старіючими вихідними елементами ще два показники надійності: частоту відмов та інтенсивність відмов за заданої умови готовності.

3. На основі методу твірних функцій побудовано моделі ймовірнісних показників надійності симетричних ізотропних систем , розгалужених до 2-го рівня, зі старіючими вихідними елементами, що дозволяє оцінювати ймовірнісні показники таких систем. Побудовано моделі для розрахунку традиційних характеристик надійності невідновлюваних систем, передбачених державними стандартами України, для симетричних ізотропних систем, розгалужених до 2-го рівня, зі старіючими вихідними елементами, що дозволяє оцінювати такі показники надійності для вказаних систем.

4. У світовій практиці одним із основних показників надійності вважається інтенсивність відмов, особливо для комплектуючих. З точки зору прогнозування надійності розгалужених систем, яке потрібне для оцінок ймовірностей відмов, порівняння структур систем з надійності, визначення потенційних проблем надійності, планування обслуговування та планування забезпечення запасними виробами і ремонтними послугами, оцінки витрат на експлуатацію, важливим є використання цих традиційних показників, що у свою чергу дасть можливість визначити експлуатаційні показники розгалужених систем, такі, як частоту профілактики, ремонту, кількість запасних частин.

1. *Голинкевич Т.А.* Прикладная теория надежности. – М.: Высшая школа, 1977 – 159 с.
2. *Козлов Б., Ушаков И.* Справочник по расчету надежности аппаратуры радиоэлектроники и автоматики. – М.: Советское радио, 1975. – 472 с.
3. *Райншке К., Ушаков И.* Оценка надежности систем с использованием графов. – М.: Радио и связь, 1988. – 209 с.
4. *Ушаков И.А.* Вероятностные модели информационно-вычислительных систем. – М.: Радио и связь, 1991.

Поступила 11.02.2010р.

УДК 629.52.7

О.А. Машков, д.т.н., професор, ВАК України; В.Р. Косенко

МЕТОДИКА СТВОРЕННЯ ЕКСПЕРТНОЇ СИСТЕМИ ФУНКЦІОНАЛЬНО-СТІЙКОГО ІНФОРМАЦІЙНО-КЕРУЮЧОГО КОМПЛЕКСУ ДИНАМІЧНОГО ОБ'ЄКТА НА ОСНОВІ БАЗИ ЗНАНЬ

ПОСТАНОВА ЗАДАЧІ ДОСЛІДЖЕНЬ. На даний час склалася певна технологія розробки експертної системи, яка включає шість таких етапів: ідентифікація, концептуалізація, формалізація, виконання, тестування і

дослідна експлуатація.

Етап ідентифікації. Етап ідентифікації пов'язаний, передусім, з осмисленням тих задач, які має розв'язати майбутня експертна система, і формуванням вимог до неї. Результатом даного етапу є відповідь на питання, що треба зробити і які ресурси необхідно задіяти (ідентифікація задачі, визначення учасників процесу проектування та їх ролі при цьому, виявлення ресурсів і цілей).

Як правило, у розробці експертної системи беруть участь не менше трьох-чотирьох чоловік: експерт, один або два інженери зі знань та програміст, котрий залучається для модифікації та узгодження інструментальних засобів. Крім того, до процесу розробки експертної системи можуть по мірі необхідності залучатися інші учасники. Наприклад, інженер зі знань може запросити інших експертів, щоб пересвідчитися в правильності висновків основного експерта, показовості тестів, які демонструють особливості задачі, яка розглядається; збігу поглядів різних експертів на якість розв'язків, що пропонуються. При складних системах вважається доцільним залучати до основного циклу розробки декількох експертів. Однак у цьому випадку, як правило, потрібно, щоб один з експертів відповідав за несуперечливість знань, які повідомляються колективом експертів.

Ідентифікація задачі полягає в складанні неформального (вербального) опису, у якому вказуються загальна характеристика задачі, підзадачі, що виділяються всередині даної задачі, ключові поняття (об'єкти), їх вхідні (вихідні) дані, ймовірний спосіб розв'язування задачі, а також відповідні знання для її вирішення.

У процесі ідентифікації задачі інженер зі знань та експерт працюють у тісному контакті. Початковий неформальний опис задачі експертом використовується інженером зі знань для уточнення термінів і ключових понять. Експерт коректує опис задачі, пояснює, як її розв'язати і які міркування лежать в основі того або іншого розв'язку. Після декількох циклів, що уточнюють опис, експерт та інженер зі знань отримують остаточний неформальний опис задачі.

При проектуванні експертної системи типовими ресурсами є джерела знань, час розробки, обчислювальні засоби і об'єм фінансування. Для експерта джерелами знань служать його попередній досвід з розв'язання задач, книги, відомі приклади розв'язування задач, а для інженера зі знань - досвід у розв'язуванні аналогічних задач, методи представлення знань і маніпулювання ними; програмні інструментальні засоби. При визначенні часу розробки враховуються терміни розробки і впровадження експертної системи, яку складають, як правило, не менше ніж за рік (при трудомісткості 5 чол.). Визначення обсягу фінансування впливає істотним чином на процес розробки, оскільки при недостатньому фінансуванні перевага може бути віддана адаптації існуючої, а не розробці оригінальної нової системи.

При ідентифікації цілей важливо відрізняти те, ради чого створюється експертна система, а також задачі, які вона повинна розв'язувати. Прикладами можливих цілей є: формалізація неформальних знань експертів; поліпшення якості рішень, що приймаються експертом; автоматизація рутинних аспектів роботи експерта (користувача); тиражування знань експерта.

Етап концептуалізації. На даному етапі проводиться змістовний аналіз проблемної сфери, виявляються поняття, якими послуговуються, та їх взаємозв'язки, визначаються методи розв'язування задач. Цей етап завершується створенням моделі предметної сфери (ПС), що включає основні концепти і відносини. На етапі концептуалізації визначаються такі особливості задачі: типи доступних даних; початкові дані, що виводяться, підзадачі основної задачі; стратегії, які використовуються, і гіпотези; види взаємозв'язків між об'єктами ПС, типи відносин, що використовуються (ієрархія, причина - наслідок, частина - ціле тощо); процеси, задіяні в ході розв'язування; склад знань, потрібний для розв'язання задачі; типи обмежень, накладені на процеси, які задіяні в ході розв'язування; склад знань, необхідний для обґрунтування рішень.

1. МОДЕЛЬ ПРЕДМЕТНОЇ СФЕРИ

Існує два підходи до процесу побудови моделі предметної сфери, яка є метою розробників експертної системи на етапі концептуалізації. Ознаковий або атрибутивний підхід передбачає наявність отриманої від експертів інформації у вигляді трійок *об'єкт - атрибут - значення атрибута*, а також наявність початкової інформації. Цей підхід розвивається в рамках напряму, що отримав назву формування знань або "машинне навчання" (machine learning).

Другий підхід, так званий структурний (або когнітивний), здійснюється шляхом виділення елементів предметної сфери, їх взаємозв'язків і семантичних відносин.

Для атрибутивного підходу характерна наявність найбільш повної інформації про предметну сферу, а саме: об'єкти, їх атрибути і значення атрибутів. Крім того, істотним моментом є використання додаткової початкової інформації, яка задається групуванням об'єктів у класи за тим або іншим змістовним критерієм. Трійки *об'єкт - атрибут - значення атрибута* можуть бути отримані за допомогою так званого методу рекласифікації, який заснований на припущенні, що задача є об'єктно-орієнтованою і об'єкти задачі добре відомі експерту. Ідея методу полягає в тому, що конструюються правила комбінації значень атрибутів, які дозволяють відрізнити один об'єкт від іншого. Навчальна інформація може бути задана на основі прецедентів правильних експертних висновків, наприклад за допомогою методу здобування знань, що отримав назву "аналіз протоколів думок уголос".

При наявності початкової інформації для формування моделі предметної сфери на етапі концептуалізації можна використати весь арсенал методів, що розвиваються в рамках задачі розпізнавання образів.

Структурний підхід до побудови моделі предметної сфери передбачає виділення таких когнітивних елементів знань: поняття; взаємозв'язки; мета поняття; семантичні відносини.

Поняття *предметної сфери,* що виділяються, повинні утворювати систему, під якою розуміється сукупність понять, що володіє такими властивостями: унікальністю (відсутністю надмірності); повнотою (досить повним описом різних процесів, фактів, явищ предметної сфери); достовірністю (відповідністю виділених одиниць смислової інформації їх реальним найменуванням) і несуперечністю (відсутністю омонімії).

При побудові системи понять за допомогою *"методу локального представлення"* експерта просять розбити задачу на підзадачі для переліку цільових станів і опису загальних категорій мети. Далі для кожного розбиття (локального уявлення) експерт формулює інформаційні факти і дає їм чітке найменування (назву). Вважається, що для успішного розв'язування задачі побудови моделі предметної сфери кількість таких інформаційних фактів у кожному локальному уявленні, якими людина здатна одночасно маніпулювати, повинно дорівнювати приблизно семи.

"Метод обчислення коефіцієнта використання" заснований на такій гіпотезі. Елемент даних (або інформаційний факт) може вважатися поняттям, якщо:

1. Він використовується у великій кількості підзадач.
2. Використовується з великою кількістю інших елементів даних.
3. Рідко використовується разом з іншими елементами даних у порівнянні із загальним числом його використання в усіх підзадачах (це і є коефіцієнт використання).

Отримані значення можуть служити критерієм для класифікації всіх елементів даних і, таким чином, для формування системи понять.

"Метод формування переліку понять" полягає в тому, що експертам (бажано, щоб їх було більше двох) дається завдання скласти список понять, які належать до предметної сфери, що досліджується. Поняття, виділені всіма експертами, включаються в систему понять, інші підлягають обговоренню.

"Рольовий метод" полягає в тому, що експерту дається завдання навчити інженера зі знань розв'язуванню деяких задач предметної сфери. Таким чином, експерт відіграє роль учителя, а інженер зі знань - роль учня. Навчальний процес записується на магнітофон. Третій учасник прослуховує магнітофонну плівку і виписує на папері всі поняття, ужиті вчителем і учнем.

При використанні методу *"складання списку елементарних дій"* експерту дається завдання скласти такий список при розв'язуванні задачі в довільному порядку.

У методі *"складання змісту підручника"* експерту пропонується представити ситуацію, в якій його попросили написати підручник. Для цього необхідно скласти на папері перелік передбачуваних розділів, параграфів, пунктів і підпунктів книги.

"Текстологічний метод" формування системи понять полягає в тому,

що експерту дається завдання виписати з покажчика (книг за спеціальністю) терміни деяких елементів, які являють собою одиниці смислової інформації.

Група методів установлення взаємозв'язків передбачає встановлення семантичної близькості між окремими поняттями. В основі встановлення взаємозв'язків лежить психологічний ефект *"вільних асоціацій",* а також фундаментальна категорія близькості об'єктів або концептів.

Ефект вільних асоціацій полягає в такому. Учень відповідає на задане слово найпершим словом, що спало на думку. Як правило, реакція більшості учнів (якщо слова були загальновживаними) виявляється однаковою. Кількість переходів у ланцюжку може служити мірою *"смислової відстані"* між двома поняттями. Численні досліди підтверджують гіпотезу, що для будь-яких двох слів (понять) існує асоціативний ланцюжок, який складається не більше ніж із семи слів.

"Метод вільних асоціацій" заснований на психологічному ефекті, описаному вище. Експерту пропонується назвати якомога швидше перше-ліпше поняття, що спало на думку, зі сформованої раніше системи понять. Далі проводиться аналіз отриманої інформації.

У методі *"сортування карток"* початковим матеріалом служать поняття, виписані на картки. Застосовуються два варіанти методу. У першому - експерту задаються певні глобальні критерії предметної сфери, якими він повинен оперувати при розкладанні карток на групи. У другому випадку, коли сформулювати глобальні критерії неможливо, експерту дається завдання розікласти картки на групи відповідно до інтуїтивного розуміння семантичної близькості понять, що пред'являються.

"Метод виявлення регулярності" заснований на гіпотезі про те, що елементи ланцюжка понять, які людина згадує з певною регулярністю, мають тісний асоціативний взаємозв'язок. Для експерименту довільно відбирається 20 понять. Експерту надається одне з відібраних чисел. Процедура повторюється до 20 разів, причому кожного разу початкові концепти повинні бути різними. Потім інженер зі знань аналізує отримані ланцюжки з метою знаходження понять, що постійно повторюються регулярності). У середині виділених таким чином угрупувань установлюються асоціативні взаємозв'язки.

Крім розглянутих вище неформальних методів, для встановлення взаємозв'язків між окремими поняттями застосовуються також формальні методи. Сюди насамперед належать методи семантичного диференціала і репертуарних гратів.

Виділені поняття предметної сфери і встановлені між ними взаємозв'язки служать основою для подальшої побудови системи метапонять, осмислених у контексті предметної сфери системи угрупувань. Для визначення цих угрупувань застосовують як неформальні, так і формальні методи.

Інтерпретація, як правило, легше дається експерту, коли угрупування отримані неформальними методами. У цьому випадку виділені класи більш зрозумілі експерту. Причому в деяких предметних сферах зовсім не обов'язково встановлювати взаємозв'язки між поняттями, оскільки

метапоняття, образно кажучи, "лежать на поверхні".

Останнім етапом побудови моделі предметної сфери при концептуальному аналізі є встановлення семантичних відносин між виділеними поняттями і метапоняттями. У*становити семантичні відносини* означає визначити специфіку взаємозв'язку, отриманого внаслідок застосування тих або інших методів. Для цього необхідно кожний зафіксований взаємозв'язок осмислити і віднести його до того або іншого типу зв'язків.

Існує близько 200 базових зв'язків, наприклад "частина - ціле", "рід - вид", "причина - наслідок", просторові, часові та інші зв'язки. Для кожної предметної сфери, крім загальних базових зв'язків, можуть існувати й унікальні взаємозв'язки.

"Прямий метод" установлення семантичних зв'язків базується на безпосередньому осмисленні кожного взаємозв'язку. У тому випадку, коли експертам важко дати інтерпретацію виділеного взаємозв'язку, йому пропонується така процедура. Формуються трійки: поняття 1 - зв'язок - поняття 2. Поруч з кожною трійкою записується коротка пропозиція або фраза, побудована так, щоб поняття 1 і поняття 2 входили б у цю пропозицію. Як зв'язки використовуються тільки змістовні відносини і не застосовуються невизначені зв'язки типу "схожий на" або "пов'язаний з".

Для *"непрямого методу"* необов'язково мати взаємозв'язки, а досить лише наявності системи понять. Формулюється певний критерій, для якого з системи понять вибирається певна сукупність концептів, яка надається експерту з позицією дати вербальний опис сформульованого критерію. Концепти пред'являються експерту всі відразу (бажано на картках). У разі ускладнень вдаються до розбиття відібраних концептів на групи за допомогою більш дрібних критеріїв. Початкова кількість концептів може бути довільною, але після розбиття на групи в кожній з таких груп повинно бути не більше десяти концептів. Після того як складені описи по всіх групах, експерту пропонують об'єднати їх в один.

Наступний крок у непрямому методі встановлення семантичних відносин — це аналіз тексту, складеного експертом. Концепти замінюють цифрами (допускається початкова нумерація), а зв'язки залишають. Тим самим вибудовується певний граф, вершинами якого служать концепти, а дугами - зв'язки (наприклад, "внаслідок", "приводить до", "висловлюючись з одного боку", "зумовлюючи", "поєднуючись", "визначає", "аж до" тощо). Цей метод дозволяє встановлювати відносини не лише базові, але специфічні для конкретної предметної сфери.

Розглянуті вище методи формування системи понять і цільових уявлень, встановлення взаємозв'язків і семантичних відносин у різних комбінаціях застосовуються на етапі концептуалізації при побудові моделі предметної сфери.

Етап формалізації. Відтак усі ключові поняття і відносини виражаються певною формальною мовою, яка або вибирається з числа вже існуючих, або створюється наново. Іншими словами, на даному етапі визначаються склад засобів І способи представлення декларативних і

процедурних знань, здійснюється це подання і в результаті формується описання розв'язування задачі експертної системи запропонованою (інженером зі знань) формальною мовою.

Виходом етапу формалізації є описування того, як дана задача може бути представлена у вибраному або розробленому формалізмі. Сюди належить припис способів представлення знань(фрейми, сценарії, семантичні мережі і т.д.), визначення способів маніпулювання ними (логічний висновок, аналітична модель, статистична модель тощо) та інтерпретації знань.

Етап виконання. Мета цього етапу - створення одного або декількох прототипів експертної системи, що розв'язують необхідні задачі. У подальшому на даному етапі за результатами тестування і дослідною експлуатацією створюється кінцевий продукт, придатний для промислового використання. Розробка прототипу полягає в програмуванні його компонентів або виборі їх із відомих інструментальних засобів і наповненні бази знань.

У створенні прототипу головне полягає в тому, щоб він забезпечив перевірку адекватності ідей, методів і способів представлення знань задачам, що розв'язуються. Створення першого прототипу повинно підтвердити, що вибрані методи розв'язувань і способи уявлення придатні для успішного розв'язування, принаймні, ряду задач з актуальної предметної галузі, а також продемонструвати тенденцію до отримання високоякісних і ефективних розв'язків для всіх задач предметної сфери в міру збільшення обсягу знань.

Після розробки першого прототипу *експертної системи-1* коло задач, що пропонуються для розв'язання, розширюється, і збираються побажання та зауваження, які повинні бути враховані в черговій версії системи *експертна система-2.* Здійснюється розвиток експертної системи-1 шляхом додавання "дружнього" інтерфейсу, засобів для дослідження бази знань і ланцюжків висновків, що генеруються системою, а також засобів для збору зауважень користувачів і засобів для зберігання бібліотеки задач, розв'язаних системою.

Етап тестування. У ході даного етапу проводиться оцінка вибраного способу представлення знань в експертній системі загалом. Для цього інженер зі знань підбирає приклади, що забезпечують перевірку всіх можливостей розробленої експертної системи.

Розрізняють такі джерела помилок у роботі системи: тестові приклади, уведення-виведення, правила висновку, керуючі стратегії.

Показові тестові приклади є найбільш очевидною причиною невдалої роботи експертної системи. У гіршому випадку вони можуть виявитися взагалі поза предметною сферою, на яку розрахована експертна система. Однак частіше безліч тестових прикладів виявляються дуже однорідними і не охоплюють усю предметну сферу. Тому при підготовці тестових прикладів потрібно класифікувати їх за підпроблемами предметної сфери, виділяючи стандартні випадки, визначаючи кордони скрутних ситуацій тощо.

Уведення-виведення характеризується даними, отриманими під час діалогу з експертом, і висновками, виробленими експертною системою в ході

пояснень. Методи надбання даних можуть не давати необхідних результатів, оскільки, наприклад, задавалися неправильні питання або зібрана інформація не була вичерпною. Крім того, питання системи можуть бути важкими для розуміння, багатозначними і не відповідними до знань користувача. Помилки при введенні можуть виникати також через незручну для користувача вхідну мову. У ряді додатків для користувача зручне введення не тільки в друкованій, але і в графічній або звуковій формі.

Вихідні повідомлення (висновки) системи можуть виявитися незрозумілі користувачеві (експерту) з різних причин. Наприклад, їх може бути дуже багато або, навпаки, дуже мало. До того ж причиною помилок може бути невдала організація, невпорядкованість висновків або невідповідний користувачеві рівень абстракцій з незрозумілою йому лексикою.

Найбільш поширене джерело помилок у міркуваннях стосується правил висновку. Причина тут часто полягає у відсутності врахування взаємозалежності сформованих правил, та помилковості, часом суперечності й неповноті правил, якими користуються. Якщо невірне посилання правила, то це може призвести до застосування його в невідповідному контексті. Якщо дія правила помилкова то важко передбачити правильний кінцевий результат. Правило може бути помилковим, коли при коректності його умови і дії порушена відповідність між ними.

Нерідко до помилок у роботі експертної системи призводять керуючі стратегії, що застосовуються. Зміна стратегії, наприклад, спричиняється необхідністю, коли експертна система аналізує сутності в порядку, який відрізняється від "природного" для експерта. Послідовність, з якою розглядаються дані експертної системи, не тільки впливає на ефективність її роботи, але може приводити і до зміни кінцевого результату. Так, розгляд правила А до правила В здатний призвести до того, що правило В завжди буде ігноруватися системою. Зміна стратегії буває також необхідною і у разі неефективної роботи експертної системи. Крім того, недоліки в керуючих стратегіях можуть призвести до надмірно складних висновків і пояснень експертної системи.

Критерії оцінки експертної системи залежать від різних поглядів. Наприклад, при тестуванні експертної системи-1 головною в оцінці роботи системи є повнота і безпомилковість правил висновку. При тестуванні промислової системи переважає думка інженера зі знань, якого насамперед цікавить питання оптимізації представлення і маніпулювання знаннями. І нарешті, при тестуванні експертної системи після дослідної експлуатації оцінка проводиться з погляду користувача, зацікавленого в зручності роботи і отриманні практичної користі.

Етап дослідної експлуатації. На цьому етапі перевіряється придатність експертної системи для кінцевого користувача. Придатність експертної системи для користувача визначається в основному як зручністю роботи з нею, так і її корисністю. Під корисністю експертної системи розуміється її здатність у ході

діалогу визначати потреби користувача, виявляти й усувати причини невдач у роботі, а також задовольняти вказані потреби користувача (розв'язувати поставлені задачі). У свою чергу, зручність роботи з експертною системою передбачає природність взаємодії з нею (спілкування у звичному стані, який не стомлює користувача), гнучкість експертної системи (здатність її налаштовуватися на різних користувачів, а також урахування, змін у кваліфікації одного і того ж користувача) і стійкість системи до помилок (здатність не розладнюватися при помилкових діях недосвідченого користувача).

У ході розробки експертної системи майже постійно здійснюється її модифікація. Виділяють такі види модифікації системи: переформулювання понять і вимог, переконструювання подання знань у системі та удосконалення прототипу.

2. СТРАТЕГІЇ УПРАВЛІННЯ ВИСНОВКОМ

Розробка стратегії. Одним із важливих питань, які виникають при проектуванні керівного компоненту систем, заснованих на знаннях, є вибір методу пошуку розв'язку, тобто стратегії висновку. Від обраного методу пошуку буде залежати порядок застосування і спрацювання правил. Процедура вибору зводиться до визначення напряму пошуку і способу його здійснення. Процедури, що реалізовують пошук, звичайно "зашиті" в механізм висновку, тому до більшості систем інженери знань не мають доступу, а отже, не можуть у них нічого змінювати за своїм бажанням.

При розробці стратегії управління висновком необхідно відповісти на два питання:

1. Яку точку в просторі станів брати за вихідну? Справа в тому, що ще до початку пошуку розв'язку система, заснована на знаннях, повинна якимось чином вибрати початкову точку пошуку — у прямому або зворотному напрямках.
2. Як підвищити ефективність пошуку рішення? Щоб домогтися підвищення ефективності пошуку розв'язку, необхідно знайти евристики вирішення конфліктів, пов'язаних з існуванням декількох можливих шляхів для продовження пошуку в просторі станів, оскільки потрібно відкинути ті з них, які явно не ведуть до пошукового розв'язку.

Основні методи пошуку. У системах, база знань яких нараховує сотні правил, цілком бажаним е використання певної стратегії управління висновком, що дозволяє мінімізувати час пошуку розв'язку і тим самим підвищити ефективність висновку. До таких стратегій належать: а) пошук у просторі станів; б) розбиття на підзадачі; в) використання формальної логіки при розв'язуванні задач.

Пошук у просторі станів. Суть пошуку полягає в тому, що при виборі чергової підцілі у просторі станів перевага, за можливістю, надається тій, яка відповідає наступному, більш детальному рівню опису задачі. Простір станів - це граф, вершини якого відповідають ситуаціям, що зустрічаються в задачі ("проблемні ситуації"), а розв'язування задачі зводиться до пошуку шляху в

цьому графі. При пошуку в ширину, навпаки, система аналізує всі ознаки, що знаходяться на одному рівні простору станів, і лише потій переходить до ознак наступного рівня детальності. Фахівці деяких вузьких сфер більш цінують пошук у глибину, оскільки він дозволяє зібрати воєдино всі ознаки, пов'язані з висунутою гіпотезою. Універсали ж віддають перевагу пошуку в ширину, оскільки в цьому випадку аналіз не обмежується заздалегідь окресленими довкола ознаками. Особливості простору пошуку багато в чому визначають доцільність застосування тієї або іншої стратегії: наприклад, програми для гри в шахи будуються на основі пошуку в ширину, оскільки при використанні пошуку в глибину *число* ходів, що аналізується, може бути надто великим. До методів пошуку в просторі станів належить альфа-бета алгоритм. Задача зводиться до зменшення простору станів шляхом видалення в ньому гілок, не перспективних для пошуку успішного розв'язку. Тому переглядаються тільки ті вершини, у які можна попасти внаслідок наступного кроку, після чого неперспективні напрями виключаються з подальшого розгляду. Наприклад, якщо колір предмета, який ми шукаємо, не червоний, то його безглуздо шукати серед червоних предметів. Альфа-бета алгоритм знайшов широке застосування в основному в системах, орієнтованих на різну гру, наприклад у шахових програмах.

Розбиття на підзадачі. При такій стратегії в початковій задачі виділяються підзадачі, розв'язок яких розглядається як досягнення проміжних цілей на шляху до кінцевої мети. Якщо вдається правильно зрозуміти суть задачі й оптимально розбити її на систему ієрархічно пов'язаних цілей - підцілей, то можна досягти того, що шлях до її вирішення в просторі пошуку буде мінімальний. Однак якщо задача є погано структурованою, то зробити це неможливо. При зведенні задачі до підзадач проводиться дослідження початкової задачі з метою виділення такої множини підзадач, щоб розв'язок якоїсь певної підмножини цих підзадач містив би в собі розв'язок початкової задачі. Кожна з підзадач може бути розв'язана із застосуванням певного методу. До них можуть бути застосовані методи, що використовують простір станів, або ж їх можна проаналізувати з метою виділення для кожної зі своїх підзадач тощо. Якщо продовжити процес розбиття підзадач, що виникають на ще більш дрібні, то зрештою ми перейдемо до деяких елементарних задач, розв'язок яких може вважатися тривіальним. На кожному з етапів може виникнути декілька альтернативних множин підзадач, до яких може бути зведена дана задача. Оскільки деякі з цих множин у кінцевому результаті, можливо, не приведуть до остаточного розв'язку задачі, то, як правило, для розв'язування первинної задачі необхідний пошук у просторі множин підзадач.

Використання формальної логіки при розв'язуванні задач. Часто для розв'язування задач потрібне або проведення логічного аналізу в певному обсязі, або пошук розв'язку, який істотно змінюється після такого аналізу. Іноді такий аналіз показує, що певні проблеми нерозв'язні.

3. ПРЕДСТАВЛЕННЯ ЗАДАЧ У ПРОСТОРІ СТАНІВ

Опис станів. Щоб побудувати опис задачі з використанням простору станів, ми повинні мати певне уявлення про те, що являє собою стан у цій задачі. Але процес розв'язання задачі, у якій розв'язок відшукується без реального переміщення, може спрацювати не за самими конфігураціями, а лише за їх описом. Таким чином, важливим етапом побудови певного опису задачі з використанням простору станів є вибір певної конкретної *форми* опису станів цієї задачі. По суті будь-яка структура величин може бути використана для опису станів. Це можуть бути рядки символів, вектори, двомірні масиви, дерева і списки. Часто обрана форма опису має схожість з певною фізичною властивістю розв'язуваної задачі. Так природною формою опису станів може бути масив 4x4. Вибираючи форму опису станів, треба передбачити, щоб застосування оператора, що перетворює один опис в інший, виявилося б досить легким.

Оператори. Стани і оператори. Прямолінійний підхід пошуку розв'язку полягає в спробі перебрати всілякі, ходи, поки не вдасться отримати цільову конфігурацію. Така спроба по суті пов'язана з пошуком за методом проб і помилок. Передбачаємо, що такий пошук може бути у принципі виконаний на певній обчислювальній машині. Відштовхуючись від початкової конфігурації, можна побудувати всі конфігурації, які виникають унаслідок виконання кожного з можливих ходів, потім побудувати наступну множину конфігурацій після застосування наступного ходу і т.д., поки не буде досягнута цільова конфігурація.

Для обговорення такого різновиду пошуку розв'язку виявляється корисним уведення понять *станів* і *операторів* для даної задачі. Початкова і цільова конфігурації являють собою відповідно початковий і цільовий стани. *Простір* станів, сягаючи від початкового стану, складається з усіх тих конфігурацій, які можуть бути утворені внаслідок допустимих правилами переміщень. Багато задач мають надзвичайно великі (якщо не безкінечні) простори станів.

Оператор перетворює один стан в інший. Мовою станів і операторів розв'язання певної проблеми являє собою послідовність операторів, яка перетворює початковий стан у цільовий.

Простір станів, сягаючи від даного початкового стану, варто уявляти у вигляді графа, вершини якого відповідають цим станам. Вершини такого графа пов'язані між собою дугами, що відповідають операторам.

Стосовно методу розв'язування задач, заснованого на поняттях станів і операторів, то це підхід до задачі з погляду *простору станів.*

Оператори перетворюють один стан в інший. Таким чином, їх можна розглядати як функції, визначені на множині станів і які набувають значення з цієї множини. Оскільки процеси розв'язування задач засновані на роботі з описом станів, то можемо передбачити, що оператори - функції цих описів, а їх значення - нові описи. Загалом можна припускати, що оператори - це *обчислення,* які перетворюють одні описи станів у інші. У всі процедури

дослідження простору станів входить побудова нових описів станів, із подальшою їх перевіркою, з тим щоб пересвідчитися, що вони описують стан, який відповідає поставленій меті. Часто це просто перевірка того, чи відповідає певний опис стану даному цільовому опису стану, але іноді повинна бути зроблена більш складна перевірка. Принаймні, та *властивість,* якій повинен задовольняти опис стану, для того щоб цей стан був цільовим, передбачає вичерпну характеристику.

У деяких задачах оптимізації недостатньо знайти будь-який шлях, що веде до мети, необхідно знайти саме той, який оптимізує певний критерій (зокрема, той, що мінімізує число застосувань операторів). І такими задачами найлегше працювати, зробивши так, щоб пошук не закінчувався доти, поки не буде знайдений певний оптимальний розв'язок. Таким чином, для повного представлення задачі у просторі станів необхідно задати:

а) форму опису станів і, зокрема, опис початкового стану;
б) безліч операторів і їх впливів на описи станів;
в) властивості опису цільового стану.

Простір станів корисно уявляти у вигляді направленого графа.

Запис у вигляді графа. Граф складається з множини вершин (не обов'язково кінцевих). Деякі пари вершин сполучені за допомогою *дуг,* які направлені від одного члена цієї пари до іншого. Такі графи мають назву *направлених графів.* Якщо певна дуга спрямована від вершини *ni* до вершини *nj,* то вершина *nj* є дочірньою для вершини *ni*, а вершина *ni* є батьківською вершиною для *nj*. Може виявитися, що дві вершини будуть дочірніми одна для одної. У цьому випадку пара направлених дуг називається іноді ребром графа. У випадку, коли граф використовується для представлення простору станів, з його вершинами пов'язують опис станів, а з його дугами - оператори.

Послідовність вершин *ni1, ni2,..., nik, у* якій кожна вершина *nij* дочірня дня *ni,* j=2, *k,* називається шляхом довжини *k* від вершини *ni1,* до вершини *nik.* Якщо існує шлях, який веде від вершини *ni* до вершини *nj,* то вершину *nj* називають досяжною з вершини *ni* або нащадком вершини *ni*. У цьому випадку вершина *ni* називається також предком для вершини *nj.* Очевидно, що проблема знаходження послідовності операторів, що перетворюють один стан в інший, еквівалентна задачі пошуку шляху на графі.

4. МЕТОДИ ПОШУКУ У ПРОСТОРІ СТАНІВ

Процеси пошуку на графі. *Граф* характеризується безліччю вершин разом із множиною ребер, причому кожне ребро задається парою *вершин.* Якщо ребра спрямовані, то їх ще називають *дугами.* Дуги задаються *упорядкованими* парами. Такі графи називаються *напрямленими.* Ребрам можна приписувати вартості, імена або мітки довільного вигляду залежно від конкретного додатку.

При формулюванні задачі розв'язок отримується внаслідок застосування операторів до описів станів доти, доки не буде отримано вираз, що описує стан, який відповідає досягненню мети. Усі обговорювані методи перебору

можуть бути змодельовані за допомогою теоретико-графового процесу.

Отже, *початкова вершина* відповідає опису початкового стану. Вершини, які безпосередньо слідують за даною, отримуються внаслідок використання операторів стосовно до опису стану. Нехай Г- якийсь спеціальний оператор, який будує всі вершини, що знаходяться безпосередньо за даною. Нехай процес застосування оператора Г до вершини зветься *розкриттям вершини.*

Від кожної такої наступної вершини до такої, що породжує її, йдуть *покажчики,* які дозволяють знайти зворотний шлях до початкової вершини уже після того як виявлена цільова вершина. Перевіряються вершини, що знаходяться за даною, чи не є вони цільовими. Якщо цільова вершина ще не знайдена, то продовжується процес розкриття вершин (й установлення покажчиків). Коли ж цільова вершина знайдена, ці покажчики переглядаються у зворотному напрямі — від цілі до початку, унаслідок чого простежується процес розв'язування Після цього оператори над описами станів, пов'язаних з дугами цього шляху, утворюють вирішальну послідовність.

Етапи, указані вище, просто описують основні елементи процесу перебору. При повному описі процесу перебору треба ще задати порядок, у якому потрібно розкривати вершини. Якщо вершини розкриваються в тому ж порядку, у якому вони породжуються, то відбувається процес *повного перебору.* Якщо ж спочатку завжди розкривається та вершина, яка була побудована останньою, то виходить процес *перебору в глибину.* Процеси повного перебору в глибину іменуються ще *процедурами сліпого перебору,* оскільки розташування цілі не впливає на порядок розкриття вершин. Однак можливо, що є певна евристична інформація про глобальний характер графа і загальне розташування мети пошуку. Такого роду інформація часто може бути використана для того, щоб "відштовхнути" пошук у бік цілі, розкриваючи, насамперед, найбільш перспективні вершини.

Розглянемо більш детально методи сліпого перебору. *Деревом* називається граф, кожна вершина якого має одну вершину (батьківську), що безпосередньо їй передує, за винятком виокремленої вершини, під назвою *корінь дерева,* яка зовсім не має вершин що їй передували. Отже, корінь дерева служить початковою вершиною. Для перебору дерева простіше за графи передусім тому, що при побудові нової вершини ми можемо бути упевнені, що вона ніколи раніше не будувалася і ніколи не буде побудована знову. Таким чином, шлях від кореня до даної вершини єдиний.

5. МЕТОДИ ПЕРЕБОРУ

Методи повного перебору. У методі повного перебору вершини розкриваються в тому ж порядку, у якому вони будуються. Простий алгоритм повного перебору на дереві складається з такої послідовності кроків:

1) помістити вершину в список, під назвою ВІДКРИТО;
2) якщо список ВІДКРИТО порожній, то на вихід подати сигнал про невдалий пошук, у противному разі перейти до наступного кроку;

3) узяти першу вершину зі списку ВІДКРИТО і перенести її в список ЗАКРИТО назвемо цю вершину n;

4) розкрити вершину n, утворивши всі вершини, які безпосередньо ідуть за n. Якщо таких вершин немає, то відразу ж перейти до кроку (2). Розташувати наступні, що ідуть безпосередньо за n вершини, у кінець списку ВІДКРИТО і побудувати покажчики, які ведуть від них назад до вершини n;

5) якщо ж які-небудь із цих вершин, які знаходяться безпосередньо за n, вершин є цільовими, то на вихід подати рішення, яке отримується поздовжнім переглядом покажчиків, в іншому випадку перейти до кроку (2).

У цьому алгоритмі передбачається, що початкова вершина не відповідає поставленій меті, хоча неважко ввести етап перевірки саме такої можливості. На блок-схемі алгоритму (рис.1) зображено вершини і покажчики, побудовані в процесі перебору, які утворюють піддерево всього неявно визначеного дерева простору станів. Назвемо таке піддерево *деревом перебору*.

У методі повного перебору неодмінно буде знайдений найкоротший шлях до цільової вершини за умови, що такий шлях узагалі існує. (Якщо такого шляху немає, то у вказаному методі буде оголошено про невдачу в разі кінцевих графів, а у разі нескінченних графів алгоритм ніколи не завершить свою роботу).

Метод однакових цін. Часом зустрічаються задачі, у яких до рішення пред'являються дещо інші вимоги, відмінні від вимоги отримання найкоротшої послідовності операторів. Надання дугам дерев певних цін (з подальшим знаходженням вирішального способу, що має мінімальну вартість) відповідає багатьом з таких обіцяних критеріїв. Більш загальний варіант методу повного перебору, так званий *метод однакових цін,* дозволяє в усіх випадках відшукати такий шлях від початкової вершини до цільової, вартість якого мінімальна. У той час як у вище описаному алгоритмі розповсюджуються лінії однакової довжини шляху від початкової вершини, у більш загальному алгоритмі, який буде описаний нижче, розповсюджуються лінії рівноцінності шляху. Передбачається, що задана функція вартості *в(ni, nj),* яка задає вартість переходу від вершини ni до певної вершини nj, яка знаходиться за нею.

У методі однакових цін для кожної вершини n у дереві перебору треба пам'ятати про вартість шляху, побудованого від початкової вершини *s* до вершини *n.* Нехай *g(n)* вартість від вершини *s* до вершини n у дереві перебору. У випадку дерев перебору ми можемо бути упевнені, що *g(n)* є до того ж вартістю того шляху, який має мінімальну *вартість* (оскільки цей шлях єдиний).

У методі однакових цін вершини розкриваються в порядку зростання вартості *g(n).* Цей метод характеризується такою послідовністю кроків:

1) помістити початкову вершину ***s*** у таж званий список ВІДКРИТО. Припустімо, що *g(s)=0;*

2) якщо список ВІДКРИТО порожній, то на вихід подати сигнал про невдалий пошук, в іншому випадку перейти до наступного кроку;

3) узяти зі списку ВІДКРИТО ту вершину, для якої величина ***g*** має найменше значення, і вмістити її в список ЗАКРИТО. Дати цій вершині назву n. (При збігові значень вибирати вершину з мінімальними *g* довільно, але завжди віддавати перевагу цільовій вершині);

4) якщо n є цільова вершина, то на вихід видати вирішальний шлях, що отримується переглядом у зворотному напрямку, орієнтуючись на покажчики; у протилежному разі перейти до наступного кроку;

5) розкрити вершину n, побудувавши всі вершини, які знаходяться безпосередньо за нею. Якщо таких немає, перейти до кроку (2). Для кожної такої безпосередньо наступної (дочірньої) вершини ni *обчислити* вартість g(n), *узявши до уваги, що g(ni)=g(n)+3(n, ni).* Помістити ці вершини разом з відповідними їм щойно знайденими значеннями *g* у список ВІДКРИТО і побудувати покажчики, що йдуть назад до n;

6) перейти до кроку (2).

На блок-схемі алгоритму (рис. 2) показано перевірку того, чи є цільовою вершина, включена в цю схему так, що гарантується виявлення шляхів мінімальної вартості.

Алгоритм, який працює за методом однакових цін, може бути використаний також для пошуку шляхів мінімальної довжини, якщо просто узяти вартість кожного ребра за *одиницю.* Коли є декілька початкових вершин, алгоритм просто модифікується: на кроці (1) усі початкові вершини вміщуються в список ВІДКРИТО. Коли стани відповідають поставленій меті, то вони можуть бути описані явно, бо процес перебору можна спрямувати у зворотному напрямі, узявши цільові вершини за початкові й використовуючи звертання оператора *Г.*

Метод перебору в глибину. У методах перебору в глибину передусім розкриваються ті вершини, які були побудовані останніми. Визначимо глибину вершини на дереві таким чином:

- глибина кореня дерева дорівнює нулю;
- глибина будь-якої подальшої вершини дорівнює одиниці плюс глибина вершини, яка безпосередньо їй передує.

Таким чином, вершиною, що має найбільшу глибину в дереві перебору, є та, яка повинна бути розкрита на даний момент.

Такий підхід може привести до процесу, що розпочинається вздовж якогось безрезультатного напрямку, тому треба ввести певну процедуру повернення. Після того як у ході процесу вибудується вершина з глибиною, що перевищує певну граничну глибину, розкриваємо вершини найбільшої глибини, що не перевищує цієї межі і т.д.

Метод перебору в глибину визначається такою послідовністю кроків (рис.3):

1) помістити початкову вершину в список під назвою ВІДКРИТО;

2) якщо список ВІДКРИТО порожній, то на вихід подати сигнал про невдалий пошук, в іншому випадку перейти до кроку (3);

3) узяти першу вершину зі списку ВІДКРИТО і перенести у список

ЗАКРИТО. Дати цій вершині назву n;

4) якщо глибина вершини n дорівнює граничній глибині, то перейти до (2), в іншому випадку - до (5);

5) розкрити вершину n, побудувавши всі вершини, які знаходяться безпосередньо за нею. Помістити їх у довільному порядку на початок списку ВІДКРИТО і побудувати покажчики, що йдуть від них до n. Якщо одна з цих вершин цільова, то на вихід видати рішення, переглядаючи для цього відповідні покажчики, в іншому випадку перейти до кроку (2).

В алгоритмі пошуку в глибину спочатку відбувається перебір уздовж одного шляху доті, доки не буде досягнута максимальна глибина, а вже потім розглядаються альтернативні шляхи тієї ж або меншої глибини, які відрізняються лише останнім кроком, після чого розглядаються шляхи, що відзначаються останніми двома кроками і т.д.

Зміна при переборі на довільних графах. При переборі на графах (а не на деревах) треба внести певні природні корективи у вказані алгоритми. У простому методі повного переберу не слід вносити ніяких змін, потрібно лише перевіряти, чи не знаходиться щойно побудована вершина в списку ВІДКРИТО або ЗАКРИТО з тієї причини, що вона вже будувалася раніше внаслідок розкриття якоїсь вершини. Якщо це так, то її не треба знову вміщувати у список ВІДКРИТО.

Перш ніж робити будь-які зміни в алгоритмі перебору в глибину, слід вирішити, що розуміти під *глибиною* вершини в графі. Відповідно до звичайних визначень, глибина вершини дорівнює одиниці плюс глибина найбільш близької батьківської вершини, причому передбачається, що глибина початкової вершини дорівнюватиме нулю. Тоді пошук у глибину можна було б отримати, вибираючи для розкриття найглибшу вершину списку ВІДКРИТО (без перевищення граничної глибини). Коли породжуються вершини, що вже є в списку ВІДКРИТО або в списку ЗАКРИТО, перерахунок глибини такої вершини може виявитися необхідним.

Навіть у тому випадку, коли перебір здійснюється на повному графі, безліч вершин і покажчиків, побудованих у процесі перебору, тим не менш, утворюють дерево. (Покажчики вказують лише на одну породжуючу вершину).

Обговорення евристичної інформації. Методи сліпого перебору, повного перебору або пошуку в глибину є вичерпними процедурами пошуку шляхів до цільової вершини: У принципі ці методи забезпечують розв'язування задачі пошуку шляху, але часто подібні методи неможливо використати, оскільки при переборі доведеться розкрити дуже багато вершин, перш ніж потрібний шлях буде знайдено. Оскільки завжди існують практичні обмеження на час обчислення й обсяг пам'яті, то потрібні інші методи, більш ефективні, ніж методи сліпого перебору.

Для багатьох задач можна сформулювати правила, що дозволяють зменшити обсяг перебору. Подібні правила використовуються для прискорення пошуку і залежать від специфічної інформації про задачу, що подається у вигляді графа. Назвемо таку інформацію *евристичною*

інформацією, яка допомагає знайти розв'язок, процедури пошуку, що використовують її, назвемо *евристичними методами пошуку.* Один зі шляхів зменшення перебору полягає у виборі більш "інформованого" оператора Г, який не будує багато вершин, що не стосуються справи. Цей спосіб застосуємо як у методі повного перебору, так і у методі перебору в глибину. Інший шлях полягає у використанні евристичної інформації для модифікації кроку (5) алгоритму перебору в глибину. Замість того, щоб розміщувати заново побудовані вершини в довільному порядку на початку списку ВІДКРИТО, їх можна розташувати в ньому таким певним чином, що вони залежатимуть від евристичної інформації. Так, при переборі в глибину насамперед буде розкриватися та вершина, яка є найкращою.

Більш гнучкий, але і більш витратний шлях використання евристичної інформації полягає в тому, щоб згідно з певним критерієм на кожному кроці переупорядкувати вершини списку ВІДКРИТО. У цьому випадку перебір міг би відбуватися далі на тих ділянках кордону, які вважаються найбільш перспективними. Для того щоб застосувати процедуру упорядкування, нам необхідні заходи, які б дозволяли оцінювати "перспективність" вершин. Такі заходи називають *оцінними функціями.*

Іноді вдається виділити евристичну інформацію (евристику), що зменшує зусилля, які затрачуються на перебір (скажімо, до вершини меншої, ніж при пошуку методом однакових цін), без втрати гарантованої можливості знайти шлях, що складає найменшу вартість. Частіше ж евристики, що використовуються, сильно зменшують обсяг роботи, пов'язаний з перебором, ціною відмови від гарантії знайти шлях найменшої вартості в деяких, або у всіх задачах.

Використання оцінних функцій. Як уже відзначалося, звичайний спосіб використання евристичної інформації пов'язаний зі вживанням упорядкування перебору *оцінних функцій.* Така функція повинна забезпечувати можливість ранжирування вершин-кандидатів на розкриття з тим, щоб виділити ту вершину, яка з найбільшою імовірністю знаходиться на кращому шляху до мети. Оцінні функції будувалися на основі різних міркувань. Робилися спроби визначити *імовірність* того, що вершина розташована на кращому шляху.

Пропонувалося також використати відстань та інші відмінності між довільною вершиною і безліччю цільових вершин.

Передбачимо, що задана певна функція *f,* яка б могла бути використана для упорядкування вершин перед їх розкриттям. Через *f(n)* визначимо значення цієї функції на вершині n. Ця функція збігається з оцінкою вартості того зі шляхів, що йдуть від початкової вершини до цільової, проходять через вершину n. Вартість такого шляху - найменша.

Умовно розташуємо вершини, призначені для розкриття, у порядку зростання їх значень функції *f.* Тоді можна використати певний алгоритм (подібний алгоритму однакових цін), у якому для чергового, розкриття вибирається та вершина списку ВІДКРИТО, для якої значення *f* виявляється

найменшим. Назвемо таку процедуру *алгоритм впорядкованого перебору.*

Для застосування алгоритму впорядкованого перебору на довільних графах (а не тільки на деревах) необхідно передбачити можливість роботи в разі побудови вершин, які вже є, або в списку ВІДКРИТО, або в списку ЗАКРИТО. При використанні деякої довільної функції *f* треба врахувати, що величина *f* для певної вершини зі списку ЗАКРИТО може знизитися, якщо до неї знайдено новий шлях (*f(n)* може залежати від шляху *s* до *n* навіть для вершин зі списку ЗАКРИТО). Отже, потрібно перенести такі вершини назад у список ВІДКРИТО і змінити напрями відповідних покажчиків.

Після вживання цих необхідних заходів алгоритм упорядкованого пошуку може бути представлений такою послідовністю кроків:

1) помістити початкову вершину s у список під назвою ВІДКРИТО і обчислити *f(s)*;

2) якщо список ВІДКРИТО порожній, то на вихід дається сигнал про невдачу; в іншому випадку переходимо до наступного етапу;

3) узяти зі списку ВІДКРИТО ту вершину, для якої *f* має найменше значення, і помістити її в список ЗАКРИТО. Дати цій вершині назву n. У разі збігу значень вибирати вершину з мінімальними *f* довільно, але завжди віддаючи перевагу цільовій вершині;

4) якщо n є цільова вершина, то на вихід видати вирішальний напрямок, що отримується дослідженням відповідних покажчиків; в іншому випадку перейти до наступного кроку;

5) розкрити вершину n, побудувавши всі вершини, які знаходяться безпосередньо за нею. Якщо таких немає, перейти до кроку (2). Для дочірньої вершини *ni* обчислити значення *f(ni);*

6) пов'язати з тами вершинами *ni*, яких ще немає в списках ВІДКРИТО або ЗАКРИТО, щойно прочитані значення *f(ni).* Помістити ці вершини в список ВІДКРИТО і провести від них до вершини n покажчики;

7) пов'язати з вершинами, що знаходяться безпосередньо за n, які вже були в списку ВІДКРИТО або ЗАКРИТО, менші з колишніх або щойно обчислених значень *f*. Помістити у список ВІДКРИТО ті вершини, що знаходяться безпосередньо за n, для яких нове значення *f* виявилося нижчим, і змінити напрям покажчиків від усіх вершин, для яких значення *f* поменшало, спрямувавши їх до n;

8) перейти до (2).

Загальна структура алгоритму ідентична структурі алгоритму однакових цін, тому ми не наводимо для нього блок-схему. Зазначимо, що безліч вершин і покажчиків, які породжуються цим алгоритмом, утворюють дерево (дерево перебору), причому на кінцях цього дерева розташовані вершини зі списку ВІДКРИТО.

Використання інших евристик. Перебір етапами. Використання евристичної інформації може істотно зменшити об'єм перебору, необхідного для пошуку прийнятного шляху. Отже, її використання дозволяє здійснювати перебір на досип великих графах. Проте можуть виникнути випадки; коли

пам'ять виявиться вичерпаною раніше, ніж буде знайдений прийнятний шлях. У таких випадках не варто відмовлятися повністю від продовження перебору, а "відсікти" частину гілок дерева, побудованого до цього моменту в процесі перебору, звільнивши тим самим простір пам'яті, необхідний для поглиблення перебору.

Такий процес перебору може здійснюватися етапами, які відділяються один від одного операціями відсікання дерева, необхідними Для звільнення пам'яті. У кінці кожного етапу утримується певна підмножина відкритих вершин, наприклад вершини з найменшими значеннями *f.* Найдоцільніші шляхи до цих вершин запам'ятовуються, а інша частина дерева відкидається. Потім знову починається перебір, уже від цих "кращих" відкритих вершин. Цей процес продовжується доти, доки або буде знайдена цільова вершина, або будуть вичерпані всі ресурси. Хоч весь процес закінчується побудовою певного шляху, проте немає на даний момент гарантії, що цей шлях буде оптимальним.

Обмеження ряду дочірніх вершин. Інший шлях зменшення перебору, полягає в тому, щоб використати більш інформований оператор *Г,* який породжував би не безліч непотрібних вершин, а лише вершини, розташовані на оптимальному шляху, виключаючи тим самим повністю необхідність перебору.

Один із прийомів, який може дозволити знизити необхідний об'єм перебору, полягає в тому, щоб відразу після розкриття вершини відкинути майже всі дочірні вершини, залишивши лише невелику їх кількість з найменшими значеннями функції *f.* Звичайно може виявитися, що відкинуті вершини розташовані, можливо, до того ж на найкращих напрямках, так що тільки експеримент може визначити придатність такого методу відсікання гілок графа для конкретних задач.

Почергова побудова дочірніх вершин. Коли вершини, які знаходяться безпосередньо за іншими, обчислюються за допомогою операторів у просторі станів, то очевидно, що ці подальші вершини можуть будуватися окремо і незалежно одна від одної. Крім того, існують випадки, коли застосування *всіх* задіяних операторів виявляється дуже марнотратним у розумінні обчислювальних операцій. Як указувалося вище, більш інформований оператор *Г* виділяв би декілька найбільш перспективних операторів і будував би тільки ті подальші вершини, які виникають унаслідок їх застосування. Більш гнучкий підхід полягає в тому, щоб спочатку допускати застосування найперспективнішого оператора (що приведе до однієї з подальших вершин), залишаючи надалі можливість у процесі перебору побудувати й інші вершини, які знаходяться безпосередньо за даною. Для того щоб скористатися цією ідеєю, разом з оцінними функціями при упорядкуванні вершин в алгоритм упорядкованого перебору потрібно внести відповідні зміни.

6. СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ УПРАВЛІННЯ ФАКТАМИ В БАЗІ ЗНАНЬ

Мета завдання: створити механізм для введення фактів і правил у базу знань, який підтримуватиме можливість управління інформацією.

Представлення фактів у базі знань. У базі знань представляються факти, які визначають **об'єкти,** описують їх **атрибути** і надають їм певні **значення**. В експертній системі під словом *«об'єкт»* мають на увазі фізичні предмети (наприклад, «термометр») і загальні уявлення (наприклад, «жар»). З об'єктами пов'язуються атрибути, які описують їх з погляду користувача. Так, атрибутами об'єкта термометр можуть бути «скляний», «градуйований». Значення об'єкта задає точність, наприклад, значення температури може бути 36,6°С.

Пари «об'єкт - значення». Для того, щоб упорядкувати вирази фактів, їх можна об'єднати в **пари «об'єкт - значення»,** з'єднавши ім'я об'єкта з ім'ям атрибута. Наприклад, у пропозиції «термометр - температура - висока» ім'ям об'єкта буде слово «термометр», його атрибутом - «температура», а його значенням - слово «висока». Якщо представити цю пропозицію у вигляді пари, тоді об'єктом буде «термометр - температура», а значенням - «висока».

Список об'єктів. Для представлення об'єктів у базі знань використовується структура даних під назвою **зчеплений список.** Кожна одиниця цього списку іменується **вузлом** і містить поля, у які заноситься інформація про об'єкт. Одне з подів служить покажчиком, що повідомляє системі, де шукати наступний зчеплений вузол списку. Останній вузол указує на **NIL.** Це означає, що список вузлів вичерпаний.

Крім того, кожний вузол у списку об'єктів має другий покажчик, який визначає початок списку значень, пов'язаних з ім'ям об'єкта. Цей внутрішній список називається списком значень об'єктів.

Визначення констант для представлення фактів» у базі знань. З метою обмеження дюжини рядка для запису всіх об'єктів вводиться константа:

WORD_MAX = 40

Таким же чином установлюється максимальна довжина рядка у 80 символів:

LINE_MAX=80

У модулях програми є необхідність уведення таких констант «:», «.», «,», «=», «».

Визначається тип рядків, до яких належить константа. Установлюється максимальна довжина імені об'єкта і значення, потім обмежується довжина рядка, що вводиться:

word_string = string[WORD_MAX];

line_string = string[LINE_MAX]

Управління фактами в базі знань. Експертна система повинна мати механізм для введення фактів і правил у базу знань, підтримки набору виразів у базі знань і виведення бази знань для перегляду користувачами.

При розробці й підтримці списку зчеплених об'єктів потрібен механізм для додавання до вже існуючого списку нових вузлів, засобів уведення імен нових об'єктів і значень, способу знаходження і витягування конкретних імен для перегляду. Крім того, корисно перевірити, чи має об'єкт певне значення і чи істинний окремий факт.

Створення вузла в списку об'єктів. Створити процедуру (MAKE_NODE), яка додає новий вузол у вершину списку об'єктів і встановлює покажчик на об'єкт пойменований у цьому вузлі.

Вміщення імені об'єкта у зчеплений список. Створити процедуру (FIND_OBJECT), яка шукає ім'я об'єкта в списку. Якщо ім'я знайдене, покажчик установлюється на його місце в списку об'єктів, в іншому випадку він встановлюється на NIL.

Розщеплення пари «об'єкт - значення». Створити процедуру *(SPLIT),* яка повинна відрізняти ім'я об'єкта від імені значення в парі **«об'єкт — значення».** Процедура відшукує в рядку, що вводиться, позицію символа «=»; потім ім'я об'єкта й ім'я значення виводяться в окремі рядки.

Перевірка об'єктів і значень. Створити процедуру (TEST), яка перевіряє пару ***"об'єкт—значення"*** на наявність її в базі знань. Програма встановлює, чи містить вона ім'я обумовленого об'єкта в списку об'єктів, а потім шукає ім'я значення. При написанні цієї процедури використовується процедура FIND_OBJECT.

Додавання об'єкта до списку. Створити процедуру (ADD_OBJECT), яка передає два рядки: ім'я об'єкти й ім'я значення. Ця процедура додає ім'я об'єкта до списку зчеплених об'єктів і вставляє ім'я значення у відповідний список значень вузла об'єкта.

На першому етапі перевіряється наявність імені об'єкта в списку об'єктів. Якщо ім'я не знайдене, програма створює новий вузол і додає вказане ім'я об'єкта у вершину списка. На наступних кроках здійснюється пошук у списку значень і, якщо ім'я значення в ньому відсутнє, воно додається у вершину списка.

Виведення списку значень. Експертна система повинна володіти можливістю виведення повного списка всіх значень і об'єктів бази знань.

Створити процедуру (SEE_VALS), яка за допомогою покажчика об'єкта виводить на екран дисплея всі імена значень, що містяться в списку значень вузла об'єкта. Спочатку процедура знаходить вказаний вузол об'єкта, а потім перебирає всі імена в списку значень. По команді write програма виводить ім'я поточного значення на екран, а потім звертається до покажчика для визначення наступного елемента. Коли знайдено значення NIL, виведення припиняється.

Виведення фактів бази знань. Створити процедуру (SEE_OBJECT), яка буде використовуватися для виведення на екран імен усіх об'єктів і значень, уведених на даний момент у базу знань. Процедура друкує заголовок, а потім сортує імена об'єктів у списку об'єктів, виводячи всі імена й відповідні їм значення доти, доки не буде досягнуто кінця списку NIL.

7. ПИТАННЯ І ДОЗВОЛЕНІ ЗНАЧЕННЯ

Створимо механізми для введення та обробки питань і дозволених значень. Для цього: а) створимо процедури для введення дозволених значень і питань; б) поставимо питання (згідно таблиці) і введення в базу знань;

в) введемо вирішені значення об'єкта.

Дозволені значення. Експертна система задає користувачеві питання, що стосуються об'єктів, наприклад, «яка температура у пацієнта» і пропонує можливі варіанти відповідей-«висока, нормальна, знижена». Відповіді на питання і будуть **вирішеними значеннями** об'єкта. Для створення повноцінної експертної системи потрібний механізм прочитання імен дозволених значень з рядка, що вводиться, і занесення їх у зчеплений список.

Питання про** вирішені **значення. Програма повинна мати можливість задавати питання, що стосуються певного об'єкта. Коли користувач уводить дозволені значення об'єкта, кожне пойменоване значення стає однією з можливостей вибору для відповіді на питання. Спочатку користувач уводить ім'я об'єкта разом зі списком пов'язаних з ним дозволених значень. Це введення схоже на пару ***об'єкт - значення*** з тією лише різницею, що права частина рівняння може містити декілька імен значень.

Далі необхідно створити процедури, що дозволяють експертній системі сприйняти введення оператора, що містить ім'я об'єкта і список дозволених значень, і приєднати імена цих значень до зчепленого списку об'єкта. Крім того, програма повинна бути забезпечена пристроєм для додавання питань про певні об'єкти і виведення існуючих з відповідними дозволеними значеннями.

Визначення розташування дозволеного значення в рядку. Функція FIND_WORD повинна визначити розташування n-го значення в рядку вигляду об'єкт = значення 1, значення 2, ..., значення n і запам'ятати знайдене значення в змінної WORD. Якщо n-е значення не знайдене, функція набуває значення FALSE.

Ця функція спочатку визначає номер імені дозволеного значення, що міститься в рядку, потім визначає позицію наступної коми в списку дозволених значень. Якщо число значень менше n, то програма видає значення FALSE. Визначивши n-е дозволене значення, програма видає TRUE і копіює ім'я значення в рядкову змінну WORD.

Додавання вузла до списку дозволених значень. Модуль ADD_LEGAL додає новий вузол у вершину списку дозволених значень об'єкта. Спочатку процедура створює новий вузол для імені значення. Потім перевизначає вершину списку і встановлює покажчик на цей вузол. І нарешті, вона заносить у новий вузол посилання на вершину списку.

Виклик значень. Модуль FIND_LEGAL застосовується для витягання імені значення після знаходження дозволеного значення, яке відповідає змінній num у списку дозволених значень об'єкта.

Оголошення змінних визначає покажчики, які відстежують список об'єктів і список дозволених значень. Крім того, створюється лічильник, що набуває цілих значень.

Функція FIND_LEGAL спочатку розщеплює рядок, що містять ім'я об'єкта і список дозволених значень. Потім перевіряє на наявність об'єкта в списку об'єктів. Якщо ім'я об'єкта знайдене, функція набуває значення TRUE.

Далі програма встановлює значення лічильника на одиницю і переглядає список дозволених значень об'єкта доти, доки не виявить заданий елемент. Якщо кінець списку досягається раніше, ніж знайдене певне дозволене значення, видається FALSE.

Додавання дозволених значень. Процедура MAKE_LEGAL впливає на рядок вигляду об'єкт = значення1, значення 2, .., значення n., вибираючи кожне значення в рядку і додаючи його до списку дозволених значень об'єкт

Ця процедура спочатку розщеплює введений рядок для розподілу імені об'єкта й імені дозволених значень, потім переглядає список об'єктів у пошуках певного об'єкта Якщо ім'я об'єкта не знайдене, то процедура викликає процедуру MAKE_NODE для того, щоб створити новий вузол, а потім уміщує ім'я нового об'єкта у вершину списка. Імена значень одне за іншим додаються до списку дозволених значень; при кожному додаванні значення лічильника збільшується на одиницю.

Додавання питань. Процедура ADD_QUESTION впливає на введений рядок вигляду об'єкт = питання, вибираючи питання, і вміщуючи його в зчеплений список об'єктів.

Спочатку процедура розщеплює рядок, відділяючи ім'я об'єкта від питання. Потім вона шукає ім'я об'єкта в списку об'єктів, якщо об'єкт не знайдений, створює новий вузол і даний об'єкт вміщує у вершину списка. Далі програма додає питання до об'єкта.

Виведення питання на екран. Процедура P_QUESTION виводить на екран текст питання, що задається програмою оператору. Якщо текст питання не заданий, він буде формуватися автоматично у вигляді «Яке значення (об'єкт)?».

Відповідь на питання. Процедура ASK дозволяє оператору відповісти на питання, задане програмою на основі дозволених значень і попередньо введеного тексту. Модуль ASK розпізнає введене ім'я об'єкта і витягує відповідний текст питання. Зі списку дозволених значень об'єкта формується меню, і програма чекає, поки користувач зробить вибір.

8. СТВОРЕННЯ БАЗИ ПРАВИЛ

Мета завдання: створити механізми для введення правил у базу правил системи. Для цього пропонується:

а) розробити правила відповідно до таблиці;

б) спорити модулі для додавання до поточного правила передумови, читання правил із бази правил, виведення названого правила на екран дисплея;

в) створити файл правил.

Правила експертної системи. Експертна система обробляє символічне представлення реальності, вдаючись до правил, звичайно із залученням методу зворотного ланцюжка. У цьому методі консультація починається з визначення конкретної мети або кінцевого результату. Цей підхід принципово відрізняється від прямого ланцюжка, коли оператор починає з визначення симптом чи проблеми, а не з цілі.

Правило складається з двох частик: передумови і висновку, що є фактами бази знань, вираженими парами «об'єкт-значення». В експертній системі правила мають такий формат:

Якщо **ПЕРЕДУМОВА,** то **ВИСНОВОК.**

У кожному випадку ми сприймаємо такий факт: коли вірна передумова, то вірний і висновок. Ці прості відносини ***«якщо - то»*** представляють вузли розв'язку, за якими машина висновку просувається до поставленої мети.

Правило може містити булевий оператор « і » для утворення таких виразів, як ***«якщо А і якщо В, то С»***, оскільки не всі розв'язки лінійні.

Для введення правил в експертну систему користувач спочатку записує їх у текстовий файл за допомогою будь-якого редактора, а потім система включає ці правила у свою базу знань.

Допоміжна процедура для перетворення питань, що вводяться. Функція P_READ упаковує рядки в стандартний формат, виключаючи пропуски і перетворюючи всі букви в рядкові. Пропуски, що стоять між словами в питанні, будуть залишені для чіткості й ясності

Додавання передумов. Модуль ADD_PREM додає до поточного правила передумову. Кожне правило має список передумов, що містить передумови цього правила, і список висновків. Процедура розщеплює початковий рядок для видобування передумови, вираженої парою «об'єкт - значення». Потім програма створює новий вузол списку передумов правила, установлює покажчик правила на відповідне ім'я об'єкта й ім'я значення, далі до списку додається новий вузол.

Додавання висновку. За допомогою цієї функції до поточного правила додається висновок. Модуль ADD_CON розщеплює рядок, що вводиться, виділяючи висновок у формі «об'єкт - значення». Потім програма формує новий вузол у вершині списку висновків правила і додає до нього новий висновок.

Виведення правила на екран. Цей модуль дозволяє програмі на запит виводити існуюче правило на екран. P_RULE обробляє покажчик правила, знаходить правило і виводить на екран дисплея його текст.

Додавання правил у базу правил. Ця процедура дозволяє програмі вичитати з текстового файла одне правило і додати його до списку правил. Компоненти правила - передумова і висновок - обробляються роздільно. Процедура ENTER_RULE починається операторами, що створюють новий вузол у вершині списку правил. Потім програма вичитує з текстового файла передумову і додає її до списку. Після цього вона вичитує висновок і додає його у вузол.

Читання файла правил. Програмі надається можливість вичитати весь файл правил у свою базу знань. Процедура READ_FILE читає файл, що містить підготовлені користувачем правила, і обробляє команди, що зустрічаються для додавання до імені об'єкта дозволених значень або додавання питань. Потім вона викликає процедуру ENTER_RULE для занесення кожного правила у відповідний список правил.

Створення файла правил. Для створення файла правил можна використати будь-який текстовий редактор, який створює файли в ASCII - коді. При введенні кожного правила необхідно дотримуватися такого правила синтаксису:

правило (n): якщо **ПЕРЕДУМОВА, то ВИСНОВОК.**

9. СТВОРЕННЯ МЕХАНІЗМІВ ВИВЕДЕННЯ РЕЗУЛЬТАТІВ ЕКСПЕРТИЗИ

***Мета завдання*:** створити механізми для виведення результатів експертизи. Для цього:

а) створити модулі для обробки правил, пошуку правил, за якими видається, висновок експертної системи, виведенні на екран результатів консультації; б) провести експертизи.

Механізми виведення. База знань представляє опис предметної сфери експертної системи. Машина висновку є інтерпретатором правил, який використовує факти цієї бази знань для вирішення поставлених проблем. Вона здійснює це формулюванням пробних гіпотез і перевіркою їх на відповідність указаній цілі. Оператор задає мету консультації у вигляді імені об'єкта. Механізми виведення використовує набір правил, намагаючись набути значення вказаного об'єкта - цілі. Програма продовжує пошук, доки одне з передбачуваних рішень не виявиться вірним. У системах, що мають зворотний ланцюжок, виведення починається з кінцевого висновку. Програма перебирає список правил у пошуках такого, яке містить об'єкт - ціль у своїй правій частині. Потім машина перевіряє кожну частину передумови правила і процес починається знову.

Пошук відповідного правила. Функції FIND_RULE відшукує правило, що містить у правій частині ім'я заданого об'єкта. Вона застосовує покажчик висновку для перегляду списку правил до збігу імені об'єкта із заданим. Якщо відповідний об'єкт знайдено, функція набуває значення TRUE.

Отримання висновку. Модуль CONCLUDE являє собою процедуру для здійснення висновку, коли машина виведення знаходить зв'язок між передумовою одного правила і висновком іншого.

Результуючі факти. Процедура PURSUE обробляє отримане ім'я об'єкта, намагаючись застосовувати правила для присвоєння йому значення. Якщо жодне правило не дозволяє зробити висновок про об'єкт, то його значення запитується у користувача

Для фіксації поточного правила, що розглядається на даний момент, процедура використовує покажчик правил. Робота починається з перегляду списку об'єктів для перевірки наявності відшукуваного об'єкта. Якщо об'єкт не знайдено, у вершині списку створюється новий вузол. Потім програма відшукує правило, що містить у правій частині ім'я цього об'єкта. Далі перевіряється кожна складова передумови правила Якщо правило виконується, його висновок додається в базу знань. Якщо ж значення об'єкта за допомогою правил знайти не вдається, програма пропонує оператору

звести ім'я цього значення.

Друк результатів консультації. Процедура P_RESULT дозволяє системі виводити на екран результати консультації. З основної програми цій процедурі передається ім'я об'єкта - цілі. Вона роздруковує на екран дисплея список усіх значень цього об'єкта.

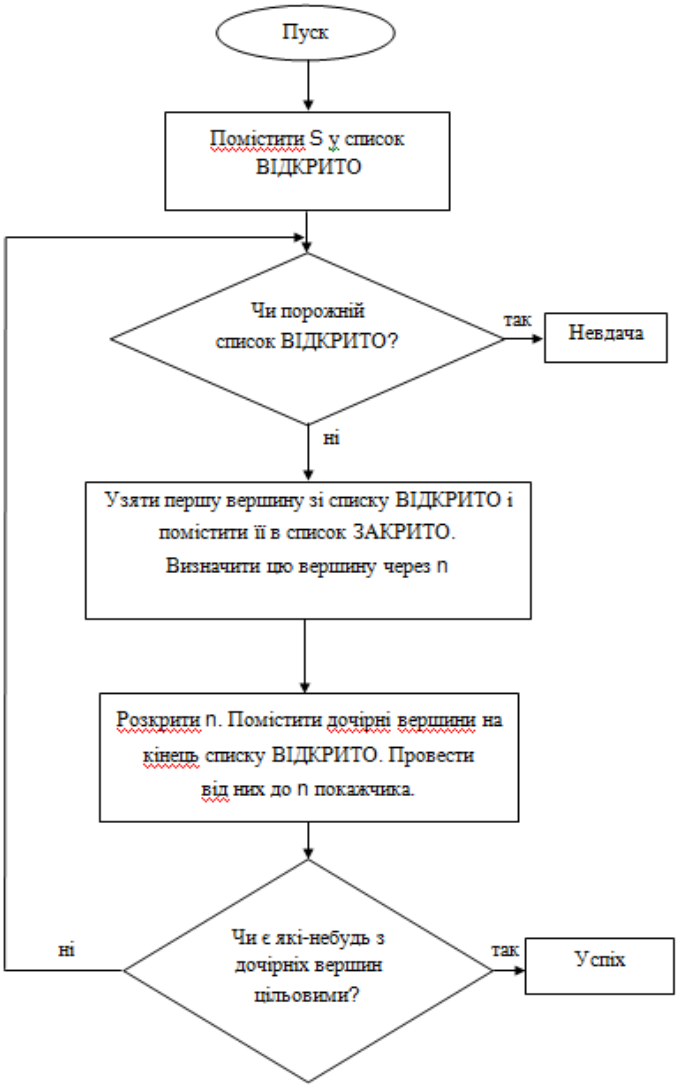

Рис. 1 – Блок-схема програми алгоритму для дерева повного перебору

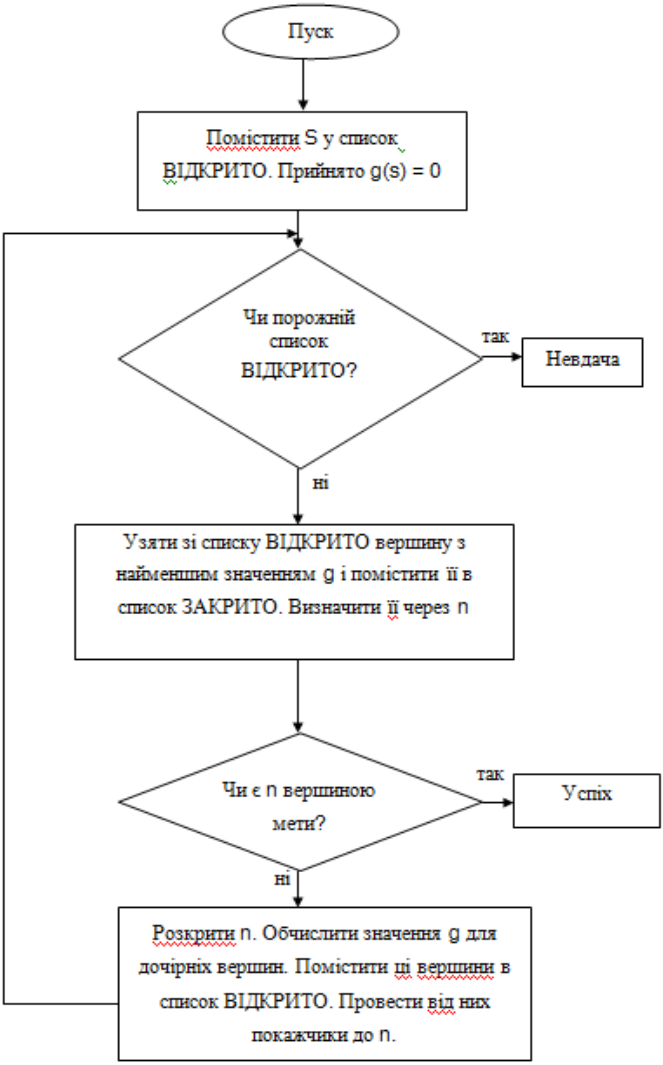

Рис. 2 – Блок-схема програми алгоритму однакових цін для дерев

1. *Азарсков, Сущенко О.А.* Експериментальні випробування та дослідження систем, К.: НАУ, 2003, 268с.
2. *Башлыков А.А.* Проектирование систем принятия решений в энергетике. – М.: Энергоатомиздат, 1986. – 120 с.
3. *Бєляєв Ю.Б., Шмельова Т.Ф., Сікірда Ю.В.* Моделювання процесу прийняття рішень оператором авіаційної ергатичної системи в особливих випадках польоту / Автоматизація виробничих процесів. – 2003. - №2(17). – с. 17-23.

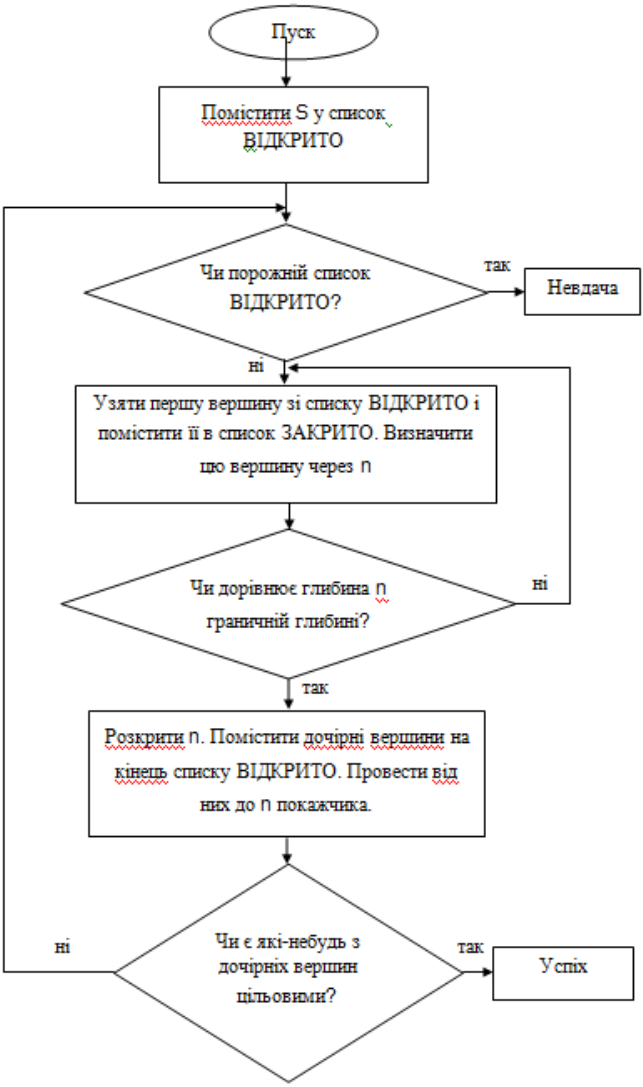

Рис.3 – Блок-схема програми алгоритму пошуку в глибину для дерев перебору

4. *Борисов А.Н., Алексеев А.В., Крумберг О.А. и др.* Модели принятия решений на основе лингвистической переменной. – Рига.: Зинатне, 1982. – 256 с.
5. *Гаврилова Т.А., Хорошевский В.Ф.* Базы знаний интеллектуальных систем. – СПб: Питер, 2001. – 384 с.
6. *Герасимов Б.М., Дивизинюк М.М., Субач И.Ю.* Системы поддержки принятия решений : проектирование, применение, оценка эффективности, Севастополь, СНИЯЭ и П,2004,320с.

7. *Заде Л.* Понятие лингвистической переменной и ее применение к принятию приближенных решений. – М.: Мир, 1976. – 167 с.
8. *Козелков С.В., Машков О.А., Барабаш О.В.* Методика побудови функціонально – стійких розподілених інформаційних систем / Матеріали науково – технічної конференції „Сучасний стан та проблеми авіаційної техніки Військово – Повітряних Сил", НЦ ВПС, 10-11 червня 2004р.
9. *Козелков С.В., Машков О.А., Барабаш О.В.* Побудова функціонально-стійких розподілених інформаційних систем наземного базування / Тез. доп. XIV науково-технічної конференції, ЖВІРЕ ім. С.П.Корольова, Житомир, 2004, с. 107-108.
10. *Машков О.А.* Концепции построения функционально-устойчивых информационно-управляющих комплексов / Тезисы докладов 6-й Всесоюзной конференции. Ч.II. – К.: АН УССР, 1991, с. 50-51.
11. *Самохвалов Ю.Я.* Декомпозиция логико-лингвистических моделей принятия решений в распределенной вычислительной среде // Кибернетика и системный анализ – 1997. - № 1. – С. 57-65.
12. *Субач І.Ю., Соколов В.В.* Організація баз даних та знань. – К: КВІУЗ, 1999.
13. *Трахтенгерц Э.А.* Компьютерная поддержка принятия решений. – М.: СИНТЕГ, 1998. – 376 с.
14. *Уотермен Д.* Руководство по экспертным системам. Пер. с англ. – М.: Мир, 1989. – 388 с.

Поступила 20.10.2010р.

УДК 683.03

О.Ю.Афанасьєва, Б.В.Дурняк, Ю.М.Коростіль, В.І.Сабат

АНАЛІЗ ПРИЧИН ВИНИКНЕННЯ НЕПРОЕКТНИХ НЕСПРАВНОСТЕЙ В СКЛАДНИХ ТЕХНІЧНИХ ОБ'ЄКТАХ

Задача діагностування несправностей являється ключовою для проблем забезпечення безпеки функціонування складних технічних об'єктів. Традиційний підхід до розв'язку цієї задачі полягає у виявленні несправностей, що зароджуються, на основі аналізу діагностичних ознак. Це означає, що такі діагностичні ознаки ζ_{ij} повинні бути відомими діагностичній системі. Діагностичний параметр ξ_{ij} відрізняється від ζ_{ij} тим, що ξ_{ij} характеризує несправність, яка уже виникла. Принципова різниця між несправностями та зародженням несправності полягає у наступному. Несправність представляє деяку подію, яка приводить до наступних можливих наслідків:

- обмеження функціональних можливостей технічного об'єкту (*ТО*),
- до розвитку несправності, що може привести до виникнення аварії, або

