УДК 004.896

М.В. ПАНЧЕНКО

ПОШУК ОПТИМАЛЬНИХ ШЛЯХІВ У ДЕРЕВІ РІШЕНЬ

1. Вступ

Розвиток сучасного виробництва, науки і всього суспільства загалом можна охарактеризувати як нелінійний та стрибкоподібний. Це обумовлено широким впровадженням сучасних технологій, "технологізацією" виробництва і, як наслідок, підвищенням продуктивності праці, темпів глобалізації світової економіки. Наслідком цього є виникнення нових ризиків та нестабільних умов для роботи великих компаній, що пов′язано з неможливістю передбачити зміни на ринках праці, ресурсів та товарів. Тому в сучасних умовах дедалі актуальнішим стає нове завдання – репрезентувати майбутнє, яке не може інтерпретуватися як звичайне продовження минулого у зв′язку з тим, що це майбутнє набуває істотно відмінних форм і структур.

Універсальних та досконалих підходів для розв′язання цієї проблеми на сьогодні не існує. Є лише спроби будувати можливі сценарії розвитку тих чи інших явищ у майбутньому. Для цього використовуються різноманітні методи якісного характеру, які базуються на використанні експертної інформації. При їх використанні постає проблема інтерпретації нечіткої експертної інформації. Також при обробці великого об′єму експертної інформації виникає проблема великої розмірності.

Разом з тим аналіз сучасного стану досліджень в галузі створення систем прогнозування нестабільних процесів свідчить про наявність таких проблем:

– жоден з існуючих якісних методів сам не може бути застосований для розв′язання наближених до реальності задач;

– проблема обробки нечітких даних при використанні різноманітних ієрархічних та сітьових структур для зберігання експертної інформації;

– недостатня дослідженість проблеми великої розмірності, яка виникає при значному обсязі експертних даних.

Метою роботи є розробка інструментарію для створення проблемно-орієнтованих систем підтримки прийняття рішень, які можуть бути ефективно застосовані для прогнозування поведінки нестабільних процесів.

У роботі [1] дан опис інструментальної системи розробки програмних систем підтримки прийняття рішень, орієнтованої на дослідження нестабільних процесів, які складно прогнозувати за допомогою звичайних методів математичної статистики та регресійного аналізу [3]. В основі системи лежить метод дерева рішення, що є одним з "найадекватніших" методів представлення експертної інформації [2]. Основні недоліки цього методу – "прокляття розмірності", яке посилюється у випадку задання інцедентностей у "нечіткій" формі (тобто експерт оцінює можливість переходу з вершини до вершини у дереві рішення за допомогою функції належності або вектора).

У даній роботі описуються та досліджуються математичні засоби (методи і алгоритми), які дозволяють, з одного боку, позбутися "прокляття розмірності", а, з іншого, – розв′язувати конкретні

практичні задачі. В основі алгоритмів, які пропонуються, лежать методи послідовного аналізу варіантів та вектора спаду [9].

2. Методи пошуку у дереві з чітко заданими дугами

Для представлення зібраної експертної інформації використовується метод дерева рішень. Такий вибір зумовлений можливістю більшої деталізації предметної області, що дозволяє підбирати групи більш вузькоспеціалізованих та кваліфікованих експертів. Аналіз зібраної інформації полягає у знаходженні оптимальних шляхів у побудованому дереві.

Якщо необхідно знайти шляхи у дереві рішень, що з'єднують певні вершини з найменшою вагою, то використовується метод послідовного аналізу варіантів на основі методу Дейкстри [8]. Він застосовується для пошуку найкоротшого шляху із вершини s у вершину t.

Крок 1. Перед початком виконання алгоритму всі вершини та дуги не пофарбовані. Кожній вершині при виконанні алгоритму присвоюється число $d(x)$, що дорівнює довжині найкоротшого шляху з s в x, який включає тільки пофарбовані вершини.

$d(s) = 0$ та $d(x) = \infty$ для всіх $x \neq s$. Пофарбувати вершину s та покласти $y = s$ (y – остання із пофарбованих вершин).

Крок 2. Для кожної непофарбованої вершини x перерахувати величину $d(x)$ таким чином:

$$d(x) = \min\{d(x), d(y) + a(y,x)\}.$$

Якщо $d(x) = \infty$ для всіх непофарбованих вершин x, закінчити процедуру алгоритму: у вихідному графі відсутні шляхи з вершини s у непофарбовані вершини. Інакше пофарбувати ту з вершин x, для якої величина $d(x)$ є найменшою. Крім того, пофарбувати дугу , яка веде в обрану на даному кроці вершину x. Покласти $y = x$.

Крок 3. Якщо $y = t$, завершити процедуру: найкоротший шлях із s в t знайдено (це єдиний шлях із s в t, що складається з пофарбованих дуг). Інакше перейти до кроку 2.

Для знаходження шляхів з найбільшою вагою застосовується метод, розроблений на основі алгоритму Форда [7] (алгоритму пошуку найкоротших шляхів у графі з від'ємними дугами).

Алгоритм пошуку найдовшого (наймовірнішого) шляху. Потрібно знайти найдовший шлях з вершини s у вершину t.

Крок 1. Фарбуємо вершину s. Покладемо $d(s) = 0, d(x) = -\infty, y = s$.

Крок 2. Для всіх вершин (включаючи і непофарбовані) перераховуємо

$$d(x) = \max\{d(x), d(y) + a(y,x)\}.$$

Якщо $d(x) = -\infty$ для всіх непофарбованих вершин x та вершина t не пофарбована, закінчити процедуру алгоритму: у вихідному графі відсутні шляхи з вершини s у непофарбовані вершини. Інакше пофарбувати ту з непофарбованих вершин x, для якої величина $d(x)$ є найбільшою. Крім того, пофарбувати дугу , яка веде в обрану на даному кроці вершину x. Покласти

$y = x$. Якщо d(x) для пофарбованої вершини x збільшується, то розфарбовуємо її та відповідну дугу.

Крок 3. Якщо для всіх непофарбованих вершин $d(x) = -\infty$ та вершина t пофарбована, то найдовший шлях знайдено, закінчити роботу алгоритму.

Твердження 1. Час роботи даного алгоритму в найгіршому випадку становить $O(1{,}5N^3)$, де N – кількість вершин у дереві.

Доведення. Спочатку проаналізуємо обчислювальну складність алгоритму Дейкстри. На першій ітерації цього алгоритму повинні бути проглянутими (N-1) непофарбовані вершини. Так як перегляд вершин відбувається за допомогою рівняння $d(x) = \min\{d(x), d(y) + a(y,x)\}$, то на першій ітерації виконуються $(N-1)$ операція додавання та $(N-1)$ операція порівняння, а також проводиться вибір найменшого з $(N-1)$ чисел (тобто виконується ще $(N-1)$ операцій порівняння). Отже, перша ітерація включає $3 \cdot (N-1)$ операцій. Аналогічно можна показати, що друга ітерація включає $3 \cdot (N-2)$ операцій, третя – $3 \cdot (N-3)$ і т.д. Загальна кількість операцій в алгоритмі Дейкстри визначається співвідношенням

$$\sum_{i=1}^{N} 3*(N-i) = 3N*(N-1)/2 .$$

Окрім указаних операцій, на кожній ітерації алгоритму необхідно також виконувати операції по знаходженню пофарбованих та непофарбованих вершин. Отже, час роботи алгоритму Дейкстри в найгіршому випадку $O(1{,}5N^2)$. Тоді можна зробити висновок, що час роботи алгоритму знаходження найдовшого шляху в найгіршому випадку $O(1{,}5N^3)$. Дійсно, в алгоритмі Дейкстри кожна вершина фарбується лише один раз, а в алгоритмі знаходження найдовшого шляху вона може фарбуватися до $(N-1)$ разів.

3. Методи пошуку у дереві з нечітко заданими дугами

У випадку задання дуг графа нечітко, тобто за допомогою векторів чисел $d(x) = (d_k(x)), k = 1,n$, застосовуються такі методи знаходження наймовірніших шляхів**:**

Метод згорток. Головна ідея цього методу полягає у заміні векторних оцінок числовими за допомогою різних згорток [5]. Наприклад, це може бути звичайне середнє значення елементів вектора чи згортка такого вигляду:

$$a_i = \sum_j a_{ij} * u_j , \qquad (*)$$

де u_j – вага відповідного елемента вектора, або значення, обчислене за методом Ходжа – Лемана,

$$a_i = \beta * \min a_{ij} + (1-\beta) * \sum_j a_{ij} * p_j ,$$

де a_{ij} – елементи вектора; p_j – їх ваги; $\beta \in (0;1)$ – коефіцієнт "колективної обережності"; $\beta = \frac{1}{n}\sum_{j=1}^{n} k_s^i$.

Після цього застосовується вищезгаданий метод пошуку найдовшого шляху. Треба зазначити, що вказані згортки є адитивними. Це означає, що якщо один з векторів є більший за інший (тобто кожен його елемент більший за відповідний елемент іншого вектора), то після згортки він отримує більшу оцінку, що дозволяє уникнути ситуації, коли після застосування згортки та методу пошуку найдовшого шляху буде знайдений шлях, який насправді (без застосування згортки) не є наймовірнішим (тобто існує шлях ймовірніший за нього). Сформулюємо та доведемо це твердження при застосуванні якої-небудь згортки (наприклад, Ходжа-Лемана та згортки (*)).

Твердження 2. Якщо $a_i \geq b_i$, то $a_i^{'} \geq b_i^{'}$, де $a = (a_{1j},...,a_{nj}), b = (b_{1j},...,b_{nj})$, та $a^{'} = \beta * \min a_{ij} + (1-\beta) * \sum_j a_{ij} * p_j$, $b^{'} = \beta * \min b_{ij} + (1-\beta) * \sum_j b_{ij} * p_j$,

де a_{ij} – елементи вектора; p_j – їх ваги; $\beta \in (0;1)$ – коефіцієнт "колективної обережності"; $\beta = \frac{1}{n}\sum_{j=1}^{n} k_s^i$.

Доведення. Так як $a_i \geq b_i$, то $\forall i, j : a_{ij} \geq b_{ij}$. Звідси випливає, що при $0 < \beta < 1$ $\beta * \min a_{ij} \geq \beta * \min b_{ij}$ та $(1-\beta) * \sum_j a_{ij} * p_j \geq (1-\beta) * \sum_j b_{ij} * p_j$. Тоді очевидно, що $a_i^i \geq b_i^{'}$.

Також для згортки (*) сформулюємо і доведемо теорему.

Теорема. Не існує шляху ліпшого за шлях, який був знайдений у дереві за допомогою методу згорток із застосуванням алгоритму знаходження найдовшого шляху.

Тобто, якщо був знайдений шлях $a = (a_1,...,a_k)$, довжина якого $a^{'} = (a_1^{'},...,a_n^i)$, де $a_i^{'} = \sum_{j=1}^{k} a_{ij}$, то не існує шляху $b = (b_1,...,b_m)$ з довжиною $b^{'} = (b_1^{'},...,b_n^{'})$, де $b_i^{'} = \sum_{j=1}^{m} b_{ij}$, для якого $b^{'} > a^{'}$, тобто $\forall i = \overline{1,n} : b_i^{'} > a_i^{'}$ і, відповідно, $\forall i = \overline{1,n} : \sum_{j=1}^{m} b_{ij} > \sum_{j=1}^{k} a_{ij}$.

Доведення. Нехай такий шлях знайдено. Отже, $\exists b : b = (b_1,...,b_m), b^{'} = (b_1^{'},...,b_n^{'}), b_i^{'} = \sum_{j=1}^{m} b_{ij}$ та $b^{'} > a^{'}$. Тобто для якого

$$\forall i = \overline{1,n} : \sum_{j=1}^{m} b_{ij} > \sum_{j=1}^{k} a_{ij} . \qquad (**)$$

Була застосована згортка $a_i = \sum_i a_{ij} * u_j$. Це означає, що у шляху $a = (a_1,...,a_k)$ кожна дуга отримала оцінку $a_i^{"} = \sum_{j=1}^{n} a_{ij} * u_j$ і весь шлях в цілому має довжину $a^{"} = \sum_{i=1}^{k}\sum_{j=1}^{n} a_{ij} * u_j$.

Відповідно і у шляху $b=(b_1,...,b_m)$ кожна дуга отримала оцінку $b_i^{"}=\sum_{j=1}^{n} b_{ij} * u_j$, і весь шлях має довжину $b^{"}=\sum_{i=1}^{k}\sum_{j=1}^{n} b_{ij} * u_j$. Відомо, що після виконання згортки та застосування алгоритму знаходження найдовшого шляху був знайдений шлях $a=(a_1,...,a_k)$, довшого за який не існує. Тобто $a^{"} \geq b^{"}$. Це означає, що $\sum_{i=1}^{k}\sum_{j=1}^{n} a_{ij} * u_j \geq \sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij} * u_j$ або $\sum_{j=1}^{n} u_j \sum_{i=1}^{k} a_{ij} \geq \sum_{j=1}^{n} u_j \sum_{i=1}^{m} b_{ij}$. Перетворюємо цю нерівність таким чином: $\sum_{j=1}^{n} u_j (\sum_{i=1}^{k} a_{ij} - \sum_{i=1}^{m} b_{ij}) \geq 0$. Враховуючи нерівність (**), отримуємо $\sum_{i=1}^{k} a_{ij} - \sum_{i=1}^{m} b_{ij} < 0$. Враховуючи те, що $\forall j = \overline{1,n} : u_j \geq 0$, отримуємо явну суперечність.

Отже, $\neg\exists b=(b_1,...,b_m) : \forall i = \overline{1,n} : \sum_{j=1}^{m} b_{ij} > \sum_{j=1}^{k} a_{ij}$. Теорему доведено.

Якщо потрібно працювати безпосередньо з нечіткими оцінками дуг дерева, то доцільно застосовувати такий метод:

Модифікований алгоритм пошуку найдовшого шляху. Треба знайти найдовший шлях з вершини s у вершину t. Дуги графа задані нечітко за допомогою векторів.

Крок 1. $d_i(s)=(0,...,0)$ та $d_i(x)=(\infty,...,\infty)$ для всіх $x \neq s$, $i=0$.

Крок 2. Для кожної непофарбованої вершини x наступним чином перерахувати величину $d_i(x)$:

$$d_i(x) = \max\{d_i(x), d_i(y) + a(y,x)\}.$$

Очевидно, що ці вектори можуть бути незрівнянними. Тоді

$$d_i(x) = d_i(x) \text{ та } d_{i+1}(x) = d_i(y) + a(y,x), \; i = i+2.$$

Тобто в цю вершину є два можливі шляхи. Якщо ж вектори є зрівнянними, то

$$d_i(x) = \max\{d_i(x), d_i(y) + a(y,x)\} \text{ та } i = i+1.$$

Отже, маємо для вершин x_i такі характеристики:

$$d(x_i) = (d_1(x_i),..., d_k(x_i)),$$

$d_i(x_i)$ – довжина одного з можливих шляхів до вершини x_i.

Після цього вибираємо домінуючі вершини x_i, де $i \in A, A$ – деяка множина, для якої виконується $\neg\exists x_j, j \notin A \neg\exists k : d_k(x_j) \geq d_p(x_i) \forall p$, та фарбуємо їх. Якщо для пофарбованої вершини $d(x)$ збільшилось, то розфарбовуємо її та відповідну дугу.

Крок 3. Якщо всі вершини, для яких $d(x) > -\infty$ пофарбовані, та після виконання кроку 2 жодне $d(x)$ не збільшилось, завершити процедуру: найкоротші шляхи із s в t знайдено. Інакше перейти до кроку 2.

В найкращому випадку, коли всі шляхи у дереві будуть зрівнянними, цей алгоритм працює зі швидкістю алгоритму Дейкстри.

Твердження 3. Час роботи модифікованого алгоритму знаходження найдовшого шляху в найгіршому випадку становить $O(1{,}5 * N^3 * n)$, де N – кількість вершин у дереві, а n – кількість елементів у векторах, якими задаються переходи у дереві.

Доведення. На кожному кроці цього алгоритму повинні бути проглянутими $(N-1)$ вершини. Так як перегляд вершин відбувається за допомогою рівня $d(x) = \max\{d(x), d(y) + a(y,x)\}$, то на кожній ітерації виконуються $(N-1)$ операція додавання та $(N-1)$ операція порівняння, а також проводиться вибір найбільшого з $(N-1)$ векторів (тобто виконується ще $(N-1)$ операцій порівняння). Отже, кожна ітерація містить $3 \cdot (N-1)$ операцій. Оскільки дії виконуються над векторами, то операції порівняння, додавання та ін. вимагатимуть ще n підоперацій, де n – кількість елементів вектора. Кожна вершина може бути пофарбована не більше, ніж $(N-1)$ раз. Загальна кількість операцій в алгоритмі Дейкстри визначається співвідношенням $3(N-1) * (N-1) * (N-1) * n$. Тому час роботи модифікованого алгоритму знаходження найдовшого шляху в найгіршому випадку становить $O(1{,}5 * N^3 * n)$.

"Матричний" метод. Він використовується в тому випадку, коли дерево розбивається на певні рівні. Причому кожна вершина дерева посилається тільки на вершину нижчого рівня. Кожен рівень дерева задається матрицею $A^k = \{a_{ij}\}^k = \{a_{ij}^1, ..., a_{ij}^m\}^k$, де a_{ij} – це імовірність, з якою можливий перехід з i-ї вершини k-го рівня до j-ї вершини $k+1$ рівня. Причому, якщо у k-ї матриці було i стовпчиків, то у $k+1$ матриці буде i рядків.

До кожної матриці дописуємо один рядок та стовпчик. Якщо перша матриця має лише один рядок, то в останній стовпчик записується одиничний вектор. Якщо ж вона має декілька рядків, то в останній стовпчик записується вектор $(v_1, ..., v_p)$, де $v_i = (\frac{1}{c}, ..., \frac{1}{c})$, c – кількість вершин на першому рівні (тобто всі вершини першого рівня рівнозначні). Елемент останнього рядка $a_{l+1,i} = (\sum_{j=1}^{n} a_{ij}^1 * a_{jm+1}^1)$, де l та m –розмірність матриці.

Таким чином, у останньому стовпчику останньої матриці ми отримаємо ваги вершин найнижчого рівня у дереві. Використовуючи отриману інформацію про ваги вершин у дереві, можна визначити наймовірніші шляхи у дереві.

Доведення. Очевидно, що кількість кроків у алгоритмі дорівнює сумі кількості рядків у всіх матрицях, тобто кількості вершин у дереві. На кожному кроці для знаходження ваг вершин нижчого рівня проглядаються всі елементи поточного рядка, які є векторами довжиною n. Тобто на кожному

кроці переглядаються всі дуги, які ведуть з поточної вершини до вершин нижчого рівня. Це означає, що ми перебираємо всі дуги у дереві. Отже, час роботи "матричного" алгоритму становить $r \cdot n$. В найгіршому випадку, коли кількість дуг у дереві максимальна і становить N^2, час роботи цього алгоритму буде $N^2 * n$.

4. Наближені методи пошуку у дереві

Час роботи алгоритму пошуку найдовшого шляху досить великий. Наприклад, при розмірності матриці інцедентності 100×100 і заданні лише тих її елементів, які знаходяться над головною діагоналлю (тобто нема зворотніх зв'язків у дереві), алгоритм виконуватиме приблизно 1600000 операцій (що обумовлено необхідністю знаходження не тільки довжини найдовшого шляху, але й визначенням вершин, які входитимуть у цей шлях). Це означає, що при застосуванні комп'ютера з процесором Pentium600 програма, яка втілюватиме даний алгоритм і розроблена за допомогою C++Builder 5.0, буде працювати приблизно одну хвилину. Отже, необхідно запропонувати наближені методи розв'язку задачі, які вимагатимуть меншого часу для виконання. Наприклад, *модифікацію відомого методу вектора спаду* [6].

Треба знайти найдовший шлях з вершини a_s у вершину a_t. Відомий деякий початковий шлях $a = (a_s, ..., a_t)$, який з'єднує ці вершини. Визначаємо певний окіл r (глибину пошуку).

Крок 1. $y = s$.

Крок 2. За допомогою описаних методів знаходимо найдовший шлях з a_y в a_{y+r}. Замінюємо відповідний сегмент у шляху а: $y = y + 1$.

Крок 3. Якщо $y + r \le t$, то перейти до кроку 2. Якщо ж $y + r > t$, то знаходимо найдовший шлях з y в t. Замінюємо відповідний сегмент у шляху a. Закінчуємо роботу алгоритму.

Таким чином, покращуючи початковий шлях a, ми знаходимо оптимальний (локально оптимальний) шлях. Він не обов'язково буде найдовшим (глобально оптимальним), але у випадку великої кількості даних та обмеженого часу цей метод доцільно застосовувати**.**

Твердження 5. Час роботи модифікованого алгоритму вектора спаду в найгіршому випадку становить $O(R^3 * (K - R))$, де R – глибина пошуку; K – довжина початкового шляху.

Доведення. На кожному кроці алгоритму ми знаходимо за допомогою методу пошуку найдовшого шляху найдовший шлях з вершини a_j у вершину a_{j+R}. Як вже було доведено, це вимагатиме $O(R^3)$ часу (в загальному випадку, час пошуку найдовшого шляху з a_1 в a_n та з a_j в a_{j+R} при $j = \overline{1, n}$, звичайно, однаковий. Але коли граф є деревом, то завжди існує можливість так перенумерувати його вершини, що в матриці інцедентності ненульовими будуть лише елементи над головною діагоналлю. Тоді легко змінити алгоритм так, що час пошуку найдовшого шляху з a_j в

a_{j+R} буде $O(R^3)$ в найгіршому випадку. Для цього потрібно переглядати не всі вершини, а тільки з a_j по a_{j+R}, де $j = \overline{1,n}$). Таких кроків буде $K - R$, бо для останніх R вершин початкового шляху обчислень не проводиться. Отже, час роботи модифікованого алгоритму вектора спаду в найгіршому випадку становить $O(R^3 * (K - R))$.

Також у випадку великої кількості інформації можливе розбиття дерева на піддерева іншими методами, в кожному з яких за допомогою вищезгаданих алгоритмів будуть знаходитися наймовірніші шляхи. Після цього процедура "склейки" будує глобальний наймовірніший шлях.

Декомпозиційний метод. Застосовується, якщо можливе розбиття дерева на певні рівні. Вершини кожного з цих рівнів посилаються лише одна на одну та на верхні вершини наступного рівня.

Потрібно знайти найдовший шлях з вершини s у вершину t.

Крок 1. $y = s$. Визначаємо рівні, до яких належать вершини s та t. Знаходимо найдовші шляхи з вершини s до всіх найнижчих вершин її рівня (тобто до вершин, які посилаються лише на вершини наступного рівня) $y = s$.

Крок 2. $y = y + 1$. Знаходимо найдовші шляхи з верхніх вершин (вершин, на які посилаються нижні вершини вищого рівня) поточного рівня до нижніх. Виділяємо пару вершин, з'єднаних найдовшим шляхом.

Крок 3. Якщо поточний рівень містить вершину t і вона відноситься до верхніх вершин, то перейти до кроку 4. Якщо поточний рівень містить вершину t і вона не відноситься до верхніх вершин, то знайти найдовші шляхи з верхніх вершин поточного рівня у вершину t. Знайти ту з верхніх вершин, шлях з якої до t найдовший. Перейти до кроку 4. Якщо поточний рівень не містить вершину t, то перейти до кроку 2.

Крок 4. "Склеюємо" шляхи сусідніх рівнів таким чином: спочатку намагаємося знайти найдовший шлях з останньої вершини шляху верхнього рівня до першої вершини шляху нижчого рівня. Якщо ці вершини взагалі неможливо з'єднати, то відкидаємо останню вершину шляху з вищого рівня і знаходимо найдовший шлях з останньої вершини вищого рівня до першої вершини нижчого рівня. Якщо ж і ці вершини неможливо з'єднати, то відкидаємо першу вершину шляху нижчого рівня, і т.д.

В найгіршому випадку доведеться шукати найдовший шлях між першою вершиною вищого рівня та останньою вершиною нижчого рівня. Включаємо знайдені вершини у загальний шлях. Якщо ж з'єднати таким чином вершини двох рівнів не вдалося, то в даному дереві неможливо з'єднати вершини s та t.

Продовжуючи дану процедуру, ми з'єднаємо вершини s і t.

Очевидно, що знайдений шлях може не бути глобально оптимальним. Але такий підхід дозволяє обробляти за короткий час великі об'єми даних.

Нехай $a^i = (a^i_j), i = \overline{1,n}, j = \overline{1,m^i}$ – i-ий рівень дерева, a^i_j – вершини цього рівня. На кожному рівні m^i вершин. $b^i = (b^i_k), k = \overline{1,q^i}$ – "верхні" вершини цього рівня, $c^i = (c^i_t), t = \overline{1,w^i}$ – "нижні" вершини цього рівня. Тоді обробка декомпозиційним алгоритмом цього рівня вимагатиме часу $q^i * w^i * (m^i)^3$ (тут $(m^i)^3$ – це час пошуку найдовшого шляху u^i від верхньої вершини до нижньої в найгіршому випадку). Процедура "склеювання" шляхів i-го та $i+1$-го рівнів в найгіршому випадку займає

$$(m^i + m^{i+1})^3 * (m^i + m^{i+1}) * 3.$$

В цій формулі $(m^i + m^{i+1})^3$ – це час пошуку найдовшого шляху між будь-якими двома вершинами рівнів i та $i+1$.

Спочатку відкидається остання вершина шляху u^i. Шукається найдовший шлях, який з'єднує останню вершину шляху u^i та першу вершину шляху u^{i+1}. Якщо ці вершини можливо з'єднати, то знаходження відповідного шляху в найгіршому випадку займатиме $(m^i + m^{i+1})^3$. Якщо ж такого шляху не існує, то на з'ясування цього в найгіршому випадку піде час $(m^i + m^{i+1})^3$.

Якщо шлях не був знайдений, вилучаємо першу вершину шляху u^{i+1} та знаходимо найдовший шлях з останньої вершини шляху u^i до першої вершини шляху u^{i+1}. Навіть якщо такого шляху не існує, в найгіршому випадку на з'ясування цього піде $(m^i + m^{i+1})^3$ часу.

Аналогічна ситуація виникає і коли вилучається остання вершина шляху u^i та перша вершина шляху u^{i+1}. Отже, маємо три ситуації, при розгляді кожної в найгіршому випадку вимагається $(m^i + m^{i+1})^3$. Очевидно, що в найгіршому випадку доведеться таким чином перебрати всі елементи двох шляхів і в результаті з'єднати першу вершину шляху u^i та останню вершину шляху u^{i+1}. Тому у формулі (*) другим додатком є $(m^i + m^{i+1}) * 3$.

Отже, загалом можна сформулювати таке твердження:

Твердження 6. Час роботи декомпозиційного алгоритму в найгіршому випадку становить

$$\sum_{i=1}^{n} q^i * w^i * (m^i)^3 + \sum_{i=1}^{n-1} (m^i + m^{i+1})^3 * (m^i + m^{i+1}) * 3$$

або

$$\sum_{i=1}^{n-1} (q^i * w^i * (m^i)^3 + (m^i + m^{i+1})^3 * (m^i + m^{i+1}) * 3) + q^n * w^n * (m^n)^3.$$

Алгоритм послідовного аналізу варіантів. Необхідно знайти найдовший шлях з вершини s у вершину t.

Крок 1. $y = s$. Визначаємо рівні, до яких належать вершини s та t.

Крок 2. Знаходимо найдовші шляхи із вершини s до всіх найнижчих вершин її рівня (тобто до вершин, які посилаються лише на вершини наступного рівня). $y = s$.

Крок 3. Якщо поточний рівень містить вершину t і вона відноситься до верхніх вершин, то перейти до кроку 4. Якщо поточний рівень містить вершину t і вона не відноситься до верхніх вершин, то знайти найдовші шляхи з верхніх вершин поточного рівня у вершину t. Знайти ту з верхніх вершин, шлях з якої до t найдовший. Перейти до кроку 4. Якщо поточний рівень не містить вершину t, то перейти до кроку 2.

Крок 4. "Склеюємо" шляхи сусідніх рівнів i и $i+1$ таким чином: задаються коефіцієнти u_i та h_i. Для шляху g, яким буде проходити через вершини цих двох рівнів, повинно виконуватися $g \geq u_i(f_i + f_{i+1})$, де f_i, f_{i+1} – найдовші шляхи рівнів i та $i+1$. Так само задається глибина пошуку t. Знаходимо найдовший шлях з останньої вершини шляху верхнього рівня до першої вершини шляху нижнього рівня. Якщо $g \geq u_i(f_i + f_{i+1})$, то процедуру склейки завершено. Інакше відкидаємо вершини із шляхів поточних рівнів (дозволяється відкидати не більш, ніж h_i вершин) та повторюємо процедуру "склейки". Якщо не вдалося з'єднати два рівні, то понижуємо значення u_i і повторюємо все з початку.

5. Висновки

Пропонується методика прогнозування розвитку нестабільних процесів на основі використання експертної інформації відносно характеру розвитку окремих параметрів, що впливають на результати прогнозу. Пошук наймовірніших шляхів у дереві рішень здійснюється за допомогою оригінальних методів. Основна їх наукова новизна така:

– вперше розроблено метод, який дозволяє знаходити наймовірніші шляхи у дереві рішень із чітко заданими дугами;

– вперше запропоновано метод, що дозволяє знаходити наймовірніші шляхи у дереві рішень, побудованому на основі нечіткої експертної інформації;

– вперше розроблено методи пошуку оптимальних шляхів у дереві рішень, які дозволяють знаходити локальні розв'язки та вирішують проблему великої розмірності, що виникає при значному обсязі даних.

Подальші дослідження у цьому напрямку полягають у розробці більш досконалих методів збору експертної інформації та наближених методів пошуку оптимальних шляхів у дереві рішень, які не будуть вимагати задання початкового шляху або розбиття дерева на певні рівні (адже при розв'язанні практичних задач це не завжди можливо здійснити) .

СПИСОК ЛІТЕРАТУРИ

1. Волошин О.Ф., Панченко М.В. Використання експертного оцінювання для якісного прогнозування на основі багатопараметричних залежностей// Математичні машини і системи. – 2002. – № 2. – С. 83 – 90.
2. Згуровський М. Хто бачить майбутнє, той перемагає// Дзеркало тижня. – 2001. – № 25. – С. 14.
3. Тэрано Т., Асаи К., Сугэно М. Прикладные нечеткие системы. – М.: Мир, 1993. – 287 с.
4. Волошин А.Ф. Метод локализации области оптимума в задачах математического программирования // Доклады АН СССР. Серия: кибернетика и теория регулирования. – Т. 293. – 1989. – № 3. – С. 549 – 553.
5. Макаров В. И. и др. Основы принятия решений и теории выбора. – М.: Наука, 1983. – 352 с.

6. Волошин О.Ф., Гнатієнко Г.М. Побудова колективного ранжування за мірою рангів об'єктів // Вісник Київського університету. Серія: кібернетика. –2001. – № 4. – С.140 – 147.
7. Волошин А.Ф., Гнатиенко Г.М. Построение компромиссной ранжировки в задачах группового выбора // Труды Всесоюзной конференции "Проблемы теоретической кибернетики". – Волгоград. – 1990. – С.15 –18.
8. Майника Э. Алгоритмы оптимизации на сетях и графах. – М.: Мир, 1981. – 476 с.
9. Сергиенко И.В. Математические модели и методы решения задач дискретной оптимизации. – Киев: Наукова думка, 1985. – 384 с.