделював проблему в [1], був ациклічним. Для орієнтованих ациклічних графів існують алгоритми топологічного сортування за лінійний час $\Theta(V+E)$ [2]. Зокрема, для розв'язання задачі знаходження критичного шляху, сформульованої в [1] побудовано процедуру TOPOL_SH_PATHS в середовищі MATHCAD v.11, яка використовує алгоритм пошуку в глибину DFS [3]. Граф, що описує дану задачу організації управління, не є ациклічним в загальному випадку. Отже, топологічне сортування не можна застосувати.

1. Алгоритм розв'язання.

1. Виділити фірми, між якими склалися тісні цикли виробництва-використання продукції;
2. Створити керівний орган для управління кожною компонентою;
3. Здійснювати координацію дій між компонентами (через уповноважені органи).

Крок перший, з точки зору теорії графів, означає розклад графа G на сильно зв'язні компоненти (СЗК).

Крок другий лежить поза компетенцією теорії графів.

Крок третій означає скористатися ключовою властивістю графа компонентів, а саме: граф компонентів є орієнтованим ациклічним графом, що випливає з леми 22.13 в [3].

Відомо, що сильно зв'язний компонент орієнтованого графа $G=(V,E)$ – це максимальна множина вершин $C\subseteq V$, така що для кожної пари вершин u і v з C: u і v досяжні одна з одної. Таржан в [4] розробив алгоритм пошуку сильно зв'язних компонент за лінійний час. Ми скористаємося алгоритмом (варіант Ахо, Хопкрофт і Ульман [4]), який

використовує транспонування вихідного графа G (транспонований граф $G^T = (V, E^T)$, складається з ребер G, взятих у протилежному порядку). Цікаво відмітити, що графи G і G^T мають одні й ті ж сильно зв'язані компоненти. Граф компонентів утворюється з вихідного графа G, якщо стиснути в точку всі ребра між суміжними вершинами в кожному сильно зв'язному компоненті.

2. Розклад на сильно зв'язні компоненти. Для знаходження СЗК скористаємось алгоритмом STRONGLY_CONNECTED_COMPONENTS(G), описаним в [3]:

1. Виклик DFS(G^T) для обчислення часу закриття f[u] вершин графа G.

2. Побудова транспонованого графа G^T.

3. Виклик DFS(G^T), але в головному циклі процедури DFS вершини перебирати в порядку спадання значень f[u], обчислених у пункті 1.

4. Дерева пошуку в глибину, отримані в пункті 3 алгоритму – сильно зв'язні компоненти.

Теорема 22.16 з [3] гарантує, що цей алгоритм коректно обчислює СЗК.

Алгоритм STRONGLY_CONNECTED_COMPONENTS реалізуємо в середовищі MATHCAD v.13 (процедура STRONG_COMP). Слід відзначити, що в [1] засобами MATHCAD побудовано процедуру DFS (пошук в глибину), але вона є чутливою до нумерації вершин. Тому, при її застосуванні виникають певні труднощі, пов'язані з перенумераціями. В цій роботі запропоновано варіант, який усуває цей недолік.

Нагадаємо також, що програма DFS проводить класифікацію ребер графа, а саме, виділяє: а) ребра дерева пошуку в глибину – "Т"; б) обернені ребра – "В"; в) прямі, або перехресні ребра – "FC". Отже, відповідно пункту 4 алгоритму, вершини, сполучені ребрами "Т" складають сильно зв'язні компоненти графа G^T, а відтак – графа G.

3. Програмна реалізація. Розглянемо роботу процедури STRONG_COMP на конкретному прикладі. Нехай виробничі стосунки, що склалися між фірми, роботу яких потрібно оптимізувати, відображає орієнтований граф (рис.1).

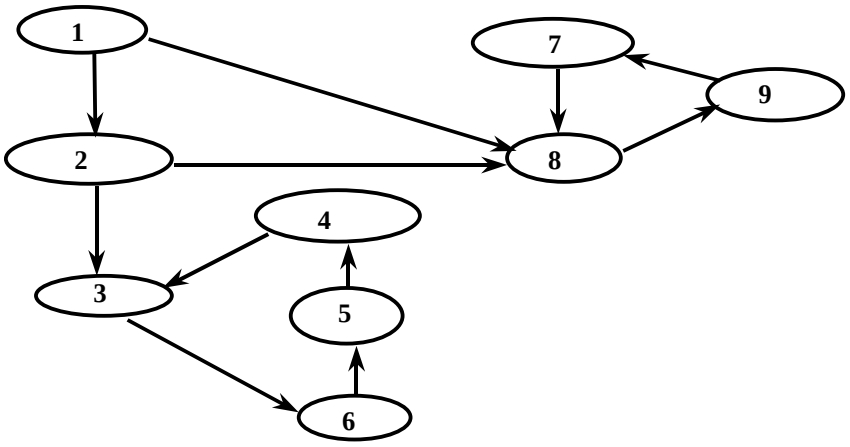

Рис. 1. Граф, що відображає структуру підприємства

1) Задати граф списками суміжностей.

$$T1 := \begin{pmatrix} 2 \\ 8 \end{pmatrix} \quad T2 := \begin{pmatrix} 3 \\ 8 \end{pmatrix} \quad T3_1 := 6 \quad T4_1 := 3$$

$$T5_1 := 4 \quad T6_1 := 5 \quad T7_1 := 8 \quad T8_1 := 9 \quad T9_1 := 7$$

$$GRAF := \begin{pmatrix} 1 & T1 \\ 2 & T2 \\ 3 & T3 \\ 4 & T4 \\ 5 & T5 \\ 6 & T6 \\ 7 & T7 \\ 8 & T8 \\ 9 & T9 \end{pmatrix}$$

2) Операції зі стеком (див. [1]).
3) Допоміжні функції.

```
IS_WHITE(color) := color = "WHITE"

f0(i, j) := 0

f_color(i, j) := "WHITE"

f_kor(i, j) := −777
```

4) Створення структури для зберігання міток "T","B","FC" для ребер графа.

```
a(i, G) := matrix(rows(G_{i,2}), 1, f0)

ini_mark(G) :=
    N ← rows(G)
    c ← matrix(N, 1, f0)
    for i ∈ 1..N
        continue on error a(i, G)
        c_i ← a(i, G)
    c
```

5) Пошук не пройдених ребер у списку суміжності вершини.

```
FIND_MARK(n, G, M) :=
    return −1 if IsScalar(G_{n,2})
    for i ∈ 1..length(G_{n,2})
        return [(G_{n,2})_i ; i] if (M_n)_i = 0
    −1
```

6) Транспонування графа.

```
GRAF_TR(G) :=
    GT ← matrix(rows(G), 1, f0)
    for i ∈ 1..rows(G)
        continue if G_{i,2} = 0
        for j ∈ 1..length(G_{i,2})
            v ← match[(G_{i,2})_j, G^⟨1⟩]_1
            (GT_v)_1 ← G_{i,1} if GT_v = 0
            GT_v ← stack(GT_v, G_{i,1}) otherwise
    GT ← augmen(G^⟨1⟩, GT)
```

7) Пошук в глибину.

```
DFS(G) :=
    nv ← rows(G)
    color ← matrix(nv, 1, f_color)
    d ← matrix(nv, 1, f0)
    f ← matrix(nv, 1, f0)
    pi ← matrix(nv, 1, f_kor)
    M ← ini_mark(G)
    time ← 0
    S ← INI_STACK
    for k ∈ 1..nv
      if IS_WHITE(color_k)
        color_k ← "GRAY"
        d_k ← time ← time + 1
        S ← PUSH(S, k)
        while ¬STACK_EMPTY(S)
          w ← FIND_MARK(k, G, M)
          if w ≠ −1
            a ← match(w_1, G^⟨1⟩)_1
            b ← w_2
            if color_a ≠ "WHITE"
              (M_k)_b ← "B" if color_a = "GRAY"
              (M_k)_b ← "FC" otherwise
            otherwise
              (M_k)_b ← "T"
              color_a ← "GRAY"
              d_a ← time ← time + 1
              pi_a ← k
              S ← PUSH(S, a)
              k ← a
          otherwise
            k ← POP(S)_1
            S ← POP(S)_2
            color_k ← "BLACK"
            f_k ← time ← time + 1
            k ← pi_k
    augmen(G, M, pi, d, f, color)
```

8) Сильно зв'язні компоненти.

```
STRONG_COMP(G) := | WOOD ← DFS(G)
                  | f ← WOOD⟨6⟩
                  | GT ← GRAF_TR(G)
                  | GTN ← reverse(csort(augment(f, GT), 1))
                  | NGRAF ← submatrix(GTN, 1, rows(GTN), 2, 3)
                  | submatrix(DFS(NGRAF), 1, rows(G), 1, 4)
```

9)Результат.

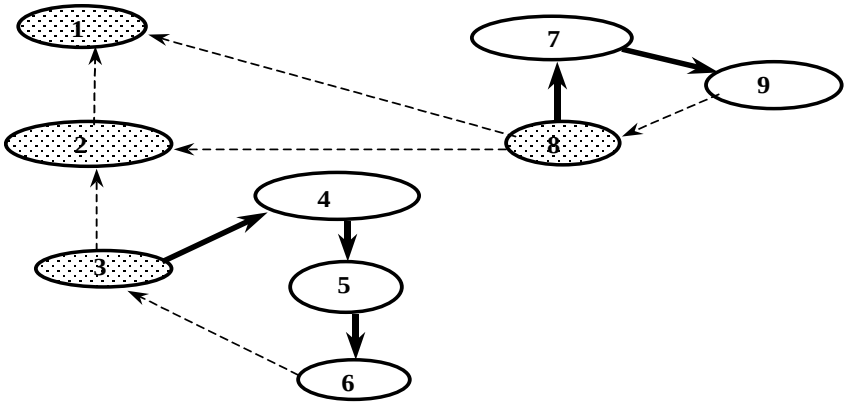

Рис. 2. Транспонований граф

На рис. 2 жирними стрілками зображено дерева пошуку в глибину на графі, транспонованому до вихідного графа. Заштриховані вершини – це корені дерев. Отже, можна виділити чотири сильно зв'язні компоненти: **1, 2, 3-4-5-6, 8-7-9**, і для ефективного управління рекомендувати схему, зображену на рис. 3.

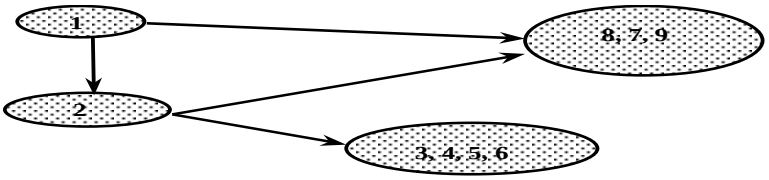

Рис. 3. Оптимальна схема управління.

Висновки. В даній роботі запропонована методика розв'язання задачі ефективного управління підприємством з застосуванням теорії графів, яка передбачає виділення сильно зв'язних компонентів орієнтованого графа і застосування процедури топологічного сортування орієнтованого ациклічного графа. Для виділення СЗК в середовищі MATHCAD розроблено процедуру STRONG_COMP.

Література

1. Кубайчук О.О., Теренчук С.А., Єременко Б.М. Застосування топологічного сортуваня у плануванні будівельних робіт // Містобудування та територіальне планування: Наук.-техн. збірник – К., КНУБА, 2007. – Вип. 28.-с. 102-108.
2. Д. Кнут. Искусство программирования, Т.1. Основные алгоритмы, 3-е изд. –М.: "Вильямс", 2000.
3. Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн, Алгоритмы. Построение и анализ, 2005.
4. Tarjan R. Depth First Seaech and Linear Graph Algorithms. // SIAM Journal on Computing, 1972. – 1(2), p. 146-160.