НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

"На правах рукопису"
УДК 004.032.26

«До захисту допущено»
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Магістерська дисертація

зі спеціальності 121 Інженерія програмного забезпечення
за спеціалізацією Програмне забезпечення розподілених систем
на тему Автоматичне розв’язання логічних задач_____
_____
_____

Виконав: студент _6_курсу, групи_ТВ-71мн__
_____Бараніченко Олексій Миколайович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, доцент, к.т.н. Шаповалова С.І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент ст. викладач кафедри АЕС та ІТФ,
к.т.н, Кондратюк В. А. _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2019

Національний технічний університет України
“ Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти другий, магістерський

зі спеціальності - 121 Інженерія програмного забезпечення
за спеціалізацією - Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
Коваль О.В. _____
(прізвище, ініціали) (підпис)

«____» _____ 2019р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Бараніченку Олексію Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема дисертації Автоматичне розв’язання логічних задач
Науковий керівник Шаповалова Світлана Ігорівна, к.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від _____ 20__ року №_____
2. Строк подання студентом дисертації _____
3. Об’єкт дослідження обчислювальні механізми розв’язання логічних задач
4. Предмет дослідження обчислювальний механізм розв’язання логічних комбінаторних головоломок
5. Перелік питань, які потрібно розробити провести аналіз існуючих методів розв’язання логічних задач, виконати формалізацію логічних задач, створити обчислювальний механізм для розв’язання логічних задач, розробити алгоритм його навчання, розробити програмне забезпечення з розв’язання логічних задач, провести обчислювальні експерименти.
6. Орієнтовний перелік ілюстративного матеріалу існуючі методи розв’язання логічних задач, формалізація логічних задач, структура мережі зв’язків, приклади (n-1)-мірних шарів мережі, алгоритм навчання, інтерфейс програмного забезпечення, обчислювальні експерименти.

7. Орієнтований перелік публікацій Шаповалова С.І., Бараніченко О.М., «Машинне навчання для розв'язання логічних головоломок», Шаповалова С.І., Бараніченко О.М., «Розв'язання логічних задач нейронними мережами», Шаповалова С.І., Бараніченко О.М., «Встановлення зв'язків між об'єктами логічної задачі», Шаповалова С.І., Бараніченко О.М., «Вдосконалення CAM-систем для невеликих виробництв»

8. Дата видачі завдання «_20_» _вересня__ _2017_ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	20.09.17 р.	
2	Збір інформації	21.09.17р. – 21.01.18р.	
3	Аналіз вимог завдання, розробка методів і засобів розв'язання поставленої задачі	22.01.18р – 27.01.19р	
4	Розробка та тестування програмного продукту	28.01.19р. – 03.03.19р.	
5	Підготовка матеріалів магістерської роботи	03.03.19р. – 29.04.19р.	
6	Написання основних розділів автореферату	29.04.19р. – 10.05.19р.	
7	Захист програмного продукту	11.03.19р	
8	Передзахист	14.05.19р	
9	Захист	20.05.19р	

Студент _____ Бараніченко О. М.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Шаповалова С. І.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Структура та обсяг дипломної роботи. Магістерська дисертація складається зі вступу, чотирьох розділів, висновку, переліку посилань з 33 найменувань, двох додатків та містить 41 рисунок, 4 таблиці. Повний обсяг магістерської дисертації складає 105 сторінок, з яких перелік посилань займає 3 сторінки, додатки – 30 сторінок.

Актуальність теми. В різних сферах виробництва та науки часто виникає необхідність розв’язання логічних задач. Однак «ручні» методи не підходять для великих та складних задач, які найчастіше виникають на практиці. Існуючі комп’ютерні методи є більш ефективними. Однак вони часто є складними в реалізації та використанні. Саме тому задача створення універсального обчислювального алгоритму та його програмна реалізація є актуальними і мають практичне значення.

Мета роботи. Розробити метод автоматичного розв’язання логічних задач на основі машинного навчання.

Завдання дослідження:

1. Провести аналіз логічних задач та існуючих методів їх класифікації та формалізації.
2. Провести аналіз існуючих методів розв’язання логічних задач.
3. Створити спеціальну обчислювальну структуру для розв’язання логічних задач.
4. Розробити алгоритми побудови та навчання створеної структури.
5. Розробити прикладне програмне забезпечення з розв’язання логічних задач.
6. Провести обчислювальні експерименти для розробленого методу розв’язання логічних задач.

Об’єкт дослідження. Обчислювальні механізми розв’язання логічних задач.

Предмет дослідження. Обчислювальний механізм розв’язання логічних комбінаторних головоломок.

Наукова новизна:

1. Запропоновано метод автоматичного розв’язання логічних задач на основі навчання спеціальної обчислювальної структури – мережі зв’язків – для підвищення ефективності логічного висновування.

2. Набув подальшого розвитку підхід до розв’язання логічних задач на основі машинного навчання, який враховує умови зв’язку між значеннями властивостей задачі.

Апробація результатів дисертації. Результати дисертації було представлено на XVI та XVII міжнародних науково-практичних конференціях аспірантів, магістрантів, студентів «Сучасні проблеми наукового забезпечення енергетики» (2018, 2019) та V науково-практичній дистанційній коференції молодих вчених і фахівців з розробки програмного забезпечення «Сучасні аспекти розробки програмного забезпечення».

Публікації:

1. Бараніченко О. М. Розв'язання логічних задач нейронними мережами / О. М. Бараніченко, С. І. Шаповалова. // Політехніка. – 2018. – №2. – С. 186–187.

2. Бараніченко О. М. Машинне навчання для розв'язання логічних головоломок. / О. М. Бараніченко, С. І. Шаповалова. // Політехніка. – 2019. – №2. – С. 93–94.

3. Шаповалова С. І. Встановлення зв'язків між об'єктами логічної задачі / С. І. Шаповалова, О. М. Бараніченко. // Черкаси: видавець Чабаненко Ю. А. – 2018. – С. 204–209.

4. Шаповалова С. І. Вдосконалення CAM-систем для невеликих виробництв / С. І. Шаповалова, О. М. Бараніченко. // Міжвідомчий науково-технічний збірник «Адаптивні системи автоматичного управління». – 2017. – №1. – С. 189.

Ключові слова. Логічна задача, zebra puzzle, машинне навчання, мережа зв’язків.

ABSTRACT

Structure and volume of the thesis. The master's thesis consists of an introduction, four sections, a conclusion, a list of references with 33 titles, two annexes and contains 41 figures, 4 tables. The volume of the master's thesis is 105 pages, of which the list of links takes 3 pages, applications – 30 pages.

Actuality of theme. In various fields of production and science, it is often necessary to solve logical tasks. However, manual methods are not suitable for large and complex tasks that often occur in practice. Existing computer methods are more effective. However, they are often difficult to implement and use. That is why the task of creating a universal computing algorithm and its software implementation are relevant and has practical importance.

The goal of the work. Develop a method for automatic solution of logical tasks based on machine learning.

Research tasks:

1. Conduct an analysis of logical tasks and existing methods of their classification and formalization.

2. Conduct an analysis of existing methods of solving logical tasks.

3. Creation a special computing structure for solving logical tasks.

4. Develop algorithms for constructing and training the created structure.

5. Develop application software for solving logical tasks.

6. Conduct computational experiments for the developed method of solving logical tasks.

Object of research. Computational mechanisms for solving logical tasks.

Subject of research. Computational mechanism for solving logical combinatorial puzzles.

Scientific novelty:

1. The method of automatic solving of logical tasks on the basis of training of a special computer structure - a network of links - is proposed for increasing the efficiency of logical conclusion.

2. A further development approach to the solution of logical tasks based on machine learning, which takes into account the connection conditions between the values of the properties of the problem.

Approbation of the results of the thesis. The results of the thesis were presented at the XVI, XVII International Scientific and Practical Conference of Postgraduate Students, Graduates, Students "Modern Problems of Scientific Supply of Energy" (2018, 2019) and V scientific and practical remote conference of young scientists and specialists in software development "Modern aspects of software development" (2018).

Publications:

1. Baranichenko O. M. Solving logical tasks by neural networks / O. M. Baranichenko, S. I. Shapovalova. // Politechnika – 2018 – №2. – P. 186-187.

2. Baranichenko O. M. Machine learning for solving logical puzzles. / O. M. Baranichenko, S. I. Shapovalova. // Politechnika – 2019 – №2. – P. 93-94.

3. Shapovalova SI, Establishing the relationship between objects of a logical task / S. I. Shapovalova, O. M. Baranichenko. // Cherkasy: publisher Chabanenko Yu. A. –2018. – P. 204-209.

4. Shapovalova S.I. Improvement of CAM-systems for small production / SI Shapovalova, O. M. Baranichenko. // Interdepartmental scientific and technical collection "Adaptive systems of automatic control". – 2017 – №1. – P. 189.

Keywords. Logical task, zebra puzzle, machine learning, network of links.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Nv	- Node value
V	- Value
C	- Combination
WPF	- Windows Presentation Foundation
XML	- eXtensible Markup Language

ВСТУП

На сьогоднішній день існує велика кількість задач які потребують для розв’язання логічних послідовних роздумів та висновків. Такі задачі називаються логічними задачами. Д о них, наприклад, відносять «Загадку Ейнштейна», в якій необхідно за наявними початковими умовами, шляхом логічного доповнення, знайти розв’язок. В житті подібні задачі можуть мати місце, наприклад, для розподілу навантаження на обладнання, приміщення, персонал на виробництвах, у логістиці, в різноманітних експертних системах тощо. Однак, на сьогоднішній день відсутні програмні рішення розв’язання таких задач, які є простими у створенні, використанні та підтримці, мають високу ефективність, забезпечують можливість розширення чи часткової заміни задачі. Це, перш за все, пов’язано з відсутністю ефективних уніфікованих алгоритмів. Складність використання існуючих алгоритмів полягає у тому, що вони або потребують індивідуального програмного налаштування під конкретну задачу, або мають низьку швидкість роботи. Саме тому задача створення універсального методу розв’язання логічних задач та його реалізація для подальшого оброблення комп’ютерною технікою є актуальною, має наукове та прикладне значення і потребує ґрунтовного дослідження.

Мета роботи: розробка методу автоматичного розв’язання логічних задач на основі машинного навчання.

Об’єкт дослідження: обчислювальні механізми розв’язання логічних задач.

Предмет дослідження: обчислювальний механізм розв’язання логічних комбінаторних головоломок.

До роботи ставляться такі задачі:

1. Провести аналіз логічних задач.

2. Виокремити задачі, які є найбільш повними для подальшого використання їх у тестуванні системи.

3. Виконати огляд існуючих методів розв’язання задач визначеного класу.

4. Провести аналіз «ручних» та машинних методів, порівняти їх.

5. Розробити універсальний продуктивний метод розв’язання логічних задач.

6. Навести алгоритми створення обчислювальної структури та її навчання.

7. Створити програмну реалізацію розробленого методу. Розроблена система повинна бути реалізована в вигляді як кінцевого програмного забезпечення, так і модулю для вбудовування в інші системи.

8. Провести обчислювальні експерименти для створеного програмного забезпечення, порівняти результати з результатами роботи існуючих методів. Навести результати роботи.

9. Вбудувати розроблений механізм розв’язання логічних задач в систему PowerCAM.

1. ПІДХОДИ ДО РОЗВ'ЯЗАННЯ ЛОГІЧНИХ ЗАДАЧ

У першому підрозділі описані логічні задачі. Під час опрацювання джерел було виокремлено найбільш повну їх класифікацію та визначено набори тестових даних. У другому підрозділі описані існуючі методи розв'язання логічних задач. Було розглянуто як «ручні», так і автоматичні методи.

1.1. Класифікація логічних задач

Логічна задача – це задача з області математичної дедукції, яка для розв'язання потребує послідовного виконання логічних дій. Історично, створювачем першої описаної логічної головоломки вважають Чарльза Лутвіджа, який навів її формулювання у книзі «Гра логіки» [1]. Однак подібні задачі виникали задовго до цього, оскільки вони мають безпосередній зв'язок з реальним світом. Це, наприклад, стародавня задача переливання води, в якій при наявності двох ємностей об'ємом 3 і 5 літрів необхідно набрати 4 літри води. На сьогоднішній день існує велика кількість логічних задач. Це, наприклад, задачі кубика Рубика, судоку, японські кросворди, математично-логічні задачі, наприклад задача на переливання чи визначення віку, тощо.

Окремим великим класом логічних задач є задачі з зв'язку фактів. Вони складаються з набору властивостей, кожній з яких відповідає набір значень. При цьому на задачу накладається ряд додаткових умов, які в загальному випадку подаються у вигляді набору відношень значень властивостей. Розв'язок задачі полягає у встановленні всіх зв'язків між значеннями різних властивостей. В англомовній літературі, цей клас задач називається «Zebra puzzle» [2,3]. Надалі під формулюванням «логічна задача» в даній роботі буде матися на увазі задачі саме цього класу. Він, в свою чергу, включається в клас задач, який називається Question-Answering (питання-відповіді).

Найбільш відомим прикладом логічних задач є «Загадка Ейнштейна». Оригінальне формулювання цієї задачі, приведене в журналі Life International у 1962 році, зображено на рисунку 1.1.

1. *There are five houses.*
2. *The Englishman lives in the red house.*
3. *The Spaniard owns the dog.*
4. *Coffee is drunk in the green house.*
5. *The Ukrainian drinks tea.*
6. *The green house is immediately to the right of the ivory house.*
7. *The Old Gold smoker owns snails.*
8. *Kools are smoked in the yellow house.*
9. *Milk is drunk in the middle house.*
10. *The Norwegian lives in the first house.*
11. *The man who smokes Chesterfields lives in the house next to the man with the fox.*
12. *Kools are smoked in the house next to the house where the horse is kept.*
13. *The Lucky Strike smoker drinks orange juice.*
14. *The Japanese smokes Parliaments.*
15. *The Norwegian lives next to the blue house.*

Now, who drinks water? Who owns the zebra?

In the interest of clarity, it must be added that each of the five houses is painted a different color, and their inhabitants are of different national extractions, own different pets, drink different beverages and smoke different brands of American cigarettes. One other thing: in statement 6, right means your right.

Рисунок 1.1 – Оригінальне формулювання «Загадки Ейнштейна»

Приклади логічних задач подано в різних джерелах [4, 5]. Однак, дослідницька група Facebook Research розробила набори логічних задач на ресурсі bAbI [6], які найбільш повно охоплюють всі їх види. В роботі виокремлюються 20 таких видів. До них відносяться:

1. Single Supporting Fact (рисунок 1.2).

Task 1: Single Supporting Fact
Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
Where is Mary?

Task 2: Two Supporting Facts
John is in the playground.
John picked up the football.
Bob went to the kitchen.
Where is the football?

Task 3: Three Supporting Facts
John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen?

Task 4: Two Argument Relations
The office is north of the bedroom.
The bedroom is north of the bathroom.
The kitchen is west of the garden.
What is north of the bedroom?
What is the bedroom north of?

Task 5: Three Argument Relations
Mary gave the cake to Fred.
Fred gave the cake to Bill.
Jeff was given the milk by Bill.
Who gave the cake to Fred?
Who did Fred give the cake to?

Task 6: Yes/No Questions
John moved to the playground.
Daniel went to the bathroom.
John went back to the hallway.
Is John in the playground?
Is Daniel in the bathroom?

Task 7: Counting
Daniel picked up the football.
Daniel dropped the football.
Daniel got the milk.
Daniel took the apple.
How many objects is Daniel holding?

Task 8: Lists/Sets
Daniel picks up the football.
Daniel drops the newspaper.
Daniel picks up the milk.
John took the apple.
What is Daniel holding?

Task 9: Simple Negation
Sandra travelled to the office.
Fred is no longer in the office.
Is Fred in the office?
Is Sandra in the office?

Task 10: Indefinite Knowledge
John is either in the classroom or the playground.
Sandra is in the garden.
Is John in the classroom?
Is John in the office?

Рисунок 1.2 – Приклади задач з ресурсу bAbI

Даний вид задач має один зв’язок значень двох властивостей типу «A - B». Питання, яке ставиться у задачі, полягає у визначенні значення з властивості A по відомому значенню з властивості B.

2. Two Supporting Facts (рисунок 1.2). Даний вид задач має два зв’язка значень трьох властивостей типу «A - B, A - C». Питання, яке ставиться у задачі, полягає у визначенні значення з властивості B по відомому значенню з властивості C. Дана задача є складніша за першу, оскільки потребує наявності пам’яті.

3. Three Supporting Facts (рисунок 1.2). Даний вид задач має три зв’язка значень чотирьох властивостей типу «A - B, A - C, B – D». Питання, яке ставиться у

задачі, полягає у визначенні значення з властивості C по відомому значенню з властивості D. Дану задачу умовно можна представити у вигляді двох попередніх задач. Вимоги, які ставляться до системи цією групою задач, полягає у можливості заміни одного зв'язку задачі групою зв'язків, що автоматично робить можливим будь-яку кількість таких замін.

4. Two Argument Relations (рисунок 1.2). Даний вид задач має зв'язок двох значень властивості типу «A1 - A2». Питання, яке ставиться у задачі, полягає у визначенні значення A1 по відомому значенню A2.

5. Three Argument Relations (рисунок 1.2). Даний вид задач має два зв'язки трьох значень властивості типу «A1 - A2, A2 - A3». Питання, яке ставиться у задачі, полягає у визначенні значення A1 по відомому A3.

6. Yes/No Questions (рисунок 1.2). Даний вид задач має аналогічне формулювання до попередніх, однак питання, яке ставиться у них, полягає не лише у визначенні коректного значення по відомому, але й порівняння його зі значенням, яке вказано у запитанні.

7. Counting (рисунок 1.2). Даний вид задач показує зв'язок значень трьох властивостей типу «A1 - B1, A1 - C1, A1 - D1, A1 - B2, ... , A1 – Bn». Питання, яке ставиться у задачі, полягає у визначенні кількості зв'язків значення A1 з властивістю B.

8. Lists/Sets (рисунок 1.2). Даний вид задач має аналогічне формулювання до попередньої, однак питання, яке ставиться у ньому, полягає у визначенні не лише кількості зв'язків, але і їх конкретних значень.

9. Simple Negation (рисунок 1.2). Даний вид задач схожий з задачею №6, однак окрім наявності зв'язку додається умова його відсутності. Наприклад «A1 - B1, A1 /- B2, A1 - B3». Даний клас задач вимагає від системи можливості задання різних додаткових умов.

10. Indefinite Knowledge (рисунок 1.2). Даний вид задач може формулюватися аналогічно до попередніх, однак в умови може додаватися імовірність зв'язку. При цьому вона може бути задана не лише у постановці задачі, але і у запитанні до неї та відповіді.

11. Basic Coreference (рисунок 1.3).

Task 11: Basic Coreference
Daniel was in the kitchen.
Then he went to the studio.
Sandra was in the office.
Where is Daniel?

Task 12: Conjunction
Mary and Jeff went to the kitchen.
Then Jeff went to the park.
Where is Mary?
Where is Jeff?

Task 13: Compound Coreference
Daniel and Sandra journeyed to the office.
Then they went to the garden.
Sandra and John travelled to the kitchen.
After that they moved to the hallway.
Where is Daniel?

Task 14: Time Reasoning
In the afternoon Julie went to the park.
Yesterday Julie was at school.
Julie went to the cinema this evening.
Where did Julie go after the park?
Where was Julie before the park?

Task 15: Basic Deduction
Sheep are afraid of wolves.
Cats are afraid of dogs.
Mice are afraid of cats.
Gertrude is a sheep.
What is Gertrude afraid of?

Task 16: Basic Induction
Lily is a swan.
Lily is white.
Bernhard is green.
Greg is a swan.
What color is Greg?

Task 17: Positional Reasoning
The triangle is to the right of the blue square.
The red square is on top of the blue square.
The red sphere is to the right of the blue square.
Is the red sphere to the right of the blue square?
Is the red square to the left of the triangle?

Task 18: Size Reasoning
The football fits in the suitcase.
The suitcase fits in the cupboard.
The box is smaller than the football.
Will the box fit in the suitcase?
Will the cupboard fit in the box?

Task 19: Path Finding
The kitchen is north of the hallway.
The bathroom is west of the bedroom.
The den is east of the hallway.
The office is south of the bedroom.
How do you go from den to kitchen?
How do you go from office to bathroom?

Task 20: Agent's Motivations
John is hungry.
John goes to the kitchen.
John grabbed the apple there.
Daniel is hungry.
Where does Daniel go?
Why did John go to the kitchen?

Рисунок 1.3 – Приклади складних задач з ресурсу bAbI

Даний вид задач додає порядок зв'язку між об'єктами. Тобто при умові «A1 - B1, A1 - B2», зв'язок значення A1 буде встановлено спочатку зі значенням B1, а потім – B2.

12. Conjunction (рисунок 1.3). Даний вид задач, на відміну від попереднього, встановлює множинний зв'язок між властивостями. Тобто при умові «A1 - B1, A1 - B2», зв'язок значення A1 буде встановлено спочатку з B1, а в кінці – з B1 та B2 одночасно.

13. Compound Coreference (рисунок 1.3). Даний вид задачі агрегує умови двох задач попереднього виду в множинному представленні. Тобто допускається умова виду «A1 - B1, A1 - B2, A2 - B2, A3 - B3».

14. Time Reasoning (рисунок 1.3). Даний вид задач формулюється аналогічно до попередніх, однак до умови може додаватися час. При цьому, він може бути заданий не лише у постановці задачі, але і у запитанні до запитанні до неї та відповіді.

15. Basic Deduction (рисунок 1.3). Даний вид задач показує зв'язок значень трьох властивостей типу «A1 - B1, B1 - C1». Питання, яке ставиться у задачі, полягає у визначенні значення A1 по відомому значенні C1. Її розв'язок використовує систему роздумів, яка широко використовується в логічних задачах. Саме така система є базовою складовою у задачах типу «Загадка Ейнштейна».

16. Basic Induction (рисунок 1.3). Даний вид задач показує зв'язок значень трьох властивостей типу «A1 - B1, A1 - C1, B1 - D1». Питання, яке у ній ставиться, полягає у визначенні значення C1 по відомому D1. Дану задачу можна представити у вигляді конкатенації двох попередніх, що фактично є умовою множинної конкатенації.

17. Positional Reasoning (рисунок 1.3). Даний вид задач аналогічний до попередніх, однак окрім умов належності «є» та «не є» додаються умови позиціонування («справа», «зліва», тощо).

18. Size Reasoning (рисунок 1.3). Даний вид задач аналогічний до попередніх, однак додаються умови виду «більший»/«менший»

19. Path Finding (рисунок 1.3). Даний вид задач має аналогічне формулювання до задачі №17, однак для відповіді на поставлені питання розв'язуючий механізм повинен мати пам'ять для знаходження повного шляху.

20. Agent's Motivations (рисунок 1.3). Даний вид задач є найбільш складним. Він показує зв'язок значень властивостей типу «A1 - B1, A1 - C1, A2 - B1». Питання, яке ставиться у задачі, полягає у визначенні зв'язку між значеннями A2 та C1 по аналогії до A1. Дані задачі може вирішувати лише «розумна система».

1.2. Класифікація методів розв’язання логічних задач

Існуючі методи розв’язання логічних задач умовно можна поділити на «ручні» та автоматичні. До перших можна віднести методи таблиць, графів, кругів Ейлера, роздумів, тощо [7-9] (рисунок 1.4).

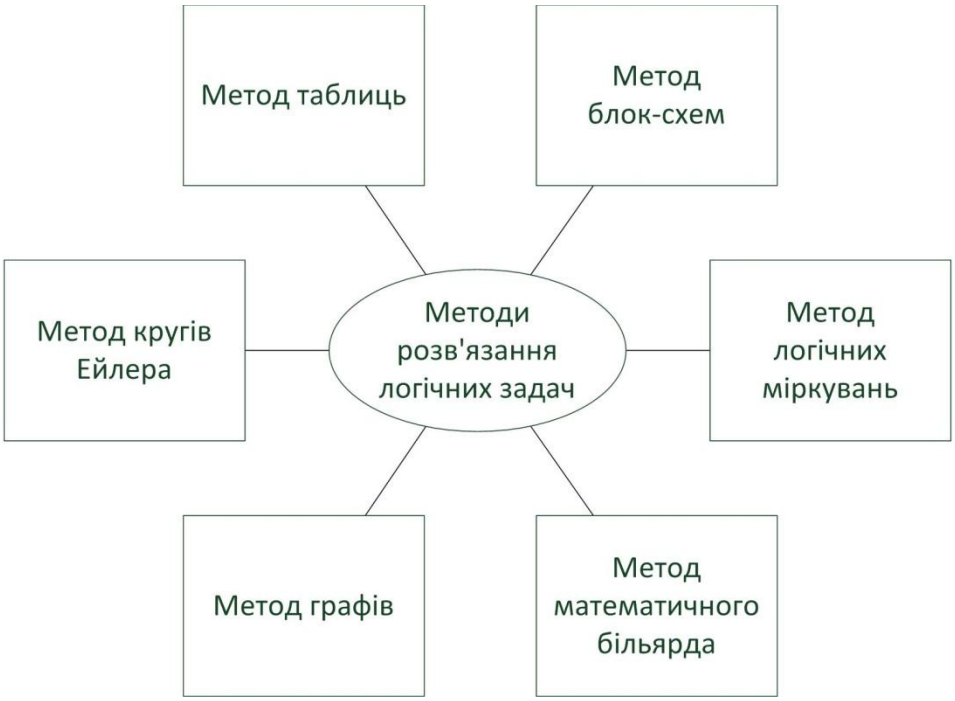

Рисунок 1.4 – Методи розв’язання логічних задач

Однак не всі вони можуть використовуватись для розв’язання логічних задач типу «Загадки Ейнштейна». Деякі з них складно застосовувати для задач зазначеного типу, а деякі – взагалі неможливо.

Метод логічних міркувань є одним з найпростіших. Його суть полягає у прийнятті певної інформації і міркування відносно неї. Для Загадки Ейнштейна даний метод виглядає наступним чином: нехай червоний колір належить третьому будинку. Тоді, в ньому живе англієць. Відповідно, німець, можливо, живе в будинку №1, №2, №4 або №5. Такі роздуми продовжуються до повного розв’язання задачі. Даний метод в програмному виконанні можна представити за допомогою алгоритму повного перебору. Однак, він є ресурсномістким та повільним через можливість хибних допущень і, як наслідок, множинного переприйняття інформації.

Метод блок-схем використовують для розв’язання функціональних задач. До них відносять, наприклад, задачі перестановки та переливання, ряд логістичних задач, тощо. Для розв’язання задачі цим методом будується блок-схема, виконуючи умови якої знаходиться коректна відповідь. Наприклад, нехай є задача: існує 2 ємкості об’ємом 3 і 5 літрів. Необхідно набрати 4 літра води. Для даної задачі можна виділити операції «набрати воду» та «вилити воду» для великої і малої ємності, а також «перелити з більшої ємності в меншу». Алгоритм задачі зображено на рисунку 1.5.

Рисунок 1.5 – Блок-схема розв’язання логічної задачі переливання

Вищеописану задачу можна розв’язати і іншими «ручними» методами, наприклад методом математичного більярду. Його суть полягає в створенні решітки, зображеної на рисунку 1.6 і руху по «шляху більярдного шару».

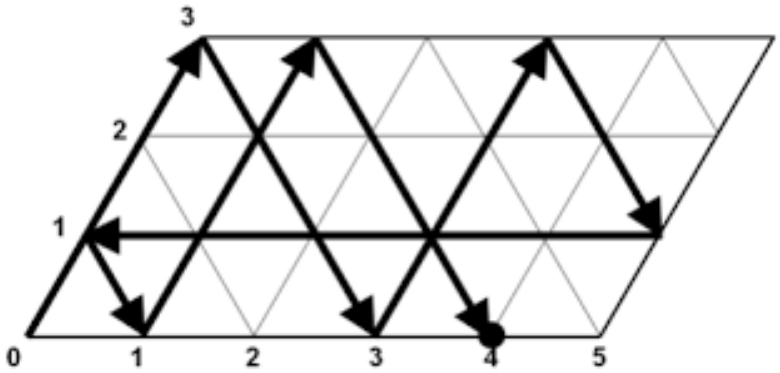

Рисунок 1.6 – Розв’язання задачі переливання методом математичного більярду

Метод таблиць є найбільш популярним та полягає у складенні таблиці зв’язків значень однієї властивості з іншими. Початок розв’язання «Загадки Ейнштейна» цим методом показано на рисунку 1.7.

№ будинку	1	2	3	4	5
Колір будинку	Жовтий	Синій	?	?	?
Національність	Норвежець	?	?	?	?
Напій	Вода	?	Молоко	?	?
Цигарки	Dunhill	?	?	?	?
Тварина	?	Кінь	?	?	?

Рисунок 1.7 – Перший етап розв’язання «Загадки Ейнштейна»

В даному методі одна із властивостей, як правило, є базовою. На рисунку в верхній частині таблиці знаходиться зафіксована властивість – номер будинку. Значення інших властивостей записують по ходу розв’язання задач відповідно до неї. Зауважимо, що даний метод також використовує метод логічних міркувань для доповнення неочевидних фактів.

1.3. Підходи до автоматичного розв’язання логічних задач

Якщо логічна задача має більше чотирьох властивостей, кожна із яких має по чотири значення, то використання «ручних» методів стає занадто складним. Такі задачі потребують ефективних комп’ютерних алгоритмів.

В роботі [10] був описаний варіант розв’язання простих задач типу “питання-відповідь” за допомогою динамічної мережі покриття (dynamic coattention network). На першому кроці система сканує текст, виконує розпізнавання лексем та початкове навчання мережі. Після цього на аналізатор подаються питання, а після їх сканування та розбиття на лексеми отримані дані подаються на мережу. Мережа вибирає перелік кандидатів з можливою відповіддю, визначає з них найбільш імовірні та надає користувачу (рисунок 1.8).

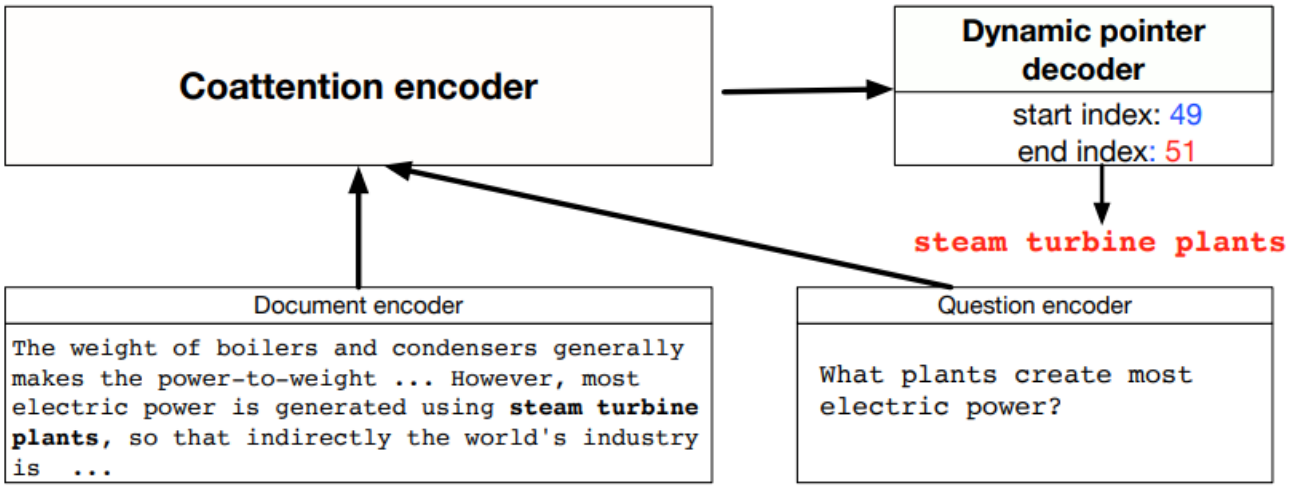

Рисунок 1.8 – Структура динамічної мережі покриття

Базовою частиною мережі є енкодер покриття (coatention encoder). Суть його роботи полягає в одночасній обробці як запитання, так і документа (рисунок 1.9). Для цього спочатку обчислюється матриця спорідненості, що містить показники взаємодії слів з тексту і запитання. Далі обраховуються контексти пар слів, після чого йде формування фраз.

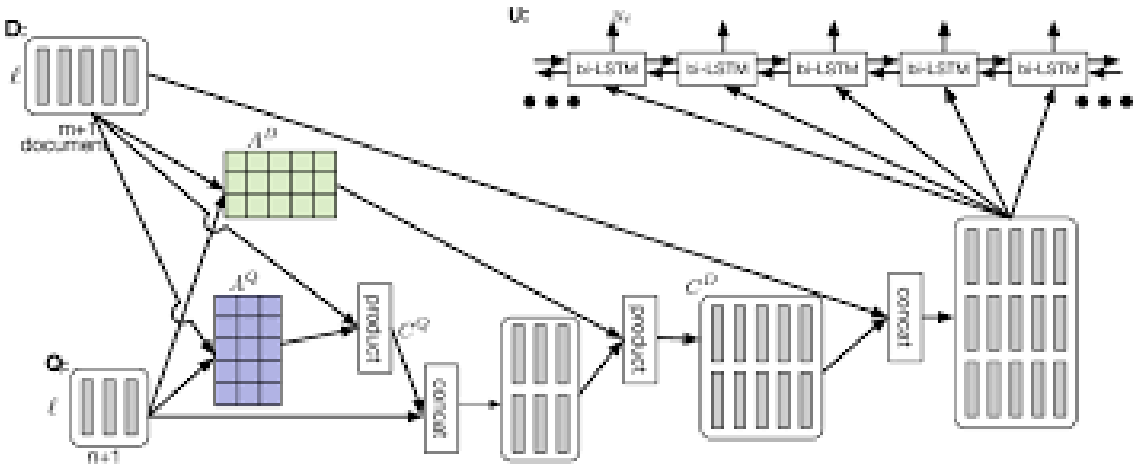

Рисунок 1.9 – Енкодер покриття

На наступному кроці фрази подаються на декодер, де обраховуються потенційні відповіді. Даний етап надає відповідям імовірнісні характеристики за рахунок використання багатошарової нейронної мережі. Результатом роботи мережі є поле імовірностей відповідей, внаслідок чого і розв'язується задача.

Відповідь на запитання формується машиною виводу за даними мережі. Для додаткового донавчання мережа може сприймати від користувача дані, щодо коректності відповіді. Однак мережа не здатна обробляти задачі, в яких потребуються логічні висновки та роздуми.

Аналогічний алгоритм розв'язання задач наведено у роботі [11]. Перш за все автори вдосконалили механізм генерування відповідей. Хоч більшість питань вимагає коротких відповідей, було розроблено алгоритм, який генерує довгі відповіді для можливості проходження тесту Тьюринга. Для розв'язання задач було використано рекурентну нейронну мережу RNN (Recurrent neural network). Вона здатна реалізовувати напрямлені зв'язки. Це дозволяє обробляти не один об'єкт, а серію. Завдяки цьому стає можливим обробка даних не визначеної довжини, що ідеально підходить для розпізнавання текстової інформації. В роботі автори перетворюють кожне слово вхідного тексту на спеціальний числовий вектор довжиною до 300 знаків. Вхідні зображення також трансформуються в числові вектори. Далі на рекурентну нейронну мережу подаються сформовані вектори. Результатом роботи нейронної мережі є відповідь у векторному виді (рисунок 1.10).

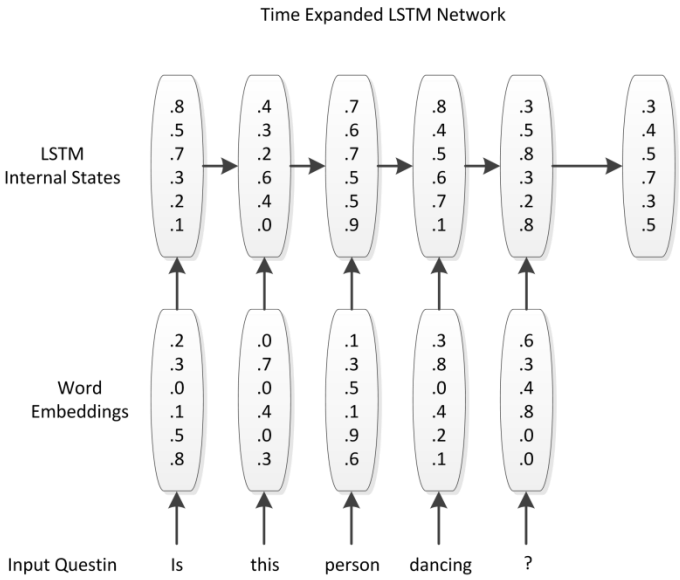

Рисунок 1.10 – RNN для логічних задач

Однак недоліком підходу є ігнорування послідовності питань. Для цього була зроблена спроба вдосконалення рекурентної нейронної мережі, що покращило результати. Однак, грубим недоліком цього методу є можливість розв’язання лише тих задач, де коректна відповідь знаходиться в тексті чи зображенні і немає необхідності будувати ланцюг логічних роздумів.

В роботі[12] було наведено приклади програмних реалізацій алгоритмів розв’язання «Загадки Ейнштейна». Автор виконав їх різними мовами, тому при необхідності їх можна вбудувати в інші системи. Однак метод розв’язання жорстко запрограмований на кількість властивостей, значень, та зв’язків між ними. Тому зміна однієї властивості чи умови потребує редагування великої частини коду. Причина цього полягає в тому, що для розв’язання було програмно реалізовано метод логічних міркувань. З одного боку це дозволило отримати швидкий механізм розв’язання логічних задач. Однак абсолютна неможливість створення гнучкої системи, тобто системи з можливістю зміни кількості та значень властивостей та вхідних умов в режимі реального часу, є серйозною перешкодою для використання у промислових системах. Очевидно, що для програмної реалізації гнучкої системи не підходять існуючі «ручні» методи розв’язання логічних задач.

Для спрощення виконання логічних роздумів задачі для систем, як правило, формалізують і подають у синтаксично зрозумілому для системи вигляді. Так, наприклад, у роботі [13] було представлено мережу “Sherlock”, яка вирішує логічні задачі шляхом використання парадигми об’єднання індуктивного навчання і логічного програмування (рисунок 1.11).

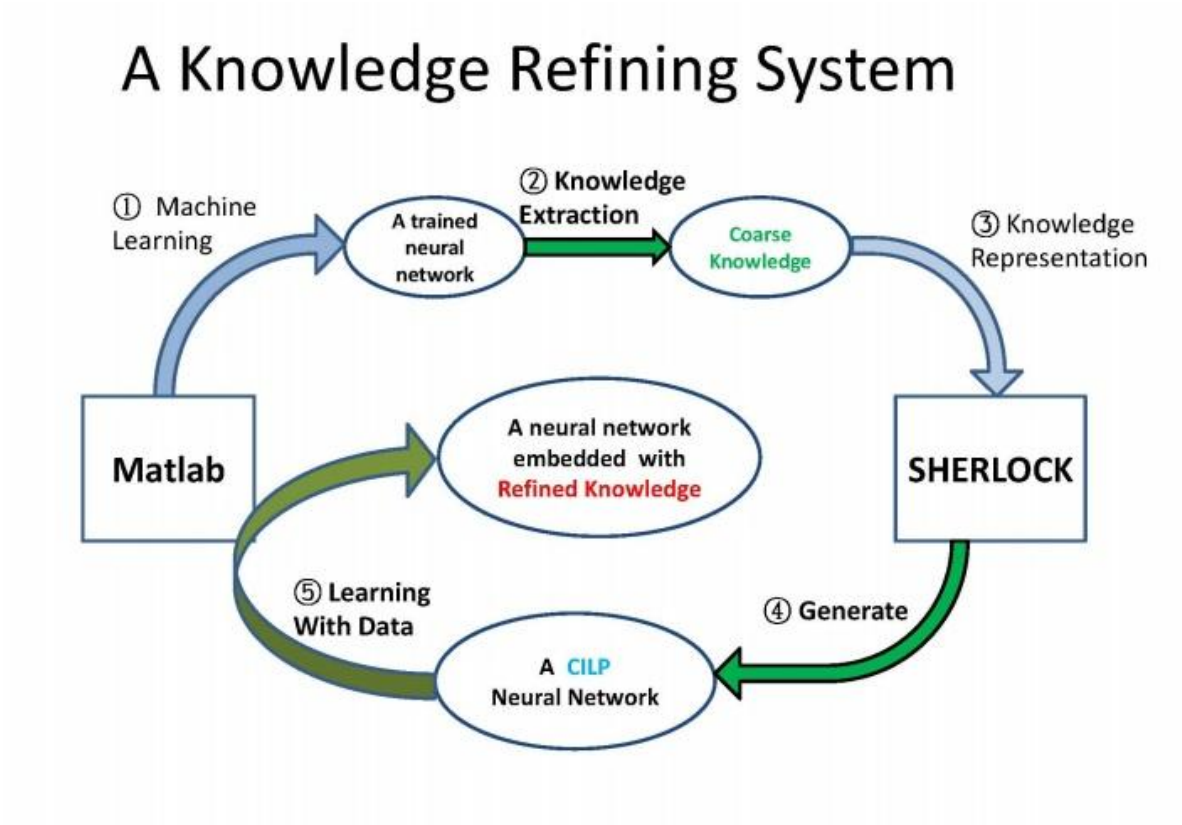

Рисунок 1.11 – Використання мережі Sherlock

Суть підходу базується на методології мови Prolog, яка заключається в програмуванні алгоритму розв’язання задачі за допомогою фактів. Властивості та значення задачі автори подають у вигляді набору логічних термів, а додаткові умови – у вигляді фактів та правил. Перевагою такого методу є відносна простота. Однак метод жорстко реалізує розв’язання конкретних задач та не може використовуватись як уніфіковане рішення. Більш того, використання даного методу вимагає не лише розробленого програмного рішення, але і додаткових програмних продуктів, наприклад MathLab.

Дану проблему частково було вирішено в роботі [14], в якій автор застосував для розв’язання задачі метод, що базується на використанні булевої алгебри (рисунок 1.12).

1	2	3	4	5
Yellow	Blue	?	?	?
Norway	?	?	?	?
?	?	Milk	?	?
Kools	?	?	?	?
?	?	?	?	?

- EnglishRed(3) XOR EnglishRed(4) XOR EnglishRed(5)
- SpainDog(2) XOR SpainDog(3) XOR SpainDog(4) XOR SpainDog(5)
- UkraineTea(2) XOR UkraineTea(4) XOR UkraineTea(5)

Рисунок 1.12 – Розв’язання «Загадки Ейнштейна» за допомогою булевої логіки

Фактично була описана пропозиція представлення логічної задачі у вигляді набору булевих функції, які, в свою чергу, спрощуються за допомогою описаного методу використання таблиць. В цьому випадку, значення властивостей задачі подаються у вигляді логічних операндів, а умови реалізуються шляхом використання стандартних логічних функцій. Однак для такого представлення задачі спеціаліст повинен мати знання в булевій алгебрі, що не завжди є можливим. Більш того, для задання чи зміни задачі потребується багато часу. Ще одним недоліком є те, що даний метод не дозволяє реалізовувати складні відношення значень властивостей логічної задачі. При збільшенні кількості властивостей, значень, і як наслідок умов, задача буде занадто складною для розв’язання даним методом.

В роботі [15] автори розробили уніфікований алгоритм розв’язання логічних задач шляхом використання лінійної алгебри (рисунок 1.13).

Table 2 Constraints in the mathematical model of the riddle

Description	Mathematical model	
The British lives in the red house.	$\sum_{j=1}^{5} j \cdot N_{5j} = \sum_{j=1}^{5} j \cdot HC_{4j}$	(1)
The Swedish keeps dogs as pets.	$\sum_{j=1}^{5} j \cdot N_{2j} = \sum_{j=1}^{5} j \cdot P_{4j}$	(2)
The Dane drinks tea.	$\sum_{j=1}^{5} j \cdot N_{4j} = \sum_{j=1}^{5} j \cdot B_{3j}$	(3)
The green house is on the left of the white house.	$\sum_{j=1}^{5} j \cdot HC_{2j} < \sum_{j=1}^{5} j \cdot HC_{5j}$	(4)
The green house's owner drinks coffee.	$\sum_{j=1}^{5} j \cdot HC_{2j} = \sum_{j=1}^{5} j \cdot B_{1j}$	(5)
The owner who smokes Pall Mall rears birds.	$\sum_{j=1}^{5} j \cdot CB_{5j} = \sum_{j=1}^{5} j \cdot P_{1j}$	(6)

Рисунок 1.13 – Розв’язання “Загадки Ейнштейна” за допомогою лінійної алгебри

В ній вони розглянули можливість використання «Загадки Ейнштейна» для вирішення задач конструктивно блокової геометрії [16]. Після короткого формулювання задачі, автори привели варіант формалізації логічних задач обраної прикладної сфери. Для цього вони провели в ній аналіз операцій. Було представлено математичну постановку задачі Ейнштейна, приведено механізм її розв’язання, представлено результати роботи системи, а також надано рекомендації щодо вдосконалення методу. Обчислювальні експерименти проводилися на задачі розподілу будівельної техніки. Метою стала мінімізація витрат визначених машин. Хоч метод і не вирішує всіх типів задач, наприклад задачі множинного відношення, а створене по ньому програмне рішення не є оптимізованим, можна вважати, що на сьогоднішній день він найбільш оптимально та повно охоплює існуючі розробки спеціалістів даної сфери. Більш того, у висновках автори запропонували подальшим дослідникам провести оптимізацію визначеного методу шляхом використання машинного навчання та нейронних мереж.

В роботі [17] було наведено аналогічний до попереднього механізм вирішення логічних задач. Автори запропонували такий метод вирішення: нехай, змінна j = 1..5

відображає національність мешканця, $j = 1$ – британець, $j = 2$ – данець, і т.д. Тоді визначимо N_{ij}, як можливість мешкання людини з певною національністю в визначеному i-му будинку. Якщо даний параметр дорівнює одиниці – в будинку живе людина визначеної національності, інакше – ні. Тоді матимуть місце твердження:

$$\sum_{j=1}^{5} N_{ij} = 1, i = 1..5$$

$$\sum_{i=1}^{5} N_{ij} = 1, j = 1..5$$

Таким чином можна визначити всі властивості задачі.

Кожна додаткова умова формулюється як зв'язок між номером будинку і значенням в умові. Так, наприклад, умову «Норвежець живе у першому будинку» можна подати у вигляді:

$$\sum_{i=1}^{5} i * N_{i3} = 1$$

Аналогічно задаються і інші умови. Якщо умова не пов'язана з номером будинку, то таку умову відображають у вигляді рівності двох сум. Наприклад, для умови «Данець п'є чай» можна скласти таке рівняння:

$$\sum_{i=1}^{5} i * N_{i2} = \sum_{i=1}^{5} i * D_{i1},$$

де D_{ij}, аналогічно до N_{ij}, дорівнює 1, якщо людина, що живе в будинку i, п'є напій з номером j, і дорівнює 0, якщо ні.

Після цього отримується система рівнянь, які можна вирішити різними числовими методами.

Недоліком даного методу, як і у попереднього, є складність вирішення більших задач з складними умовами.

Окрім описаних алгоритмів існують і інші, які, як правило, менш продуктивні та прості [18-28]. З огляду на існуючі проблеми в області розв’язання логічних задач, необхідно розробити універсальний механізм їх автоматичного розв’язання на основі машинного навчання.

Висновки до розділу 1

1. Проведено аналіз логічних задач різних типів. Наведено їх класифікацію.
2. Визначено задачі, які є еталонними для тестування та визначення ефективності спеціалізованих програмних систем.
3. Проведено аналіз існуючих методів розв’язання логічних задач без використання обчислювальних механізмів. Визначено їх переваги та недоліки.
4. Проведено аналіз існуючих методів автоматичного розв’язання логічних задач. Визначено їх переваги та недоліки.
5. Обґрунтована необхідність створення універсального механізму розв’язання логічних задач.

2. МЕРЕЖА ЗВ'ЯЗКІВ

Для розв'язання логічних задач перш за все необхідно виконати їх формалізацію. Формалізація логічної задачі – це її представлення в стандартному вигляді для можливості представлення інших задач цього класу аналогічним чином. Формалізацію логічних задач наведено у першому підрозділі. Після виконання формалізації задач потрібно розробити спеціальну обчислювальну структуру та алгоритми її навчання. Розроблена структура відповідно до поставлених вимог повинна бути гнучкою, тобто дозволяти розв'язувати задачі з різною кількістю властивостей, значень та додаткових умов. Окрім цього вона повинна давати вищу швидкість, ніж аналогічні рішення. Компоненти та архітектура розробленої структури представлена в другому підрозділі, алгоритм її навчання – в третьому.

2.1. Формалізація логічної задачі

Кожна логічна задача визначеного класу складається з набору властивостей, кожна з яких має набір можливих значень. Кількість значень може бути або однаковою для кожної властивості, або різною. Для спрощення формалізації припустимо, що кількість значень кожної властивості є однаковою. Для випадку, коли кількість значень різна використовується аналогічний підхід до формалізації.

Для задачі Ейнштейна, кількість властивостей n = 6, а кількість значень кожної з них m = 5. Тоді i-ту властивість P_i (property) можна подати у вигляді:

$$P_i = \{v_i^j\},$$

де P_i – i-та властивість, $i = 1..n$,

v_i^j – значення j властивості i, $i = 1..n$, $j = 1..m$

В таблиці 2.1 наведено перелік значень v_{ij}, групованих за властивостями, для «Загадки Ейнштейна». Зауважимо, що дана таблиця, як структура даних, не використовується в методі розв'язання, а лише слугує для більш наочного представлення інформації.

Таблиця 2.1. Властивості та значення «Загадки Ейнштейна»

v_{ij}	P_i					
i / j	1	2	3	4	5	6
	№ будинку	Колір будинку	Національність	Улюблений напій	Улюблені цигарки	Домашня тварина
1	1	Білий	Англієць	Вода	Dunhill	Кішка
2	2	Жовтий	Датчанин	Кава	Marlboro	Кінь
3	3	Зелений	Німець	Молоко	Pall Mall	Пташка
4	4	Червоний	Норвежець	Пиво	Phillip Morris	Риби
5	5	Синій	Швед	Чай	Rothmans	Пес

Кожна k-та можлива комбінація значень властивостей C_k (combination), матиме вид:

$$C_k = \left(x_k^1, x_k^2, ..., x_k^n\right),$$

де k – номер поточної комбінації,

v_i^k – можливе значення i-ї властивості, $v_i^k \in P_i$.

Наприклад, C_1 = (1, білий, англієць, вода, Dunhill, кішка), C_2= (1, білий, англієць, вода, Dunhill, кінь), та інші.

Максимальна кількість комбінацій без урахування додаткових умов:

$$q = m^n,$$

де n – кількість властивостей задачі

m – кількість значень властивостей задачі

Рішенням логічної задачі буде множина комбінацій, що не суперечать всім її умовам:

$$S = \{C_1, C_2, \ldots, C_o\},$$

де o – кількість коректних комбінацій значень властивостей, $o \le q$.

Для вирішення задачі використовуються умови відношень між двома значеннями властивостей:

$$v_{i_1}^{j_1} \prec rel \succ v_{i_2}^{j_2} \qquad (2.1)$$

де $v_{i_1}^{j_1}$ – значення j_1 властивості i_1, $j_1 \in 1..m$, $i_1 \in 1..n$,

$v_{i_2}^{j_2}$ – значення j_2 властивості i_2, $j_2 \in 1..m$, $i_2 \in 1..n$,

$\prec rel \succ$ – визначене відношення між властивостями з зазначеними значеннями.

При формалізації задачі всі зв’язки предметної області зводяться до тверджень про наявність/відсутність відповідного зв’язку, що позначається $\prec \epsilon \succ / \prec не \epsilon \succ$. Наприклад: $1 \prec \epsilon \succ$ білий, $2 \prec не \epsilon \succ$ зелений.

2.2. Компоненти та архітектура мережі зв'язків

Для розв’язання задачі було розроблено спеціальну обчислювальну структуру – мережу зв’язків. Вона представляє собою n-мірну «решітку». Розмірність «решітки» дорівнює кількості властивостей задачі. Для «Загадки Ейнштейна» розмірність дорівнює 6. Розмір по кожній вісі дорівнює кількості значень відповідної властивості. Для «Загадки Ейнштейна» розмір по кожній вісі є

однаковим і дорівнює 5. Якщо задача має однакову кількість значень для кожної властивості – мережа представляє собою n-мірну кубічну «решітку».

Оскільки зображення в тримірному Евклідовому просторі об’єктів з розмірністю більше трьох складне, тому надалі приклади будуть приводитися для n = 2..3. Однак всі підходи, методи та правила можна застосовувати і для більших розмірностей.

Мережа, яка створюється для розв’язання задачі з двома властивостями, кожен з яких має по 3 значення, зображена на рисунку 2.1.

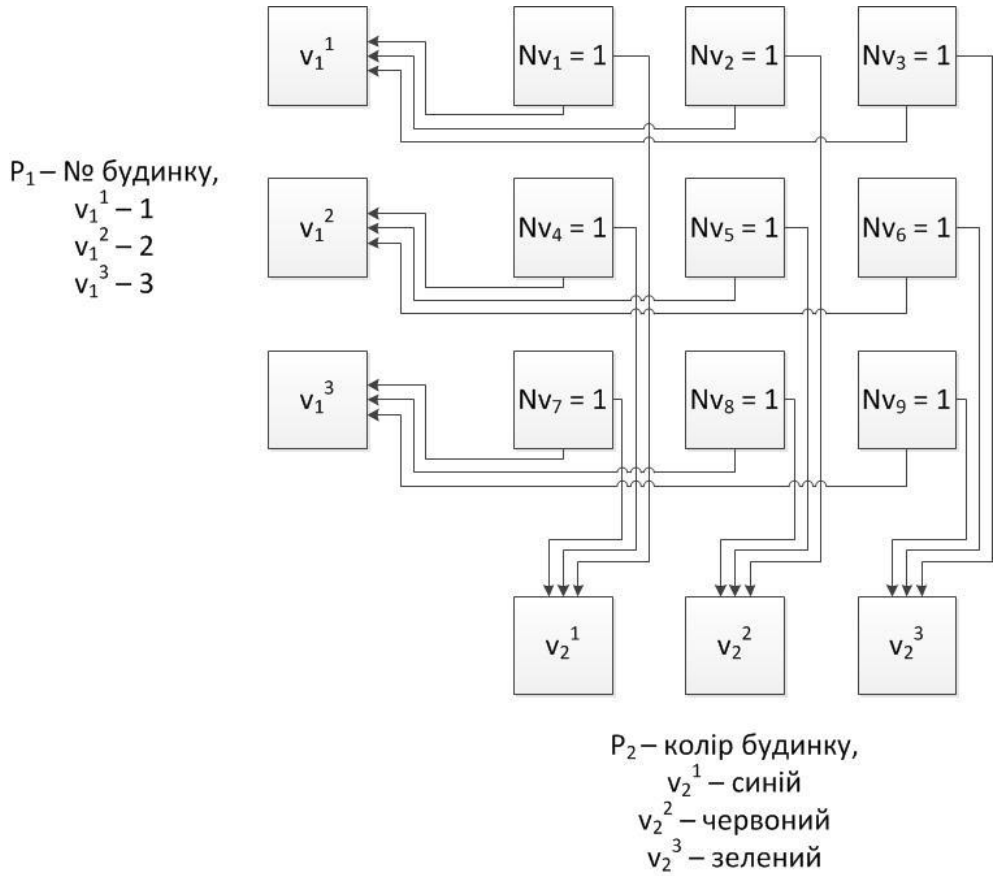

Рисунок 2.1 – Мережа для задачі на 2 властивості по 3 значення

Прикладом задачі для даної мережі може бути, наприклад, неповна «Загадка Ейнштейна», яка складається з номеру будинку, що може приймати значення 1, 2, 3, та його кольору, що може приймати значення синій, червоний, зелений.

Мережа зв’язків складається з двох груп вхідних вузлів. Перша група відповідає за номер будинку і містить три вхідні вузли, які в свою чергу

відповідають за номер 1, 2, 3 відповідно. Друга група вхідних вузлів відповідає за колір будинку і містить три вхідні вузли, які в свою чергу відповідають за синій, червоний та зелений кольори відповідно.

Вхідні вузли різних груп з'єднуються між собою, реалізуючи можливі комбінації C_k. Кожну таку комбінацію відображає конкретний обчислювальний вузол. Кожен обчислювальний вузол містить індекс і інформацію про коректність комбінації. Дана інформація може мати велику кількість варіантів побудови, наприклад набір імовірнісних характеристик. Однак для логічних задач визначеного класу, де комбінація може бути лише абсолютно коректною або навпаки, інформація про коректність може представлятися у вигляді дискретного власного значення – *Nv* (Node value). Воно приймає значення 0, якщо комбінація C_k неможлива, і 1 – якщо можлива. Таким чином, для розв'язання задачі необхідно шляхом зміни власних значень вузлів навчити мережу та виокремити коректні комбінації.

Для розв'язання логічних задач обраного класу, алгоритм використовує поняття (n-1)-мірних шарів. Даний шар отримується шляхом фіксування значення одного з параметрів і отримання всіх вузлів за цим значенням.

Кількість $Q_{n\text{-}1}$ (n-1)-мірних шарів можна визначити за формулою:

$$Q_{n-1} = \sum_{i=1}^{n} m_i ,$$

де m_i – кількість значень властивості i

Для вищезазначеного прикладу існує 6 (n-1)-мірних шарів. Їх зображено на рисунку 2.2.

З точки зору геометричного представлення, (n-1)-мірний шар мережі при:

1) n=2, є рядком або стовбцем;

2) n=3, є площиною;

3) n=4, є паралелепіпедом.

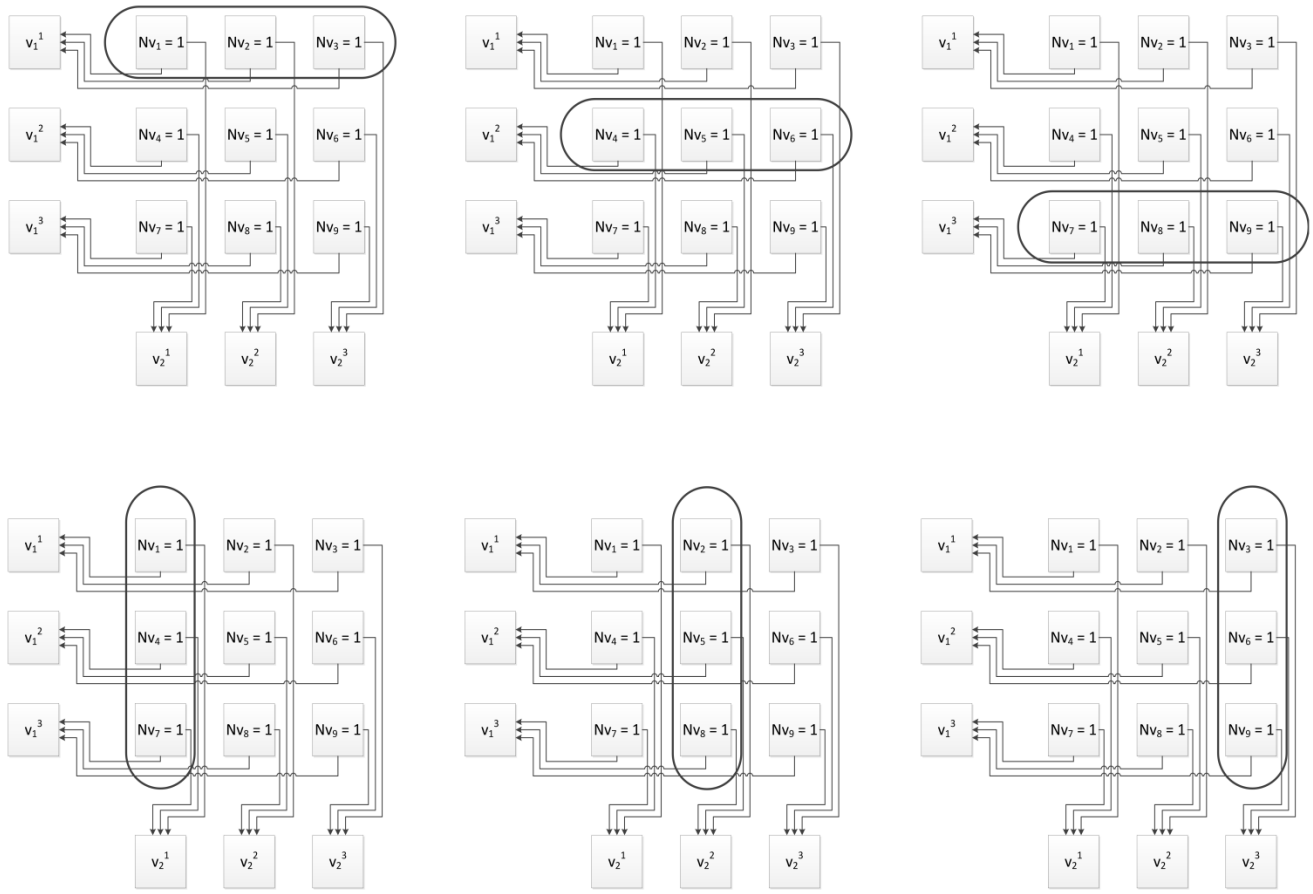

Рисунок 2.2 – (*n*-1)-мірні шари для мережі зв’язків при $n = 2$, $m = 3$

На рисунку 2.3 зображено структуру обчислювальних вузлів для мережі зв’язків, яка призначена для розв’язання задачі з трьома властивостями, кожна з яких складається з трьох, чотирьох і п’яти значень відповідно. Для неї існують три двомірні шари розміром 4x5, чотири двомірні шари розміром 3x5 та п’ять двомірних шарів розміром 3x4. Аналогічно можна виділяти шари нижчих порядків.

Для додавання нових значень властивостей мережа повинна бути перебудована. Однак повне оновлення на великих задачах може займати багато ресурсів і часу. Для того, щоб не перебудовувати всю мережу достатньо лише додати (n-1)-мірний шар. Це надає перевагу над більшістю інших існуючих методів розв’язання логічних задач, оскільки в їх алгоритмах для додавання нового значення властивості необхідно перебудовувати всю структуру.

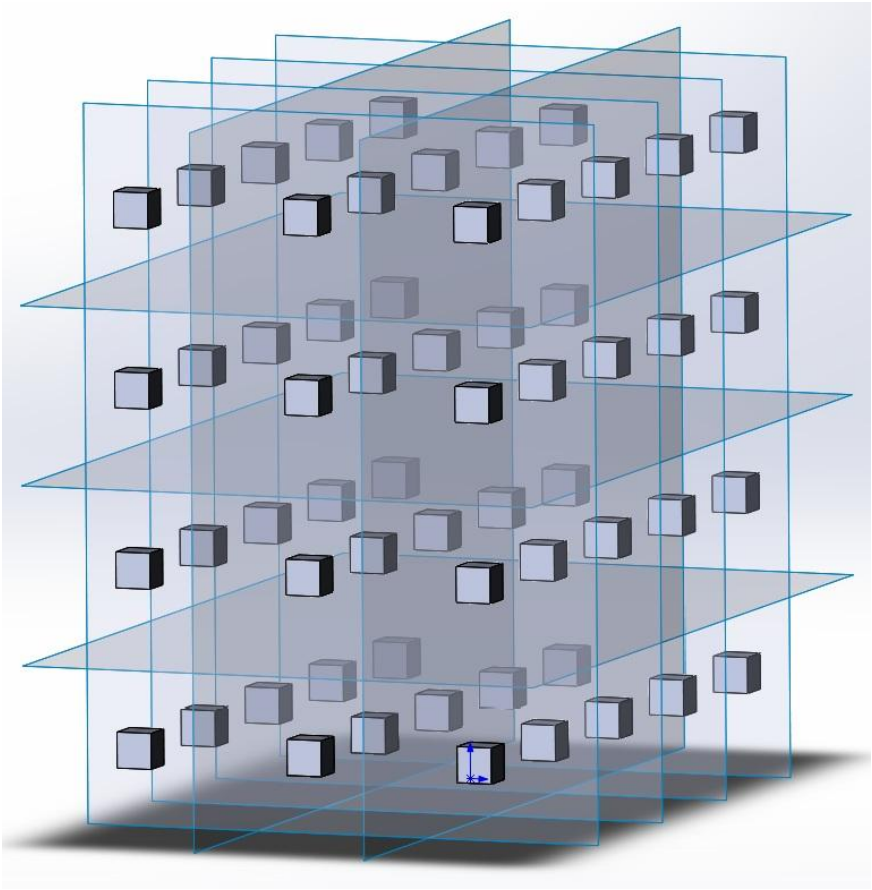

Рисунок 2.3 – Мережа зв’язків для $n = 3$, $m_1 = 3$, $m_2 = 4$, $m_3 = 5$

Для додавання нового (n-1)-мірного шару необхідно:

1. Додати до мережі обчислювальні вузли. Їх кількість Qn можна визначити за формулою:

$$Qn = \prod_{i=1}^{k-1} m_i * \prod_{i=k+1}^{n} m_i,$$

де m_i – розмір мережі по i-му виміру,

k – номер властивості, до якої додається нове значення.

Наприклад, для перетворення мережі, зображеної на рисунку 2.3, на мережу з розмірами 3, 5, 5 необхідно додати новий двомірний шар, розмір якого буде дорівнювати:

$$Qn = 3*5 = 15$$

2. Перевизначити індекси вузлів. Для цього використовується алгоритм визначення одномірного індексу з *n*-мірного.

2.3. Алгоритм навчання мережі зв’язків

Для отримання розв’язку задачі, після її повного опису і створення мережі зв’язків необхідно провести її навчання. Воно складається з ініціалізації, циклічного виконання епох навчання та перевірки закінчення навчання після кожної епохи.

Ініціалізація мережі зв’язків полягає в початковому встановленні власних значень. В межах даної роботи для розв’язання визначеного типу логічних задач вони ініціюються значенням 1 (рисунок 2.4).

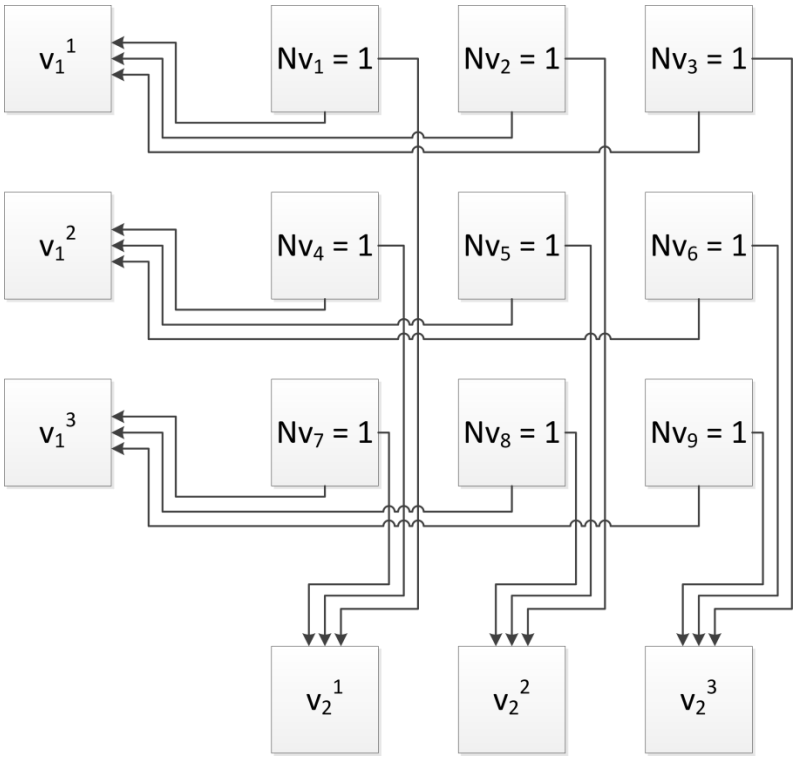

Рисунок 2.4 – Ініціалізація мережі зв’язків

Це означає, що перед початком навчання можливі всі комбінації значень властивостей задачі.

Кожна епоха навчання складається з двох основних етапів: послідовного представлення навчальних прикладів мережі і подальшого уточнення значень

обчислювальних вузлів мережі. Під час епохи навчання власні значення вузлів змінюються (рисунок 2.5).

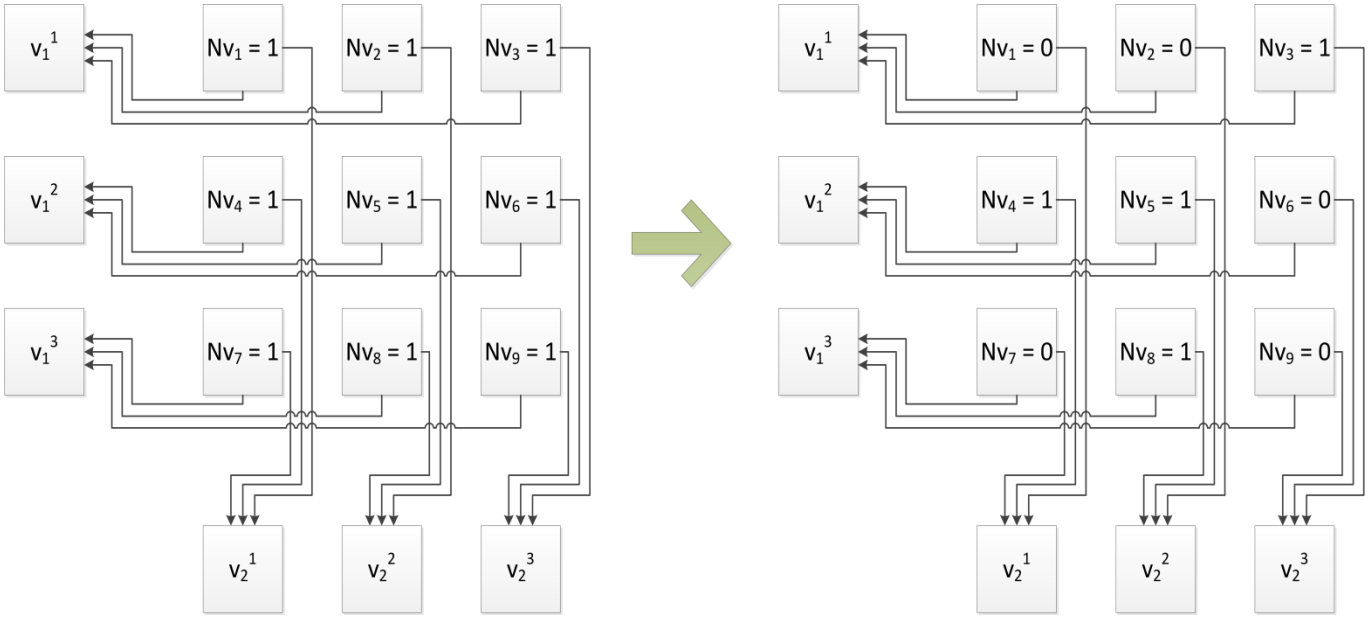

Рисунок 2.5 – Один крок навчання мережі

Навчальним прикладом називається відношення між значеннями властивостей. Однак, будь-яке множинне відношення можна представити у вигляді набору відношень між обраними значеннями двох властивостей, що виражається формулою (2.1). Саме тому доцільно розглядати саме ці відношення. Представленням навчальних прикладів називають послідовну подачу на механізми навчання мережі всіх навчальних прикладів з подальшим перерахунком вузлів мережі.

Подача навчального прикладу відбувається за таким алгоритмом:

1. Отримання всіх комбінацій мережі, які представляються її обчислювальними вузлами.

2. Для кожної комбінації:

2.1. Очищення суми збігів, шляхом встановлення їй значення 0.

2.2. Для всіх властивостей:

2.2.1. Якщо значення властивості прикладу і поточної комбінації співпадають – суму збігів збільшити на одиницю.

2.3. Оновлення значення поточного вузла за формулою:

$$N_v^{'} = N_v \times F(S,R),$$

де $N_v^{'}$ – нове значення обчислювального вузла,

N_v – значення обчислювального вузла, встановлене після подачі попереднього прикладу,

S – кількість збігів значень властивостей,

R – відношення між значеннями властивостей,

$F(S,R)$ – функція, яка залежить від кількості збігів значень властивостей і визначається в таблиці 2.1.

Таблиця 2.1. Значення функції оновлення

Вхідні параметри		Значення функції
R	S	F
$\prec \epsilon \succ$	0	1
$\prec \epsilon \succ$	1	0
$\prec \epsilon \succ$	2	1
$\prec не \epsilon \succ$	0	1
$\prec не \epsilon \succ$	1	1
$\prec не \epsilon \succ$	2	0

На рисунку 2.6 зображено стан обчислювальних вузлів після подачі навчального прикладу $v_{13} \prec \epsilon \succ v_{22}$.

1. Значення вузлів Nv_1, Nv_4, Nv_3, Nv_6, з сумою активуючих сигналів 0, не повинно змінюватись, оскільки комбінації, що відображають визначені вузли, умова прикладу навчання не змінює. Для цього значення функції $F(S,R)$ повинно дорівнювати 1.

2. Значення вузла Nv_8, з сумою активуючих сигналів 2, повинно бути встановлене в одиницю, оскільки дана комбінація є коректною. Однак, оскільки додаткова умова задається у вигляді зв’язку значень двох властивостей, то вузли, які будуть мати суму 2, будуть складати (n-2)-мірний шар.

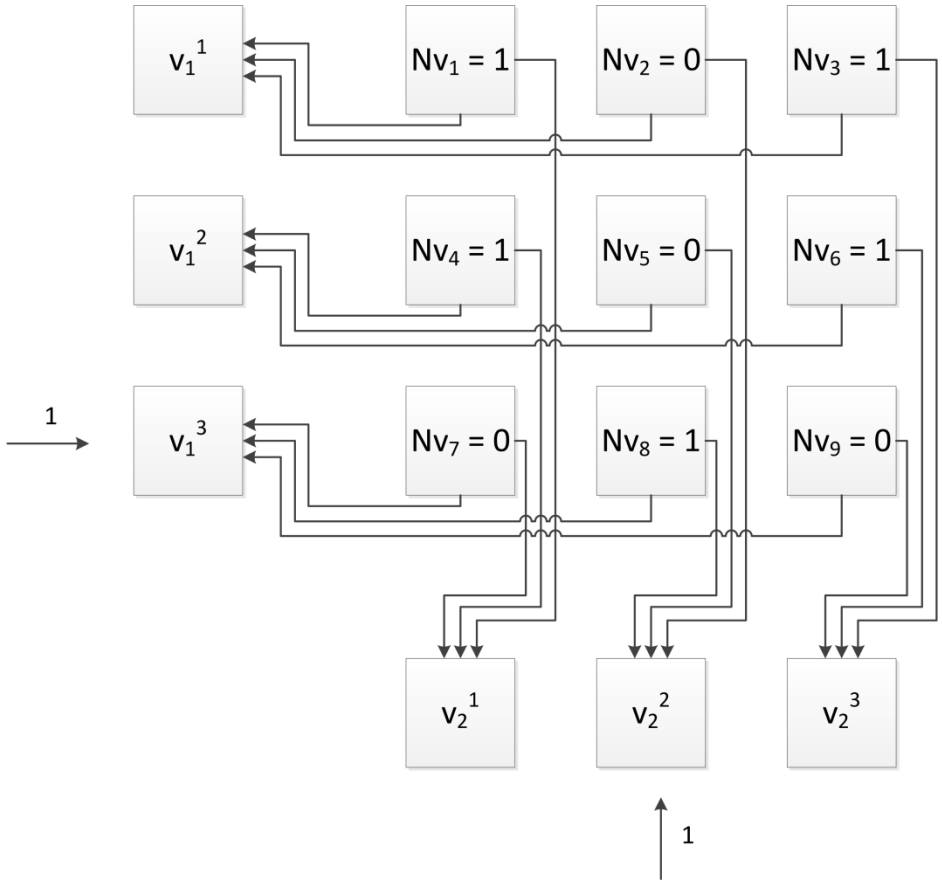

Рисунок 2.6 – Стан вузлів після подачі навчального прикладу $v_{13} \prec \epsilon \succ v_{22}$.

Оскільки після попереднього прикладу деякі вузли вже можуть мати власне значення $Nv = 0$, що означає неможливу комбінацію, то таке значення повинно бути збереженим. Саме тому необхідно не жорстко встановлювати значення 1, а множити існуюче значення на 1. Тоді, для невизначеного значення, яке після ініціалізації дорівнює 1, нове значення також буде дорівнювати 1. З іншого боку, якщо попереднє значення вузла дорівнює 0 – воно не буде зміненим.

3. Сума активуючих сигналів 1 вузлів з власним значенням Nv_2, Nv_5, Nv_7, Nv_9 означає, що дана комбінація неможлива, оскільки вона суперечить коректній комбінації вузлів, з сумою активуючих сигналів 2. Нове значення вузла повинно дорівнювати 0, тому значення функції для них також повинно бути 0.

На рисунку 2.7 зображено стан обчислювальних вузлів після подачі наступного навчального прикладу $v_{12} \prec не\epsilon \succ v_{21}$.

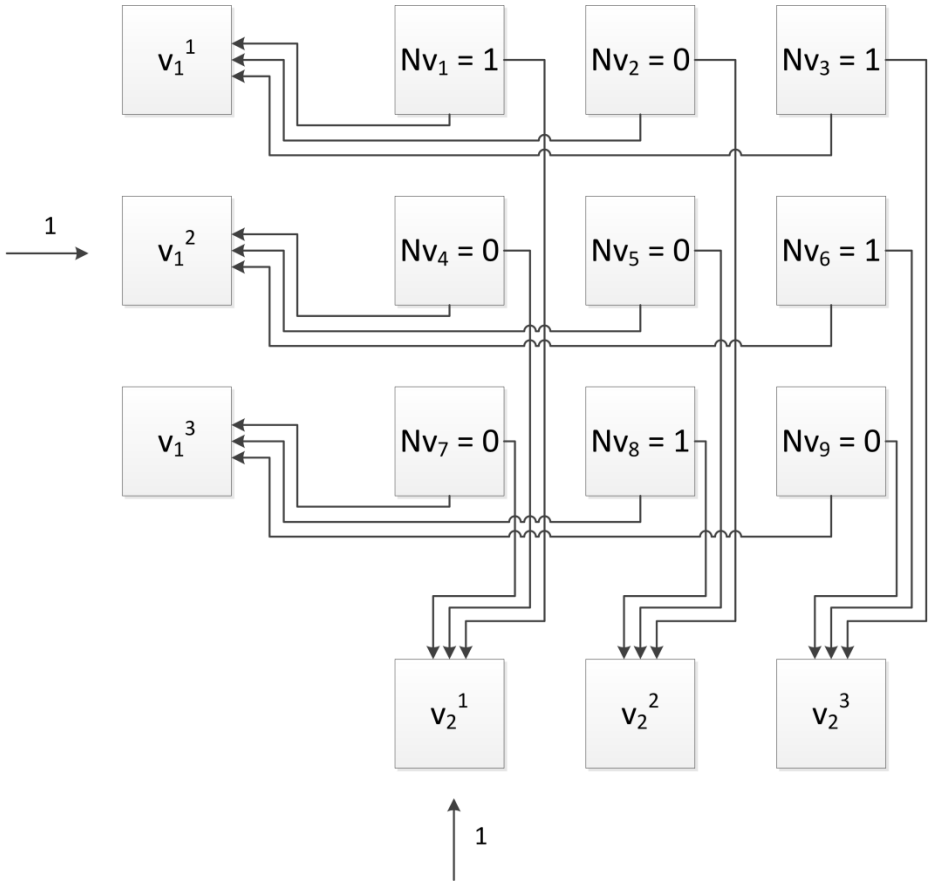

Рисунок 2.7 - Стан вузлів після подачі навчального прикладу $v_{12} \prec не є \succ v_{21}$.

1. Сума активуючих сигналів 2 вузла з власним значенням Nv_4 означає, що дана комбінація неможлива. Значення функції для таких вузлів повинно дорівнювати 0.

2. Сума активуючих сигналів 1 вузлів з власним значенням Nv_1, Nv_7, Nv_5, Nv_6 означає, що комбінація може бути як можливою так і неможливою. Для незмінності невизначеності значення функції повинно дорівнювати 1.

3. Сума активуючих сигналів 0 вузлів з власним значенням Nv_2, Nv_3, Nv_8, Nv_9 має аналогічний зміст до тих, для яких сума активуючих сигналів дорівнює 1. Значення функції в цих вузлах аналогічно повинно дорівнювати 1.

Після представлення всіх навчальних прикладів відбувається уточнення значень мережі за таким алгоритмом:

1. Виокремлення всіх (n-1)-мірних шарів.

2. Для всіх (n-1)-мірних шарів:

2.1. Обчислення суми власних значень вузлів поточного *s*-го шару Sl_s (Sum of layer).

2.2. Якщо сума $Sl_s = 1$, то відбувається формування додаткової навчальної вибірки та представлення її мережі. Вона формується шляхом створення відношень типу «є» для поточного вузла:

$$x_k^i \prec \epsilon \succ x_k^j, i = 1..n,\ j = 1..n, i \neq j,$$

де k – номер поточного вузла,

n – кількість властивостей задачі.

Закінчення навчання полягає у порівнянні стану мережі на попередньому і поточному кроці. Якщо зміни відсутні – то подальша подача навчальних прикладів не буде змінювати значення вузлів і тому навчання мережі можна вважати завершеним.

Висновки до розділу 2

1. Проведено формалізацію логічних задач. Наведено приклад формалізації для «Загадки Ейнштейна».

2. Запропоновано метод автоматичного розв’язання логічних задач на основі навчання спеціальної обчислювальної структури – мережі зв’язків – для підвищення ефективності логічного висновування.

3. Представлено алгоритм навчання мережі зв’язків. Показано на прикладах стан мережі під час навчання.

4. Набув подальшого розвитку підхід до розв’язання логічних задач на основі мережі зв’язків, який враховує умови зв’язку між значеннями властивостей задачі.

3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ МЕРЕЖІ ЗВ'ЯЗКІВ

Для програмної реалізації мережі зв'язків перш за все необхідно було провести аналіз можливості використання обраних технологій і засобів розробки, результати якого представлено у першому підрозділі. У другому підрозділі наведено структуру програмної системи з стислим описом базових компонентів. У третьому підрозділі представлено реалізацію підсистеми розв'язання логічних задач, у четвертому – графічного інтерфейсу. Для можливості обміну даними з іншими програмними системами у п'ятому підрозділі було описано структуру вихідних файлів, а у шостому – механізм зв'язку розробленого обчислювального компоненту з цими системами.

3.1. Технології і засоби розробки програмної системи

Відповідно до поставлених вимог, для реалізації програмного забезпечення була обрана платформа .Net Framework 4.5.2. Створення обчислювальної частини продукту відбувалось мовою C#. Це дозволило реалізувати повторне використання коду шляхом створення бібліотек, які вбудовуються в сторонні додатки, надати програмісту можливість зручного налаштування роботи системи і забезпечити достатньо високий рівень не лише зручності, але й швидкодії та надійності кінцевого продукту. Для графічного інтерфейсу було використано технологію WPF. Це дозволило створити простий і надійний тестовий віконний інтерфейс, який є адаптивним і зручним у користуванні. Для збереження інформації було використано XML.

3.1.1. Visual Studio

Для створення програмного забезпечення використовувалось Visual Studio 2015 [29]. Для начальних цілей дане програмне забезпечення є безкоштовним та

завдяки потужному функціоналу надає молодим спеціалістам можливість створювати програмні продукти високої якості.

Microsoft Visual Studio 2015 Community – це інтегроване середовище розробки програмних систем, створене корпорацією Microsoft (рисунок 3.1).

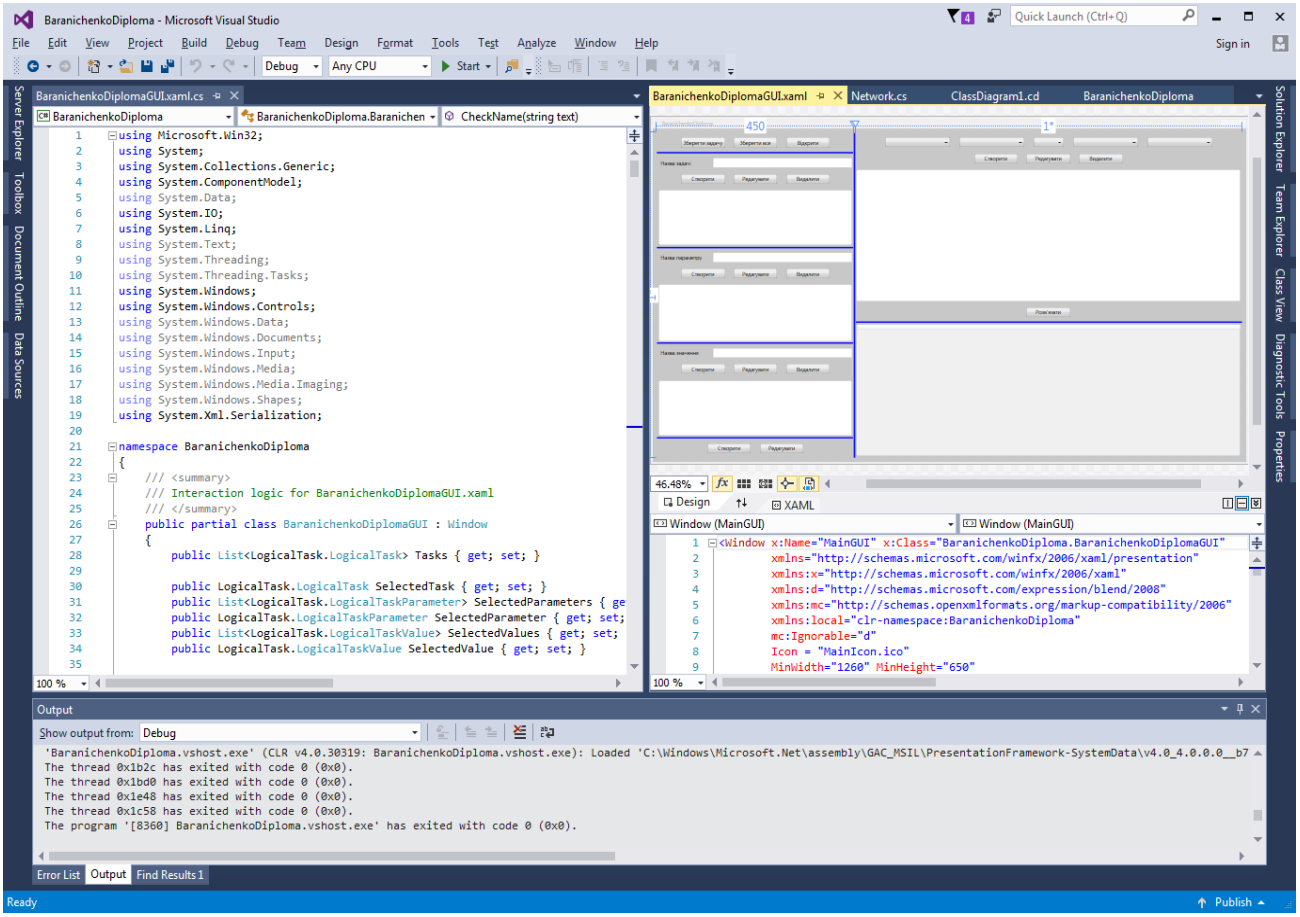

Рисунок 3.1 – Microsoft Visual Studio 2015 IDE

Воно працює під керуванням операційної системи Windows та дозволяє створювати програмні продукти на низькорівневих та високорівневих мовах. Створення продуктів можливе під персональні комп'ютери, планшети, мобільні пристрої. Також існує ряд інструментів для розробки сайтів.

В межах даної роботи було використано такі можливості середовища:

1. Інтелектуальний текстовий редактор для написання коду. Згідно з технічним завданням система розроблялася мовою C#. Під час написання коду

редактор автоматично надає довідку по методам, доповнює відомі лексеми, підсвічує синтаксичні конструкції, тощо.

2. Графічний інтерфейс користувача створювався з використанням технології WPF, тому використовувався дизайнер форм.

3. Git інструментарій. Visual Studio містить вбудовану технологію підтримки версійності програм. Для створення локальних копій версій користувачу не потрібно налаштовувати додаткові програмні комплекси – достатньо лише включити внутрішні інструменти. Оскільки GIT-технології копіюють не повні файли, а лише зміни в них, стає можливим часте створення копій. Також внутрішні механізми GIT гарантують цілісність даних і можливість повернення до попередніх версій незалежно від поточного стану.

Таким чином, використання Visual Studio надає розробнику програмного забезпечення зручний та потужний комплекс інструментів, які пришвидшують роботу і підвищують якість вихідного продукту.

3.1.2. .Net Framework

.Net Framework - це платформа для запуску програмних продуктів, написаних за стандартами .Net [30]. Дана платформа була розроблена притримуючись концепції кросплатформенності програмного забезпечення. Згідно до неї системи, написані за певними визначеними правилами, повинні виконуватись на любому пристрої з даною платформою. Більш того, написання коду можливе не конкретною визначеною мовою, а будь-якою зі списку підтримуваних. Це стало можливим шляхом створення стандартизованої структури (рисунок 3.2).

Перш ніж виконатись, програмний код написаний деякою мовою переводиться в формат Common Intermediate Language. Після цього програма, як правило, виконується базою .Net Framework – Common Language Runtime. Дане середовище використовує спеціальний JIT-компілятор, який транслює проміжний код в процесорні команди. Це дозволяє використовувати один і той же код на системах з різними характеристиками пам’яті, процесору, периферійними пристроями тощо.

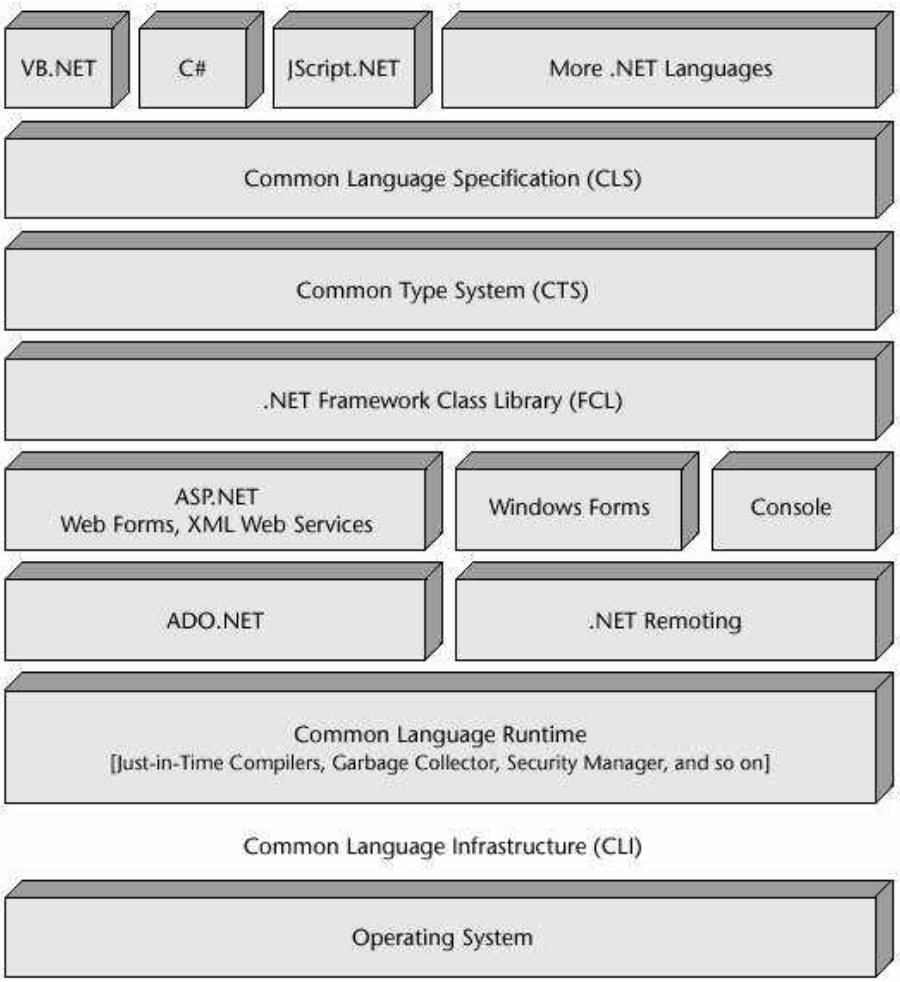

Рисунок 3.2 – Структура .Net Framework

Перевагою даного підходу є те, що програміст на стадії розробки продукту не повинен працювати на апаратному рівні. Це спрощує написання коду і підвищує його якість та надійність, оскільки нівелює необхідність роботи з пам’яттю чи машинними командами.

3.1.3. Windows Presentation Foundation

Windows Presentation Foundation (WPF) – технологія побудови сучасних віконних інтерфейсів, яка, на відміну від відомої Windows Forms, яка використовує GDI/GDI+, що базується на растровому механізмі виведення, використовує DirectX, що базується на векторній графіці (рисунок 3.3) [31]. Це дозволяє робити більш професійні та стабільні у роботі інтерфейси, оскільки вони не залежать від

роздільної здатності пристроїв виведення. Також використання WPF дозволяє підняти швидкодію системи через використання апаратних засобів прискорення.

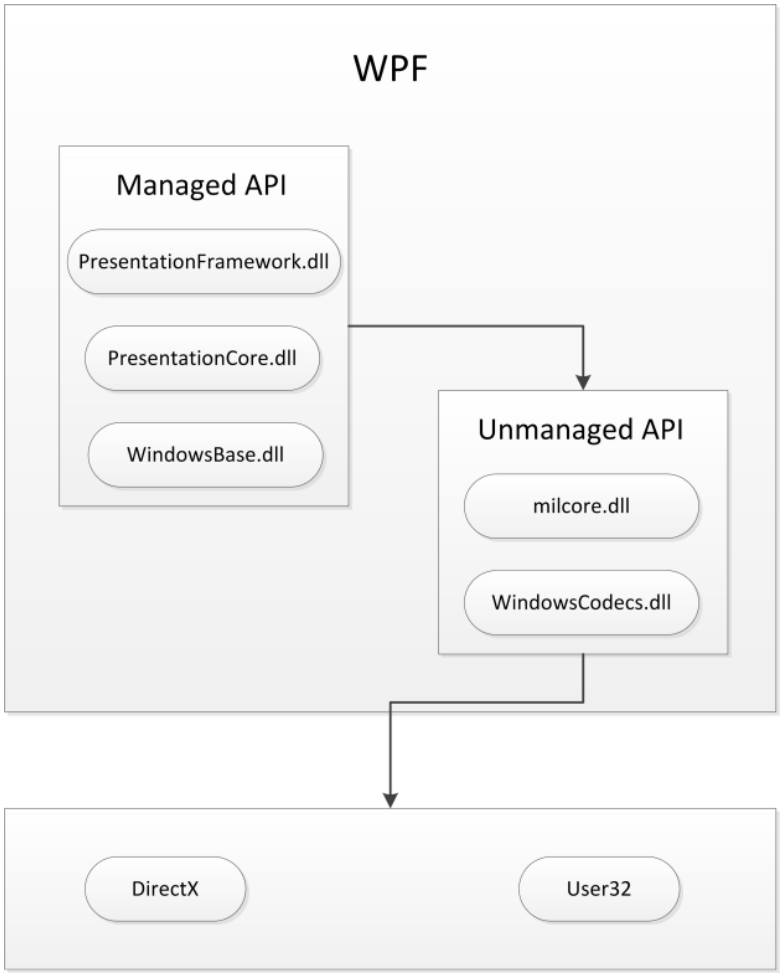

Рисунок 3.3 – Структура WPF

Опис графічних елементів відбувається на мові XAML, що дозволяє не лише створювати елементи керування, але і розширює їх можливості анімацією, стилями, шаблонами, тривимірною графікою, тощо.

Як правило, кожен WPF-елемент має внутрішні механізми зв’язування зовнішніх даних і елементів виведення. Їх використання не лише спрощує відображення даних, але і дозволяє зменшити кількість помилок і збільшити швидкодію через більш низькорівневу організацію.

3.1.4. eXtensible Markup Language

Для збереження задач в файлах було використано мову XML. XML (eXtensible Markup Language) – розширюєма мова розмітки [32]. Структурно XML-документ складається з сутностей, які містять інформацію про об’єкт, що визначається. Сутності в документі зберігаються в деревовидній структурі. Обов’язковим елементом XML-розмітки є кореневий елемент, який може включати в себе інші елементи. Розмітка документів відбувається за допомогою тегів. Теги використовуються парні і складаються з відкриваючого «<Tag Name>» і закриваючого «</Tag Name>». Окрім тегів частина метаданих зберігається у спеціальних інструкціях, наприклад «<?xml version="1.1" encoding="UTF-8" ?>», «<!DOCTYPE greeting SYSTEM "helloworld.dtd">».

Хоч у даному форматі є ряд нюансів, пов’язаних, наприклад, з символами, що використовуються для опису структури документу чи надто великим розміром вихідного файлу, він на сьогоднішній день використовується багатьма системами і технологіями, оскільки має ряд переваг над іншими методами збереження інформації, а саме:

1. Високий рівень читабельності. Інформація не спотворюється при збереженні.

2. Можливість зовнішнього створення розмітки документу. Це надає можливість зв’язувати системи шляхом створення простих трансляторів.

3. Формат підтримує Unicode.

4. Існує можливість збереження рекурсивних структур даних.

5. Має стандартизований синтаксис.

3.2. Структура програмної системи.

Програмне забезпечення складається з набору компонентів, що зображені на рисунку 3.4. Основними частинами системи є обчислювальний механізм та графічний інтерфейс.

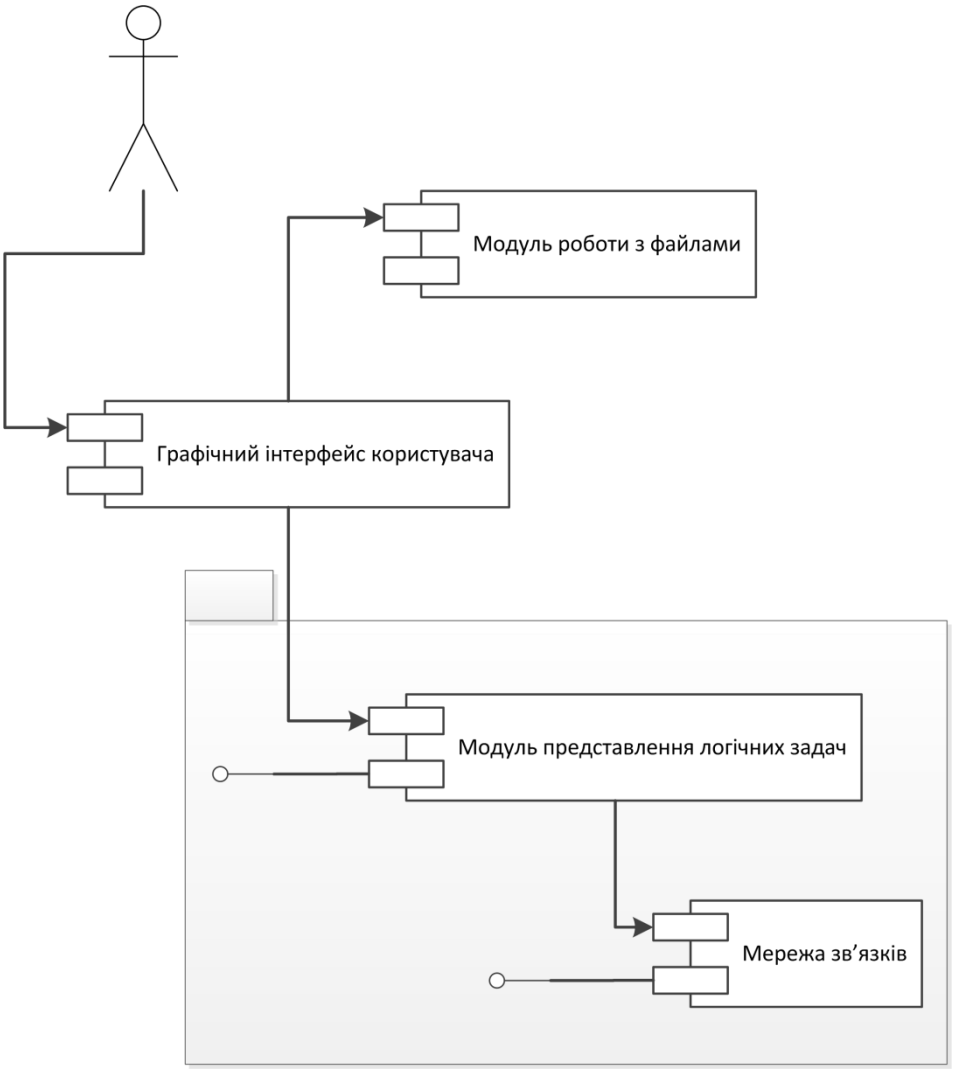

Рисунок 3.4 – Структура програмного забезпечення

Оскільки вирішена задача, як правило, працює в певній прикладній області, необхідно щоб запрограмовані обчислювальні алгоритми було можливо використовувати в прикладних програмних системах. Графічний інтерфейс, в свою чергу, повинен надавати можливість тестувати алгоритми на коректність роботи та доцільність використання в даній прикладній області.

3.3. Підсистема розв'язання логічних задач

Підсистема розв'язання логічних задач структурно складається з логічного представлення задач, модулю розв'язування. Саме ці елементи є базовими у системі,

оскільки реалізують основну її функцію. Представлення задач у системі реалізують набір класів, зображених на рисунку 3.5.

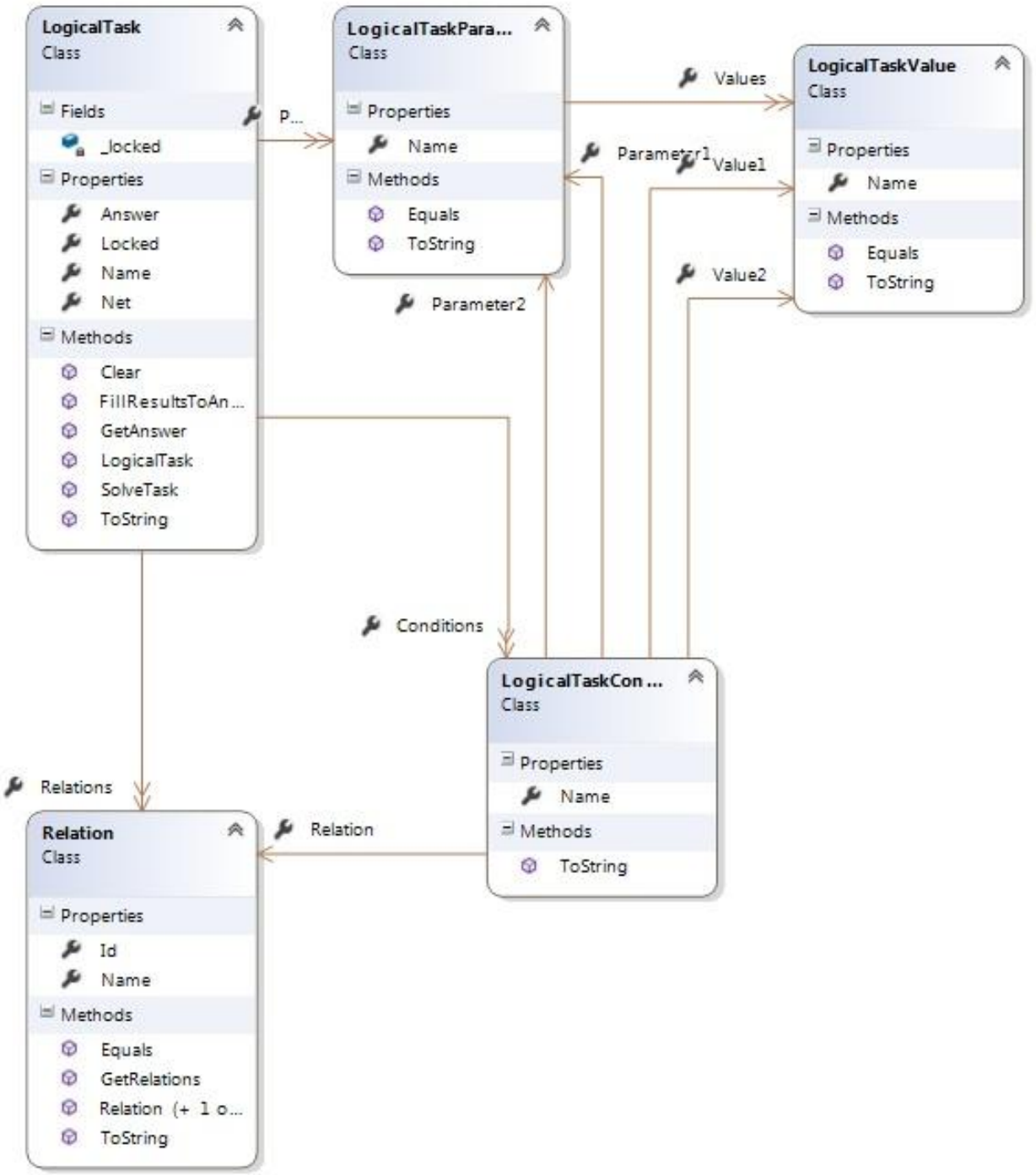

Рисунок 3.5 – Представлення задач у системі

1. LogicalTask. Відповідає за повний опис логічної задачі включаючи відповідь, якщо задача була розв’язана системою. Поле відповіді Answer

заповнюється автоматично системою під час розв’язання задачі. Воно надає повний розв’язок у табличному виді. Поле Net містить посилання на модуль розв’язування задач та за необхідності надає до нього доступ. В ньому міститься інформація про метод розв’язання, поточний стан мережі зв’язків, тощо. Це дозволяє в тестовому режимі перевіряти коректність роботи цього модулю та налаштовувати його параметри. Після передачі в задачу списку властивостей зі значеннями, а також додаткових умов, викликається метод GetAnswer(), який автоматично розв’язує задачу та заповнює поле Answer. Якщо під час розв’язання задачі виявляється, що має місце помилка в її формулюванні, виводиться відповідне повідомлення. Оскільки видалення значень властивостей задачі може призвести до помилки у відношеннях між ними, було реалізовано фіксацію задачі, що відображено властивістю Locked. Після фіксації задачі стає неможливим змінити її властивості та значення. Якщо задача буде розфіксована, то при зміні параметрів прийдеться переналаштовувати і мережу зв’язків.

2. LogicalTaskParameter. Відповідає за опис певної властивості. Всі властивості конкретної задачі зберігаються у ній у вигляді списку.

3. LogicalTaskValue. Відповідає за опис певного значення. Значення конкретної властивості зберігаються у ній в вигляді списку.

4. Relation. Надає можливі варіанти відношень між значеннями параметрів. Для «Загадки Ейнштейна» це відношення «є», «не є», «сусід справа/сусід зліва» та «сусід».

5. LogicalTaskCondition. Відповідає за відношення 2-х значень задачі.

Модуль розв’язування логічних задач складається з таких базових класів, представлених на рисунку 3.6.

1. Node. Описує один вузол мережі. В найпростішому випадку, він складається з поля, яке містить булеве значення сили зв’язків відповідних вхідних значень. Однак в більш складних випадках він може містити, наприклад, імовірнісні чи математичні функції.

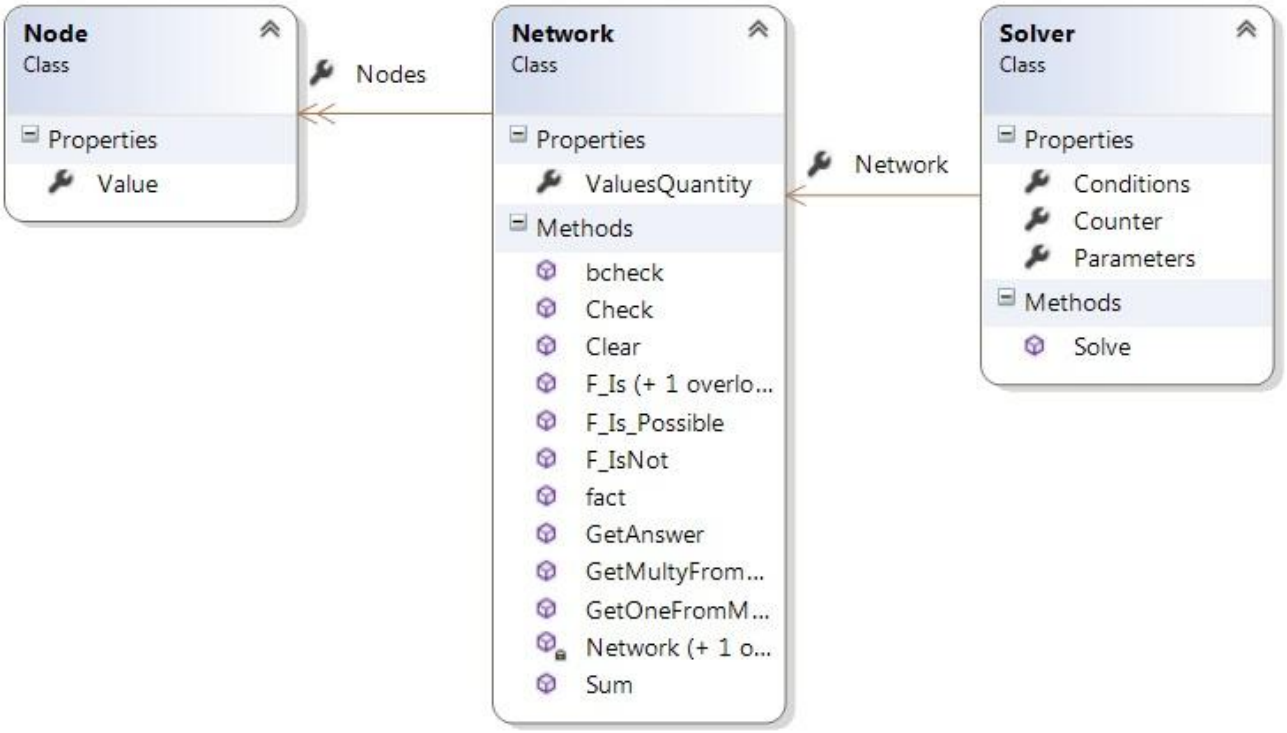

Рисунок 3.6 – Модуль розв’язування задач

2. Network. Описує решітку зв’язків. У полі ValuesQuantity описується інформація про розмірність і розмір мережі, а за допомогою методів F_Is, F_IsNot, F_Is_Possible, Check, GetAnswer реалізуються базові користувацькі функції. За допомогою методу Clear, решітку можна ініціалізувати початковими значеннями, що зручно при повторному розв’язанні задачі.

3. Solver. Роль вхідного обчислювального вузла відіграє класс Solver, який містить дані задачі, які необхідні для обчислень, а також мережу, яка за ці обчислення відповідає. Під час виклику методу Solve() він починає працювати по алгоритму навчання мережі доти, поки навчання не завершиться. Після завершення метод заповнює відповіді у задачі та надає можливість користувачу отримати їх у вигляді таблиці комбінацій.

Розроблена мережа зв’язків з точки зору програмування має вид n-мірного масиву. Оскільки комп’ютерна пам’ять має лінійну структуру, для реалізації n-мірності використовують методи перетворення індексу n-мірного масиву в

одномірний індекс і навпаки. Алгоритм, який перетворює індекс n-мірного масиву в одномірний індекс зображено на рисунку 3.7.

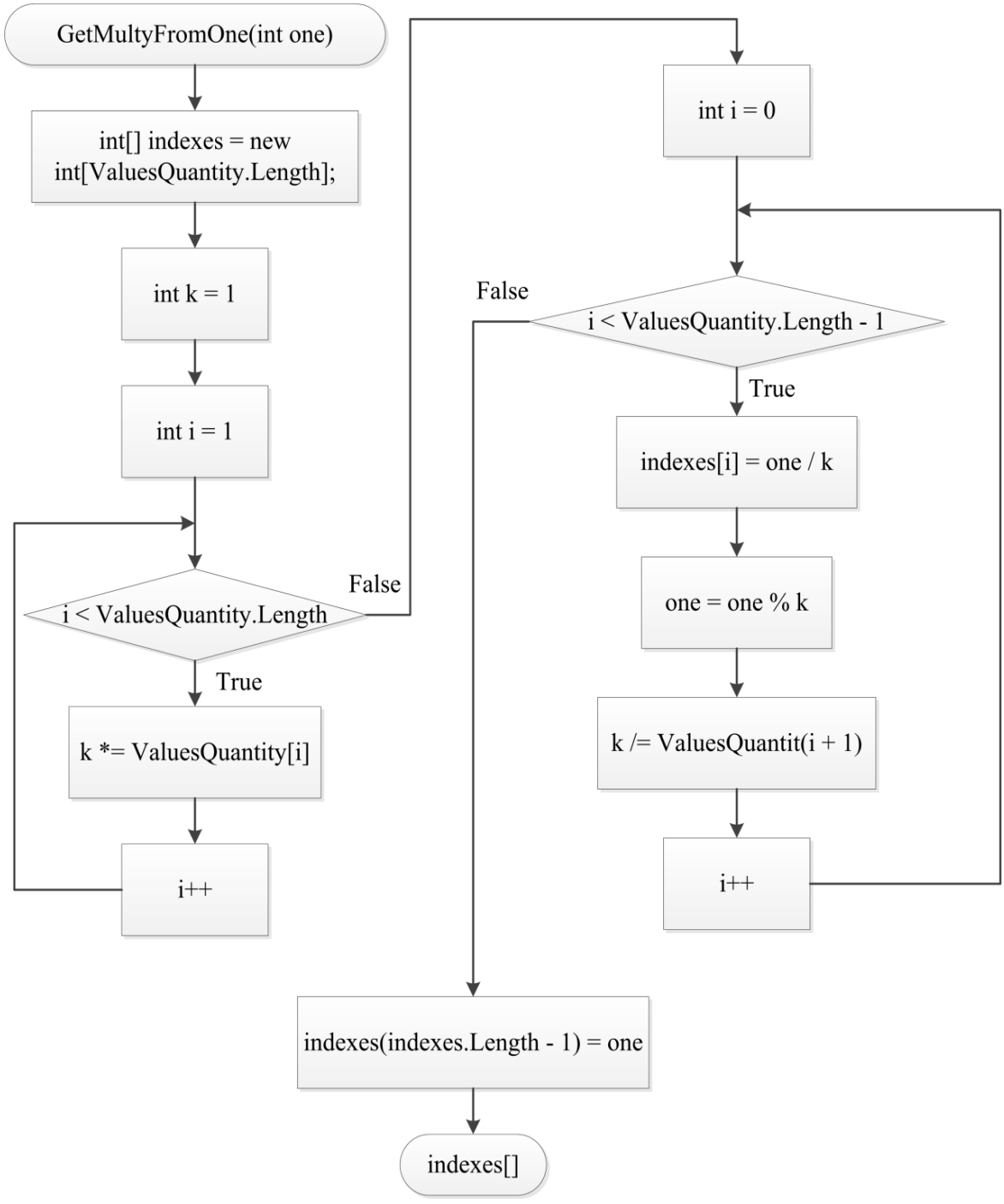

Рисунок 3.7 – Алгоритм перетворення індексу n-мірного масиву в індекс одномірного масиву

Алгоритм, який перетворює одномірний індекс в індекс n-мірного масиву зображено на рисунку 3.8.

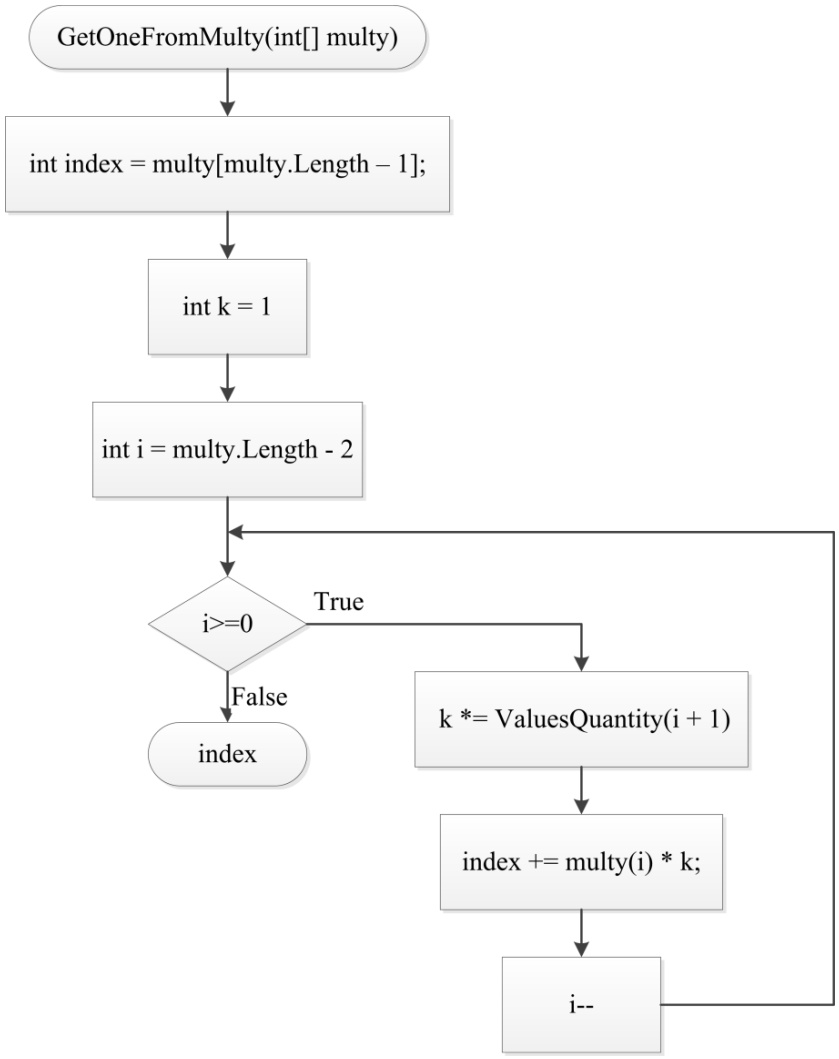

Рисунок 3.8 – Алгоритм перетворення індексу одномірного масиву в індекс n-мірного масиву

Описані вище класи зібрано в окрему бібліотеку. Для використання механізму розв’язання логічних задач при розробці програмного забезпечення необхідно підключити бібліотеку до проекту та викликати функції ініціалізації, навчання та отримання результатів відповідно до алгоритму.

3.4. Графічний інтерфейс користувача

Для задання умов задач та отримання розв’язку було створено графічний інтерфейс користувача, зображений на рисунку 3.9.

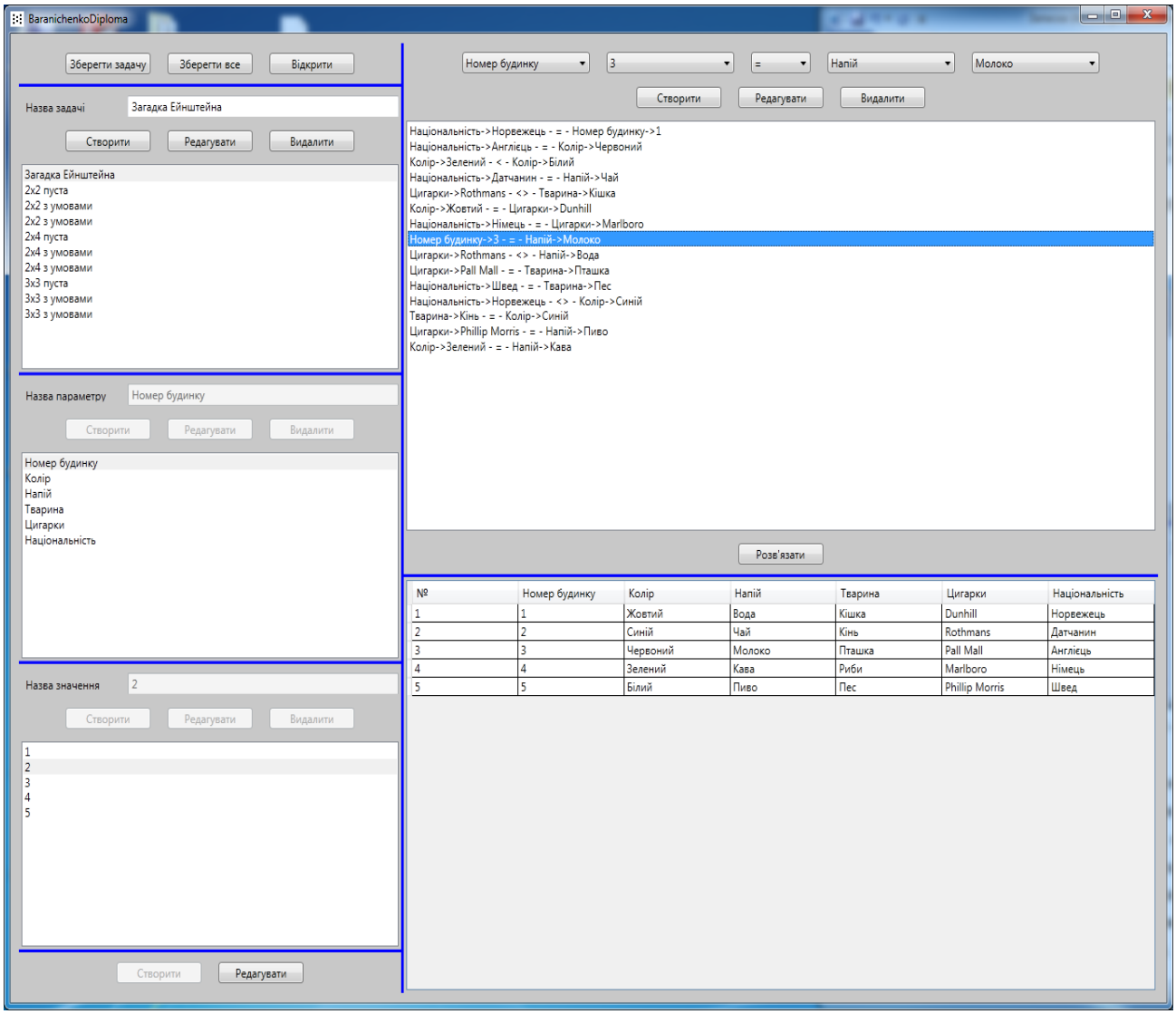

Рисунок 3.9 – Графічний інтерфейс користувача

Він розроблений у вигляді віконного застосунку. Мінімальна ширина вікна через велику кількість елементів керування складає 900px. Умовно інтерфейс можна поділити на зону створення задачі та зону розв’язання. Зона створення задачі (рисунок 3.10) знаходиться в лівій частині робочого вікна та містить елементи керування задачами, їх властивостями і значеннями. Інтерфейс має стандартні, інтуїтивно зрозумілі органи управління, які виконують базові операції з даними: створення, перегляд, редагування, видалення. Для відображення списків зручно використовувати компоненти WPF ListView. Вони дозволяють зв’язувати дані, які містяться в колекціях, з графічним компонентом без необхідності формування додаткової колекції відображення.

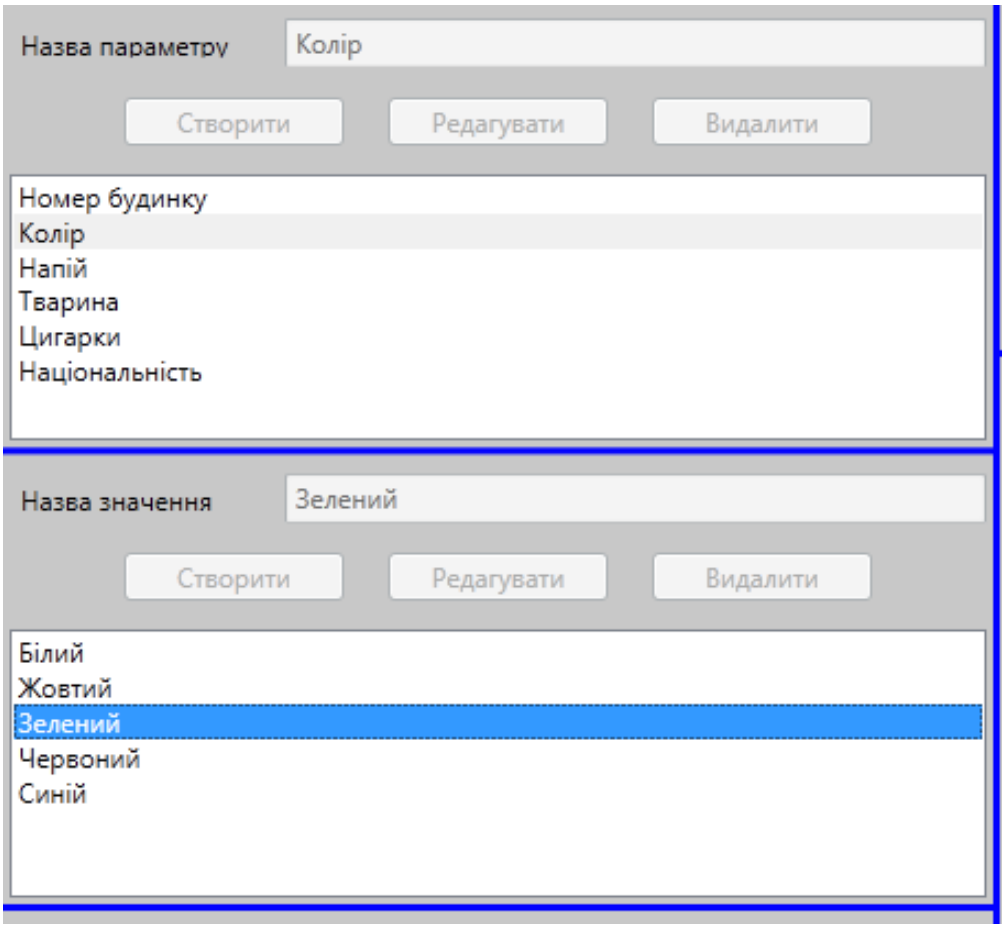

Рисунок 3.10 – Зона створення задачі

Зона розв’язання знаходиться в правій частині робочого вікна. В ній міститься таблиця розв’язків (рисунок 3.11), а також компонент керування умовами задачі (рисунок 3.12), що дозволяє створювати, переглядати, редагувати та видаляти додаткові умови.

№	Номер будинку	Колір	Напій
1	1	Білий	Вода
2	1	Білий	Молоко
3	1	Зелений	Вода
4	1	Зелений	Молоко
5	2	Жовтий	Вода
6	2	Жовтий	Кава
7	2	Жовтий	Молоко
8	3	Білий	Вода
9	3	Білий	Кава
10	3	Білий	Молоко
11	3	Зелений	Вода
12	3	Зелений	Молоко

Рисунок 3.11 – Таблиця розв’язків

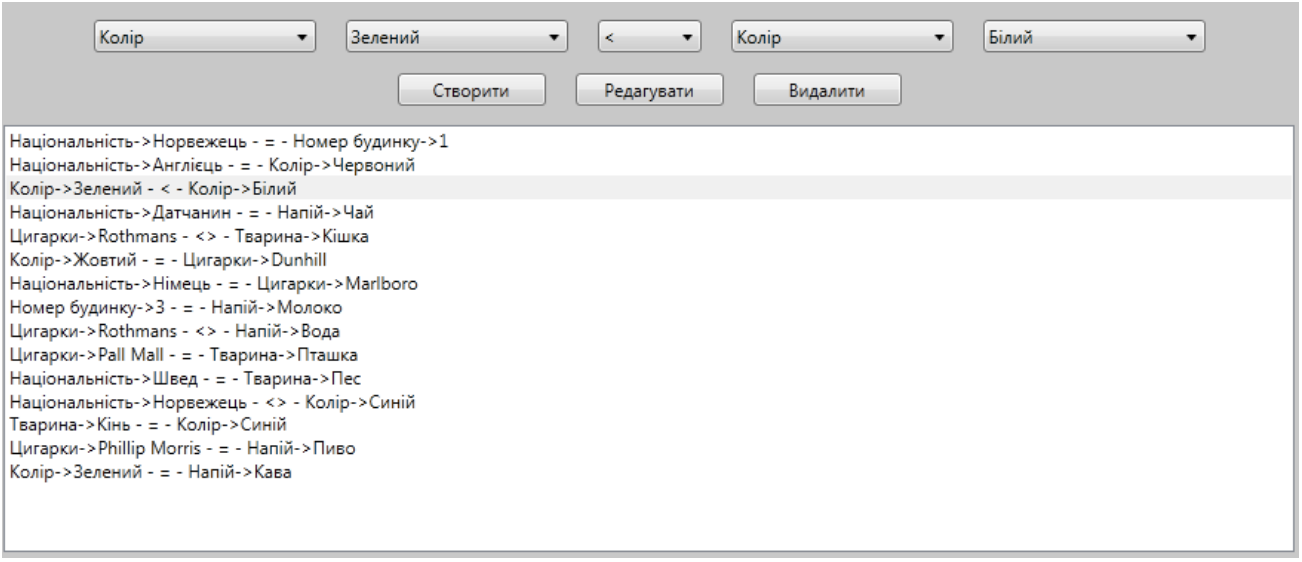

Рисунок 3.12 – Зона створення умов задачі

WPF реалізує шаблон проектування MVVM – «Model-View-ViewmModel», відповідно до якого графічний інтерфейс поділяється на розмітку і контролер. Контролер відповідає за функціональність і не залежить від розмітки. Для реалізації адаптивності інтерфейсу, всі розміточні розміри, такі як ширина і висота блоків, відстань між елементами, було задано у відносних одиницях. Для коректного відображення всіх елементів додатково було встановлено мінімальні розміри елементів, які впливають на мінімальний розмір вікна.

3.5. Структура вхідних та вихідних файлів

Для збереження створених задач і поновлення роботи з ними програма має функції роботи з файлами. В даному випадку це дозволяє не лише виконувати стандартні функції, але і є корисною, коли система, що поставляє задачі є закритою. В цьому випадку обмін файлами можна виконати за таким алгоритмом:

1. Збереження системою-постачальником файлу з задачами.

2. Транслювання збереженого файлу в файл з форматом, призначеним для відкриття розробленою в межах даної роботи системою.

3. Перевірка і завантаження задач в розв’язуючу систему.

Однак для коректного розпізнавання системою файл повинен мати певну структуру. Вона складається з набору задач - «ArrayOfLogicalTask». Кожна задача («LogicalTask») складається з імені («Name»), яке може мати довжину до 100 символів, властивостей («Parameters»), інформації про зафіксованість («Locked»), умов («Conditions»), відповіді («Answer»), зв’язків («Relations») і мережі («Net») (рисунок 3.13).

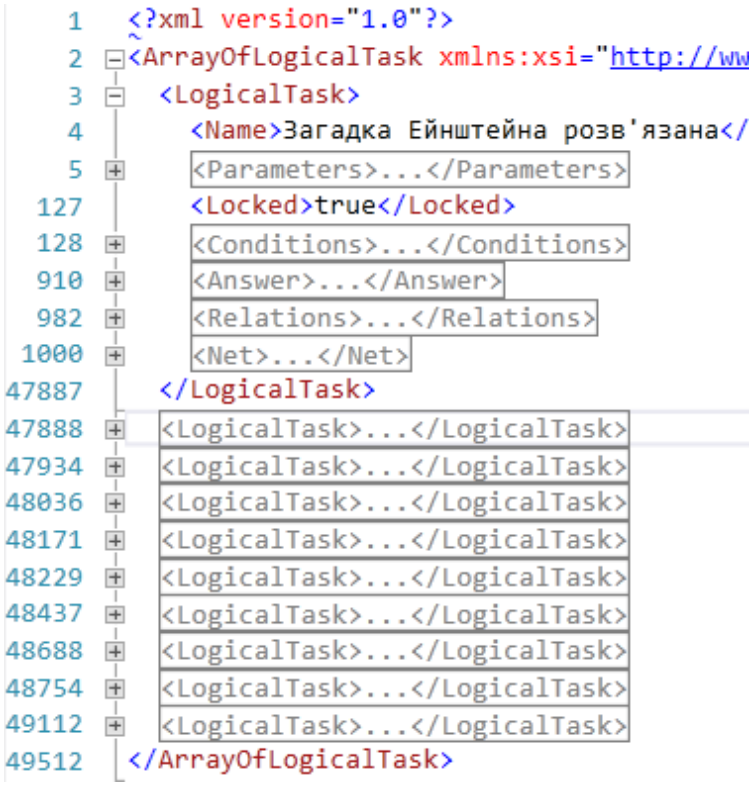

Рисунок 3.13 – Загальна структура файлу з задачами

Властивості задачі Parameters складаються з імені та набору значень Values (рисунок 3.14).

Поле Locked має логічний тип даних та відповідає за закінчення редагування властивостей та значень задачі. Якщо у нього встановлюється логічне значення «true» то відбувається фіксування логічної задачі, після чого необхідно створювати поле умов – Conditions. Якщо у полі Locked встановлене значення «false» то умови видаляються для забезпечення безпеки даних.

```
<Parameters>
  <LogicalTaskParameter>
    <Name>Номер будинку</Name>
    <Values>
      <LogicalTaskValue>
        <Name>1</Name>
      </LogicalTaskValue>
      <LogicalTaskValue>
        <Name>2</Name>
      </LogicalTaskValue>
      <LogicalTaskValue>
        <Name>3</Name>
      </LogicalTaskValue>
      <LogicalTaskValue>
        <Name>4</Name>
      </LogicalTaskValue>
      <LogicalTaskValue>
        <Name>5</Name>
      </LogicalTaskValue>
    </Values>
  </LogicalTaskParameter>
  <LogicalTaskParameter>...</LogicalTaskParameter>
  <LogicalTaskParameter>...</LogicalTaskParameter>
  <LogicalTaskParameter>...</LogicalTaskParameter>
  <LogicalTaskParameter>...</LogicalTaskParameter>
  <LogicalTaskParameter>...</LogicalTaskParameter>
</Parameters>
```

Рисунок 3.14 – Структура властивості задачі

Умови задачі складаються з двох полів вищеописаних властивостей, їх конкретних значень, та відношення між ними (рисунок 3.15).

```
<Conditions>
  <LogicalTaskCondition>
    <Parameter1>
      <Name>Національність</Name>
      <Values>...</Values>
    </Parameter1>
    <Value1>
      <Name>Норвежець</Name>
    </Value1>
    <Parameter2>
      <Name>Номер будинку</Name>
      <Values>...</Values>
    </Parameter2>
    <Value2>
      <Name>1</Name>
    </Value2>
    <Relation>
      <Name>=</Name>
      <Id>0</Id>
    </Relation>
  </LogicalTaskCondition>
```

Рисунок 3.15 – Структура умови задачі

Поля Relations та Net заповнюються програмою автоматично і не є обов’язковими.

3.6. Зв'язок мережі зв’язків з іншими системами

Оскільки розроблений метод розв'язання логічних задач є універсальним, то його програмна реалізація розроблялась з можливістю вбудовування в інші програмні комплекси. Для коректного використання обчислювальних алгоритмів потрібно виконати:

1. Визначення та аналіз логічної задачі.

2. Виокремлення властивостей та їх значень. Для цього необхідно проаналізувати задачу та знайти в ній групи характеристик об'єктів, які необхідно зв'язати. Часто це може бути порядок розташування деяких сутностей. Так, для «Загадки Ейнштейна» це номер будинку, чи іншими словами порядковий номер, та його характеристики.

3. Створення моделі вхідних даних задачі. Отримані властивості та значення необхідно сформувати в колекцію властивостей, кожна з яких має ім’я та колекцію значень (рисунок 3.5).

4. Розроблення операцій створення мережі. Для створення мережі необхідно викликати метод IsTaskCorrect(Task), який перевірить коректність введених даних. Після цього задачу можна заблокувати, присвоївши властивості Locked значення true. На цьому етапі буде створена мережа зв’язків та стане можливим задання умов навчання. Умови навчання повинні формуватися з визначених вище властивостей та значень задачі і додаватися в колекцію. Після повного створення навчальної вибірки вона присвоюється властивості List<LogicalTaskCondition > Task.Coditions.

5. Розроблення операцій навчання мережі та отримання результатів роботи. Для запуску розв’язання задачі необхідно викликати метод ThreadSolve(Task). Якщо навчання відбудеться без помилок – задача автоматично заповниться коректною відповіддю. Якщо під час навчання відбудеться помилка, то дані про неї будуть

виведені у вигляді помилки. Результати роботи програми зберігаються в табличному вигляді у полі DataTable Task.Answer.

Висновки до розділу 3

1. Визначено структуру програмного забезпечення автоматичного розв’язання логічних задач на основі мережі зв’язків.

2. Описано використані технології та засоби розробки.

3. Визначено архітектуру модулю розв’язування задач.

4. Описано графічний інтерфейс користувача та структуру вихідного файлу.

5. На основі запропонованої мережі зв’язків розроблено програмне забезпечення автоматичного розв’язання логічних задач на основі мережі зв’язків.

4. ОБЧИСЛЮВАЛЬНІ ЕКСПЕРИМЕНТИ

Для виконання обчислювальних експериментів було вибрано набір задач, приведений у першому підрозділі. У другому підрозділі було описано хід проведення експерименту, що є основним сценарієм роботи користувача з розробленою системою. Результати експериментів було зведено в таблиці та представлено у вигляді графіку у третьому підрозділі. Однією з задач, яка ставилась до даної роботи, було вбудовування розробленого методу розв’язання логічних задач в систему PowerCAM. Результати виконання даної задачі наведено у четвертому підрозділі.

4.1. Тестові логічні задачі

Тестові задачі охопили всі функціональні частини системи, а тому тестування можна вважати повним та достатнім. Приклади задач, які використовувались у експериментах для визначення характеристик системи зведено в таблицю 4.1.

Таблиця 4.1. Тестові логічні задачі

№	Назва задачі	Формулювання
1	Загадка Ейнштейна	На одній вулиці розташовано 5 будинків різного кольору (синього, білого, жовтого, зеленого і червоного). В цих будинках живе 5 людей різних національностей (швед, данець, англієць, німець та норвежець), які п'ють 5 різних видів напоїв (пиво, кава, чай, вода, молоко), курять 5 різних марок цигарок (Dunhill, Marlboro, Rothmans, Pall Mall та Phillip Morris) та розводять 5 різних тваринок (коні, птахи, рибки, коти та собаки). 1. Норвежець живе в 1 будинку. 2. Англієць живе в червоному будинку. 3. Зелений будинок знаходиться зліва від білого. 4. Данець п'є чай. 5. Той, хто курить Rothmans, живе біля того, хто розводить котів. 6. Той, хто живе в жовтому будинку, курить Dunhill. 7. Німець курить Marlboro. 8. Той, хто живе в центральному будинку, п'є молоко. 9. Сусід того, хто курить Rothmans, п'є воду. 10. Той, хто курить Pall Mall, розводить пташок.

Таблиця 4.1. Тестові логічні задачі (продовження).

		11. Швед розводить собак. 12. Норвежець живе біля синього будинку. 13. Той, хто розводить коней, живе в синьому будинку. 14. Той, хто курить Phillip Morris, п'є пиво. 15. В зеленому будинку п'ють каву.
2	bAbI: two supporting facts	1. Mary moved to the bathroom. 2. Sandra journeyed to the bedroom. 3. Mary got the football there. 4. John went to the kitchen. 5. Mary went back to the kitchen. 6. Mary went back to the garden. Where is the football? - garden
3	Розташування обладнання на виробництві	Існує приміщення, в якому повинні бути встановлені 2 шліфувальних верстати, 3 фрезерних, 4 строгальних та токарний верстати. Окрім цього, є 10 місць, де вони можуть бути розміщені. Однак до їх розмішення висуваються такі вимоги: Токарний верстат знаходиться на місці з номером 1. Фрезерні верстати повинні знаходитися поруч. Строгальні верстати повинні знаходитися поруч. Шліфувальні верстати не повинні знаходитися біля токарних та фрезерних. Визначити можливі варіанти розташування.
4	Розподіл робіт за обладнанням	Існує виробництво, якому необхідно виконати набір з 10 деталей. Кожна деталь поступово повинна пройти певний вид обробки: стругання, фрезерування, токарну обробку, свердління, плоске шліфування, кругле шліфування, шабрення, анодування, полірування та хонінгування. Існує умови порядку робіт, наприклад: Токарна обробка повинна бути перед шліфуванням. Стругання повинно бути перед всіма іншими операціями. Полірування повинно виконуватись останнім. Визначити оптимальний порядок робіт.

Також для тестування системи використовувались стресові задачі з великою кількістю вхідних даних. Вони дозволили перевірити як стабільність роботи системи під навантаженням, так і порівняти швидкодії алгоритмів.

4.2. Сценарій розв’язання логічної задачі

Результати роботи будь-якої системи залежать від апаратних та системних засобів, які використовуються для її роботи. Обчислювальні експерименти розробленого програмного продукту в межах даної роботи проводились на апаратно-програмній системі з такими характеристиками:

1. Процесор Intel Core i3-4030U (1.9 ГГц).
2. RAM 4 ГГб, DDR3L 1600 МГц.
3. Монітор з роздільною здатністю 1920x1080.
4. Встановлена операційна системаWindows 7 Professional 64bit.

Для початку роботи з програмою необхідно створити нову задачу. Для цього потрібно ввести її назву і натиснути кнопку «Створити». Після цього вона створиться та автоматично відобразиться в кінці списку задач. Задачі можна зберігати в окремий файл за допомогою кнопок «Зберегти задачу» та «Зберегти всі». За допомогою кнопки «Відкрити» можна завантажити збережені раніше задачі. Задачі можна редагувати і видаляти відповідними кнопками. Після створення задачі можна аналогічним чином додати її властивості та значення. Після заповнення задачі необхідно натиснути кнопку «Створити» знизу вікна. Після цього відбудеться автоматичне створення мережі зв'язків, а на екрані активується область додання умов. Для додання нової умови необхідно вибрати з випадаючих списків необхідні умови і натиснути нижче розташовану кнопку «Створити». Після створення умови, аналогічно до задач, їх можна редагувати та видаляти. Якщо необхідно змінити оголошення задачі, необхідно натиснути кнопку «Редагувати» розташовану знизу зліва. Зауважимо, що якщо під час створення опису задачі чи умови користувач введе некоректні дані, то йому буде виведене вікно з інформацією про причину помилки (рисунок 4.1).

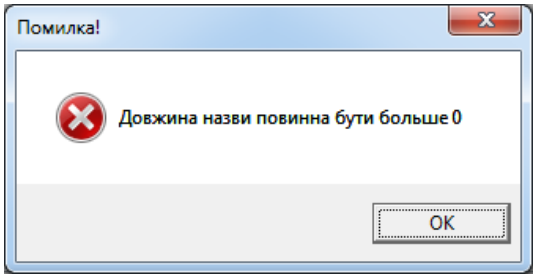

Рисунок 4.1 – Помилка створення задачі

Після створення всіх необхідних умов для розв'язання задачі потрібно натиснути кнопку «Розв'язати». Коли задача буде розв'язана, результати будуть

виведені в нижній правій частині робочого вікна, де їх можна переглянути. За необхідності можна відредагувати умови і розв’язати задачу повторно. Якщо задача має кілька можливих розв’язків, вони будуть представлені у вигляді набору комбінацій, які не суперечать вхідним умовам. Так, наприклад, для «Загадки Ейнштейна» з видаленою першою умовою, результат роботи програми буде мати вигляд, зображений на рисунку 4.2.

№	Номер будинку	Колір	Напій	Тварина	Цигарки	Національність
1	1	Жовтий	Вода	Кішка	Dunhill	Норвежець
2	1	Жовтий	Вода	Риби	Dunhill	Норвежець
3	1	Жовтий	Вода	Пес	Dunhill	Швед
4	1	Жовтий	Чай	Кішка	Dunhill	Датчанин
5	1	Жовтий	Чай	Риби	Dunhill	Датчанин
6	1	Зелений	Кава	Кішка	Marlboro	Німець
7	1	Зелений	Кава	Кішка	Rothmans	Норвежець
8	1	Зелений	Кава	Пташка	Pall Mall	Норвежець
9	1	Зелений	Кава	Риби	Marlboro	Німець
10	1	Зелений	Кава	Риби	Rothmans	Норвежець
11	1	Зелений	Кава	Пес	Rothmans	Швед
12	1	Червоний	Вода	Кішка	Rothmans	Англієць
13	1	Червоний	Вода	Пташка	Pall Mall	Англієць
14	1	Червоний	Вода	Риби	Rothmans	Англієць

Рисунок 4.2 – Результат розв’язання «Загадки Ейнштейна» з видаленою першою умовою

Даний метод відображення результатів дозволяє переглянути і уточнити умови для отримання єдино можливого розв’язку.

Після завершення роботи програми її стан зберігається у файлі LastState.xml, який знаходиться в каталозі зі встановленим програмним забезпеченням. Під час наступного запуску, файл буде автоматично зчитано і робота продовжиться. Якщо файл буде відсутній, програма автоматично заповниться тестовими прикладами для різних задач.

4.3. Результати розв’язання логічних задач на мережі зв’язків

Результати розв’язання логічних задач було оброблено і зведено в таблицю 4.2. Окрім задач, які в ній представлено система проходила тестування і на інших задачах. Однак дані задачі є найбільш показовими та значущими для порівняння.

Таблиця 4.2. Результати розв’язання логічних задач.

№	Назва задачі	Параметри задачі			Час розв’язання мережею, с	Час розв’язання перебором, с	Кількість епох	Кількість коректних комбінацій
		n	m	Кількість умов				
Ресурсні задачі для тестування коректності роботи								
1	Загадка Ейнштейна	6	5	15	8.71	~80	6	5
2	Розміщення за столом	3	4	8	1.12	~10	3	4
3	Розміщення за столом	4	3	10	1.53	~10	3	3
4	bAbI, two supporting facts	2	3	2	0.76	~5	3	3
5	bAbI, two arguments relations	2	3	3	0.83	~5	3	3
6	bAbI, positional reasoning	3	4	6	1.78	~10	4	4
Стресові задачі								
7	Задача №1	5	10	10	~70	>300	4	128
8	Задача №2	10	5	10	~60	>300	3	158
9	Задача №3	5	5	50	~100	>300	2	24
Прикладні задачі								
10	Задача розташування обладнання №1	2	10	25	4.76	~70	2	20
11	Задача розташування обладнання №2	3	20	15	7.54	~90	3	20
12	Задача розподілу робіт №1	2	10	5	2.4	~40	2	10
13	Задача розподілу робіт №2	2	20	10	5,71	~70	2	20
14	Логістична задача №1	2	20	10	5.08	~45	3	20
15	Логістична задача №1	3	30	10	8.25	~120	3	30

Порівняння ефективності роботи на перших 6 тестових задачах наочно представлено на рисунку 4.3 у вигляді діаграми.

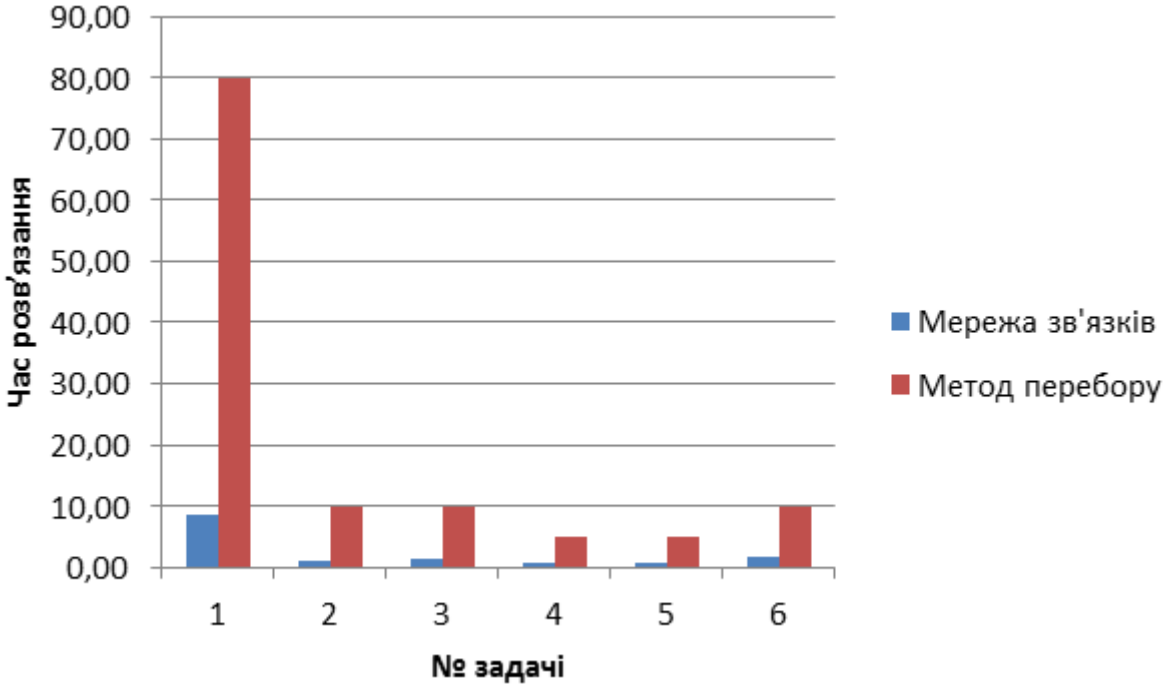

Рисунок 4.3 – Порівняння швидкості роботи методів

Як видно з діаграми, розроблений спосіб розв’язання логічних задач на відміну від стандартного методу перебору має на порядок вищу швидкість роботи. Це обумовлено тим, що він не перебирає послідовно як коректні, так і хибні варіанти, а, фактично, одразу сканує всі можливі варіанти і відкидає хибні. На коректні і потенційно коректні варіанти створюються додаткові навчальні вибірки і відбувається донавчання. Метод перебору має проблему з надто великою глибиною рекурсивного спуску, що часто призводить до некоректної роботи програм при збільшенні кількості об’єктів. Створений метод має аналогічну проблему. Вона полягає в необхідності зберігання мережі зв’язків. Це потребує використання надто великі розмірності даних. Однак цю проблему можна обходити шляхом оптимізації алгоритму та використання надвеликих структур даних.

4.4. Використання мережі зв’язків у системі PowerCAM

Окрім теоретичних задач було розв’язано декілька реальних прикладних задач. Це довело можливість використання як розробленого методу розв’язання

логічних задач, так і створеної програмної системи для вирішення промислових задач. Одна з таких задач полягала у визначенні оптимального розташування операцій обробки деталі на верстаті з ЧПК (числовим програмним керуванням). Для реалізації цього, в систему PowerCAM 2.1 було вбудовано розроблений метод розв’язання логічних задач (рисунок 4.4), налаштування якого було винесено в вкладку «Settings» у вигляді кнопки «Checker Settings».

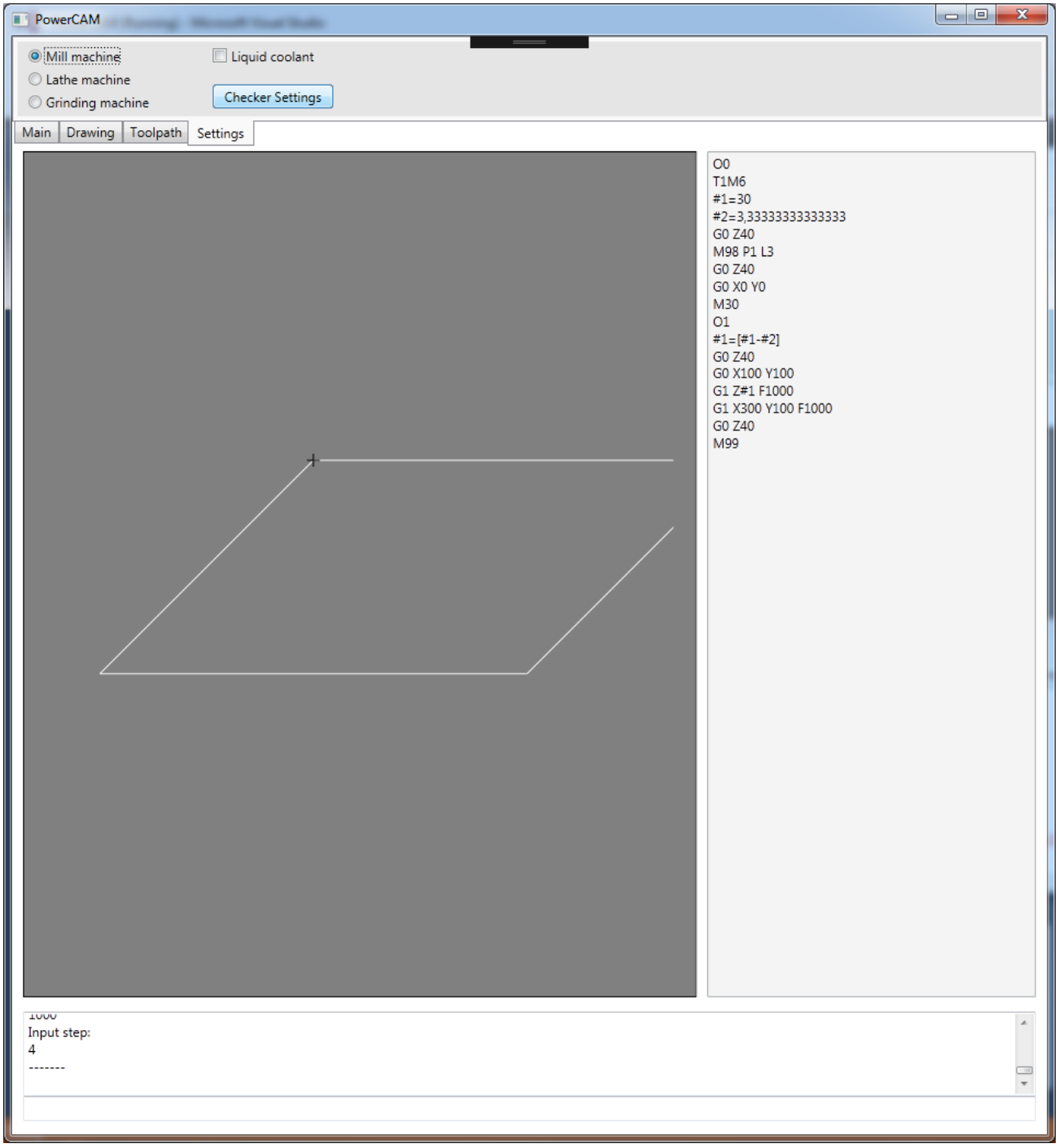

Рисунок 4.4 – Система PowerCAM з вбудованим аналізатором коду

PowerCAM – це автоматизована система, призначена для підготовки керуючих програм верстатів з ЧПК (числовим програмним керуванням) для невеликих виробництв [33]. До її функціональних можливостей відносяться:

1. Створення траєкторій обробки для фрезерних, токарних, шліфувальних, строгальних верстатів.

2. Створення простих креслень у вбудованому CAD-модулі.

3. Постпроцесування траєкторій мовою G-code.

Система розроблена мовою C# на платформі .Net Framework 4.5.2, що робить можливим вбудовування розробленого обчислювального механізму розв'язання логічних задач.

Першою властивістю логічної задачі, що ставить система PowerCAM, є типи створених траєкторій. Другою властивістю виступає їх порядок. Деякі умови задачі наведено на рисунку 4.5.

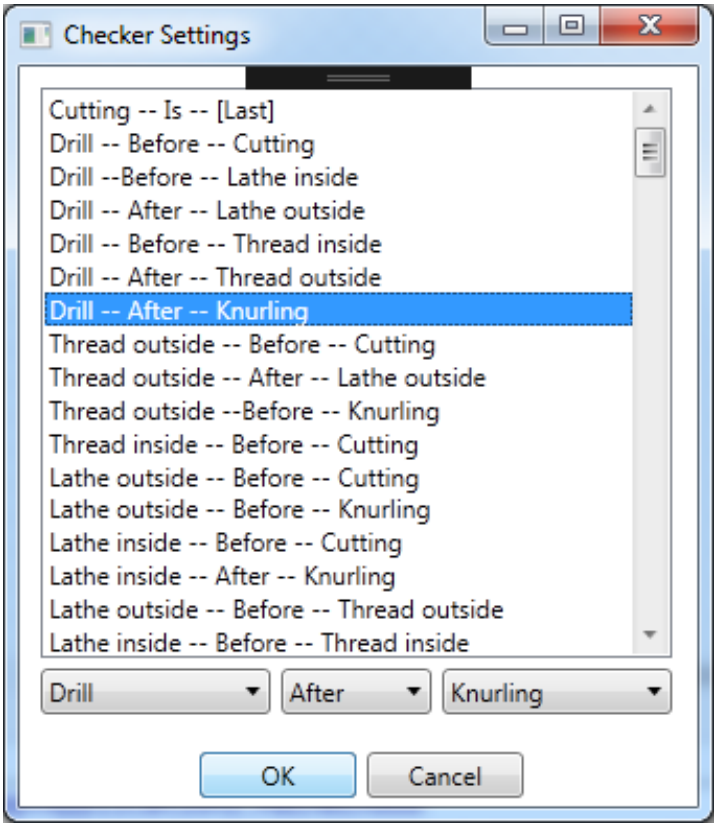

Рисунок 4.5 – Додаткові умови системи PowerCAM

Під час генерації G-коду відбувається розв’язання задачі визначення оптимального порядку траєкторій обробки і перевіряється з тим, який задав програміст. Якщо вони співпадають, то йому не виводиться ніяких повідомлень. Однак в випадку, коли оптимальний порядок не співпадає з порядком, заданим програмістом, програма виводить попередження і пропонує змінити порядок (рисунок 4.6).

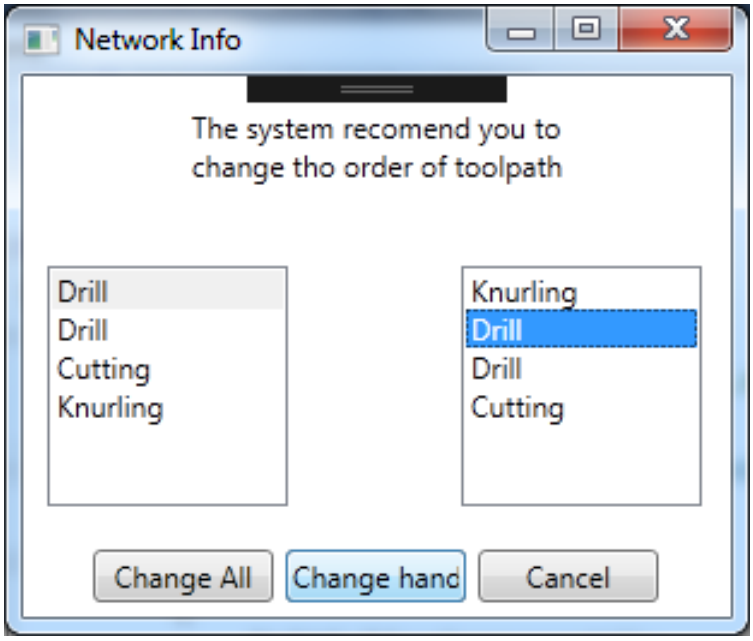

Рисунок 4.6 – Попередження мережі зв’язків

По статистиці використання, через механічні помилки програміст верстату в середньому створює 1% некоректного коду. Хоч ця цифра і мала, подібні помилки можуть призвести не лише до виходу з ладу дорогого обладнання, але й стати причиною травмування персоналу. Це однозначно визначає доцільність використання додаткових засобів комп’ютерного контролю коректності створення керуючого коду.

Результати обчислювальних експериментів показали шляхи подальших досліджень у сфері розв’язання логічних задач:

1. Створення інтелектуального лексично-синтаксичного аналізатора для автоматичного виокремлення властивостей задачі, їх значень та відношень між ними.

2. Створення машини виводу на основі вимог тесту Тьюрингу, тобто генерування відповідей на природній мові.

3. Оптимізація алгоритму для розширення його можливостей та пришвидшення роботи.

4. Розробка web-інтерфейсу. Це дозволить додатково тестувати розроблений метод на задачах різних прикладних областей.

5. Розробка інтерфейсів для виконання обчислень на GRID-мережах, оскільки великі задачі неможливо розв’язати на персональних комп’ютерах.

6. Введення імовірнісної складової комбінацій.

Висновки до розділу 4

1. Представлено тестові задачі: теоретичні зі спеціальних ресурсів та прикладні виробничі.

2. Описано сценарій роботи користувача з системою.

3. Наведено результати розв’язання задач.

4. Виконано порівняння розробленого методу з типовим методом перебору.

5. Представлено систему PowerCAM з вбудованим механізмом попередження помилок на розробленому механізмі обчислення.

ВИСНОВКИ

У роботі було запропоновано метод машинного розв’язання логічних задач, який полягає у навчанні по задачі спеціальної обчислювальної структури – мережі зв’язків. Для цього було:

1. Проведено аналіз досліджень в області розв’язання логічних задач. Визначено, що задачі типу зв’язку параметрів є одними з найбільш важливих як на науковому, так і на прикладному рівнях. Проаналізовано існуючі ручні та автоматичні методи розв’язання логічних задач визначеного типу. Визначено, що ручні методи раціонально застосовувати лише для невеликих задач. Для великих та складних задач необхідно використовувати машинні методи. Найбільш повно охоплює дану проблему метод, реалізований на базі обрахування сум. Однак його недоліками є складність формалізації та зміни задачі, що в кінцевому рахунку відображається на складності програмної реалізації. Визначено актуальність та необхідність досліджень в даному напрямі.

2. Розроблено формалізацію логічних задач за їх властивостями, значеннями та зв’язками між ними. Створено спеціальну обчислювальну структуру – мережу зв’язків, яка має вид n-мірної решітки. Її розмірність дорівнює кількості властивостей задачі, а розмір по кожній вісі – кількості значень відповідної властивості. Наведено приклади побудови двомірних та тримірних мереж зв’язків для розв’язання задач з двома та трьома властивостями відповідно. Описано алгоритм навчання мережі. Наведено приклади навчання для двомірної мережі зв’язків.

3. На основі актуальних засобів розробки та технологій створено програмне забезпечення для розв’язання логічних задач. Структурно воно поділяється на дві базові частини: обчислювальний модуль розв’язання логічних задач та графічний інтерфейс. Перша частина дозволяє не лише розв’язувати задачі в межах створеного продукту, але й використовувати механізм в інших програмних системах. Графічний

інтерфейс надає можливість тестування раціональності і можливості використання методу для конкретних прикладних областей.

4. Проведено обчислювальні експерименти системи на визначених тестових задачах. Результати зведено в таблицю. Виконано порівняння створеного способу розв’язання логічних задач з методом перебору. Визначено, що він працює швидше, ніж зазначений метод. Показано прикладне застосування обчислювального механізму в системі PowerCAM та доведено раціональність його використання в системах технологічної підготовки виробництв.

5. Визначено подальші напрями досліджень:

1) створення лексично-синтаксичного аналізатору та машини виводу для інтелектуалізації системи;

2) розробка web-застосунку та реалізація методу для GRID-мереж.

3) оптимізація створеного методу, розширення його функціональності шляхом додавання імовірнісної складової.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Carroll L. The game of logic / Lewis Carroll. – London: Macmillan and Co, 1886. – 198p.
2. Zebra Puzzles [Electronic resource]. – 2018. – Access mode: https://www.brainzilla.com/logic/zebra/.
3. The Zebra Puzzle – A Classic Logic Puzzle [Electronic resource]. – 2018. – Access mode: https://escapetheroomz.com/the-zebra-puzzle-a-classic-logic-puzzle/.
4. Einstein's riddle and grid puzzles [Electronic resource]. – 2012. – Access mode: http://brainden.com/einsteins-riddles.htm.
5. Einstein Puzzles [Electronic resource]. – 2018. – Access mode: https://www.mathsisfun.com/puzzles/einstein-puzzles-index.html.
6. The bAbI project [Electronic resource] // Facebook research. – 2016. – Access mode: https://research.fb.com/downloads/babi/.
7. Сафиуллина Л. Методы решения логических задач [Електронний ресурс] / Лейсан Сафиуллина. – 2013. – Режим доступу до ресурсу: https://www.mindmeister.com/ru/161070700/.
8. Browne, C. 2013: Deductive search for logic puzzles. Computational Intelligence in Games (CIG), 2013 IEEE Conference on: IEEE; P. 1-8
9. Кадымова Ф. Н. Методы решения логических задач / Ф. Н. Кадымова. // Стерлитамакский Филиал Башкирского Государственного Университета. – 2018. – С. 23–28.
10. Caiming X. Dynamic coattention networks for question answering / X. Caiming, Z. Victor, S. Richard. // Salesforce Research. – 2017. – P. 25–39.
11. Singh A. Deep Learning for Visual Question Answering [Electronic resource] / Avi Singh. – 2015. – Access mode: https://www.kdnuggets.com/2015/11/deep-learning-visual-question-answering.html/.

12. Zebra puzzle [Electronic resource]. – 2019. – Access mode: https://rosettacode.org/wiki/Zebra_puzzle#.22Manual.22_solution_.28Norvig-style.29.
13. Komendantskaya E. SHERLOCK — A Neural Network Software for Automated Problem Solving / E. Komendantskaya, Q. Zhang. // School of Computing. – 2012. – P. 152–164.
14. Wladston F. Solving the Zebra Puzzle with Boolean Algebra [Electronic resource] / Filho Wladston. – 2017. – Access mode: https://code.energy/solving-zebra-puzzle/.
15. Application of “Einstein's riddle” in solving construction machine allocation problems / B. Dasović, M. Čorak, M. Galić, U. Klanšek. // e-gfos. – 2016. – P. 12–22.
16. Рогожкин И. Б. Конструктивная Геометрия / И. Б. Рогожкин. // Компьютер и мы. – 1997. – №7. – С. 12–14.
17. Julian S. Y. Solving “Einstein's Riddle” Using Spreadsheet Optimization / Scott Yeomans Julian. // Institute for Operations Research and the Management Sciences. – 2019. – P. 55–63.
18. D Crabtree, J.; Zhang, X. 2015: Recognizing and Managing Complexity: Teaching Advanced Programming Concepts and Techniques Using the Zebra Puzzle, Journal of Information Technology Education: Innovations in Practice, 14 P. 171-189.
19. Gregor, M.; Zábovská, K.; Smataník, V. 2015: The zebra puzzle and getting to know your tools. Intelligent Engineering Systems (INES), 2015 IEEE 19th International Conference on: IEEE; pp. 159-164.
20. Patterson, M. C.; Harmel, B.; Friesen, D. 2007: A spreadsheet solution to Einstein's Riddle, International Journal of Technology, Policy and Management, 7 (1), pp. 49-67
21. Hindriks, K.; Vromans, J. 2007: Design and Evaluation of Formal Representations: An Incremental Approach using Logic Grid Puzzles. In: Geertzen J, Thijsse E, Bunt H, editors. International Workshop on Computational Semantics - IWCS’07. Tilburg, The Netherlands P. 334–338.

22. Hindriks K. : An Incremental Approach using Logic Grid Puzzles / K. Hindriks, J. Vromans. // International Workshop on Computational Semantics. – 2006. – №7. – С. 334–338.
23. Solving Logic Puzzles: From Robust Processing to Precise Semantics / I.Lev, B. MacCartney, M. D. C, R. Levy. // Proceedings of the 2nd Workshop on Text Meaning and Interpretation. – 2004. – №4. – P. 48–56.
24. Tomasi C. Zebra Puzzle / Tomasi. // Computer Science, Duke University. – 2010. – P. 59–64.
25. D'Avila Garcez A. Neural-Symbolic Cognitive Reasoning / A. d'Avila Garcez, L. C. Lamb, D. M. Gabbay. – London: Cognitive Technologies, 2009. – 198P.
26. Wang J. Hybrid Markov Logic Networks / J. Wang, P. Domingos. // Department of Computer Science and Engineering University of Washington. – 2008.
27. Lloyd J. W. Logic for Learning: Learning Comprehensible Theories from Structured Data / Lloyd. – New York: Springer-Verlang, 2003. – 75 P.
28. Holldobler S. Approximating the Semantics of Logic Programs by Recurrent Neural Networks / S. Holldobler, Y. Kalinke, H. Storr. // Applied Intelligence. – 1999. – №11. – P. 45–58.
29. Visual Studio IDE [Electronic resource] // Microsoft. – 2019. –Access mode: https://visualstudio.microsoft.com/ru/?rr=https%3A%2F%Fwww.google.com.
30. Общие сведения о платформе .NET Framework [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview.
31. Getting Started (WPF) [Electronic resource]. – 2018. – Access mode: https://docs.microsoft.com/en-us/dotnet/framework/wpf/getting-started/.
32. Уэйтли К. Основы XML для начинающих пользователей [Електронний ресурс] / Кей Уэйтли – 2017. – Режим доступу до ресурсу: https://www.ibm.com/developerworks/ru/library/x-newxml/index.html.
33. Шаповалова С. І. Вдосконалення САМ-систем для невеликих виробництв / С. І. Шаповалова, О. М. Бааніченко. // Міжвідомчий науково-технічний збірник «Адаптивні системи автоматичного управління». – 2017. – №1. – С. 189–197.

ДОДАТОК А

Акт впровадження

Автоматичне розв’язання логічних задач

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ3269_19М

Аркушів 2

2019

Затверджую

Заступник декана ТЕФ

КПІ ім. Ігоря Сікорського

_____ В.А. Кондратюк

«___»_____ 2019 р.

АКТ ВПРОВАДЖЕННЯ

результатів дисертаційної роботи Бараніченко Олексія Миколайовича «Автоматичне розв'язання логічних задач»

Нами, викладачами кафедри автоматизації проектування енергетичних процесів і систем (АПЕПС) КПІ ім. Ігоря Сікорського, даний акт складено про те, що результати дисертаційної роботи Бараніченко Олексія Миколайовича використано при проведенні пошукових досліджень за ініціативною темою «***Інтелектуальна обробка графічної інформації***» (Державний реєстраційний номер 0117U006081, керівник д.т.н. Аушева Н.М.) і, зокрема, впроваджено в навчальний процес при викладанні дисципліни «Сучасні технології розроблення програмного забезпечення 1: програмування систем штучного інтелекту».

професор каф. АПЕПС, д.т.н., доцент _____ Н.М. Аушева

доцент каф. АПЕПС, к.т.н., доцент _____ С. І. Шаповалова

ДОДАТОК Б

Публікації

Автоматичне розв’язання логічних задач

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ3269_19М

Аркушів 12

2019

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

1898

№ 1'(30) 2017

ISSN 1560-8956

Адаптивні Системи Автоматичного Управління

Міжвідомчий науково-технічний збірник

ISSN: 2522-9575

Міжвідомчий науково-технічний збірник «Адаптивні системи автоматичного управління» № 1' (30) 2017

УДК 004.42::621.9

С.І. Шаповалова, О.М. Бараніченко

ВДОСКОНАЛЕННЯ CAM-СИСТЕМ ДЛЯ НЕВЕЛИКИХ ВИРОБНИЦТВ

Анотація: В статті визначено характеристики CAM-систем і критерії їх оптимізації для застосування на невеликих виробництвах. Представлено розроблену за цими критеріями систему PowerCAM. Наведено приклад роботи PowerCAM для порівняння з аналогічними системами.

Ключові слова: CAM-система, керуюча програма верстатів з ЧПК.

Вступ

На теперішній час використання верстатів з числовим програмним керуванням (ЧПК) можливе не лише для великих підприємств, але і для невеликих виробництв. Однак використання технології ЧПК при малих обсягах виробництва має специфічні риси. Тому існує ряд проблем, які необхідно вирішувати для продуктивної і стабільної роботи обладнання з ЧПК. Це насамперед проблема створення ефективного коду керуючих програм, оскільки використання існуючих програмних засобів автоматизованої системи технологічної підготовки виробництва ускладнюється або наявністю надлишкового функціоналу та високою ціною, або неоптимальністю побудови вихідного коду.

Саме тому задача створення простого в застосуванні, зручного, надійного та потужного засобу програмування верстатів з ЧПК є актуальною та має практичне застосування для невеликих виробництв.

Аналіз останніх досліджень

Програмним забезпеченням (ПЗ), яке створює керуючі програми для верстатів з ЧПК, є CAM-системи (computing aided manufacturing). Сучасні дослідження в галузі розвитку CAM-систем [1-5] спрямовані на розв'язання специфічних задач генерації складних траєкторій обробки, які рідко застосовуються на невеликих виробництвах.

В роботі Л. А. Кашуби і О. В. Чекалкіна [6] було сформовано базові принципи побудови ПЗ саме для невеликих виробництв. Результатом досліджень стало створення CAM-системи ADEM. В роботі T. Merser [7] було сформовано принципи відбору ПЗ верстатів з ЧПК для невеликих

виробництв. Аналіз існуючих CAM-систем, більшість з яких використовуються досі, відбувався за такими факторами, як ефективність, функціональність, ціна, розширяємість, тощо. Однак існуючої реалізації ПЗ, яка би однозначно задовольняла вимогам невеликих виробництв знайдено не було.

В статті А. Ю. Солкіна та П.К. Кузнєцова [8] було розглянуто не менш важливу проблему ЧПК – оптимізацію вихідного коду керуючої програми. Однак, з однієї сторони, розглянуті засоби оптимізації не завжди доцільно використовувати для малосерійного виробництва через надлишковий функціонал, а з іншої – для CAM-систем з обмеженим функціоналом необхідно забезпечити оптимальність вихідного коду з точки зору мінімальної кількості команд. Тому дослідження з визначення функціональних характеристик CAM-систем та критеріїв оптимізації коду керуючих програм, необхідне для розробки засобу програмування верстатів з ЧПК, пристосованого саме для невеликих виробництв.

Ціллю статті є визначення характеристик CAM-систем і критеріїв їх оптимізації та представлення розробленої за цими критеріями системи PowerCAM для невеликих виробництв.

Структура ПЗ для роботи на верстатах з ЧПК

Основними задачами ПЗ верстатів з ЧПК є:

1. Створення креслення або моделі деталі, яку необхідно виготовити. Ці задачі вирішують CAD системи, наприклад, AutoCAD, SolidWorks, Inventor.

2. Створення керуючої програми верстату, що дозволяє задати по кресленню обробку деталі. Ці задачі вирішують CAM системи, наприклад, ArtCAM, SolidCAM, MasterCAM, RhinoCAM.

3. Керування верстатом. Ці задачі вирішують спеціалізовані керуючі системи, які можуть бути вбудовані в комп'ютер верстата. Однак, як правило, вбудовані системи зустрічаються лише на професійному обладнанні. Для верстатів, які зазвичай використовуються на невеликих виробництвах, використовується окремий комп'ютер, на якому встановлено керуючу програму. До таких програм можна віднести Mach3, vriCNC, NC-Studio.

Таким чином, базова структура ПЗ верстатів з ЧПК складається з CAD-системи, CAM-системи та керуючої системи. Але в деяких ви-

 ISSN 1560-8956

падках для керування та програмування верстатів з ЧПК використовують додаткові програмні засоби, такі як:

1. Програми ручного редагування вихідного коду, наприклад, NC corrector.

2. Програми візуалізації роботи верстату при обробці по заданому коду.

3. Програмно-апаратні системи керування верстатом, які не потребують окремого комп'ютера і містять операції для роботи і налагодження верстатів.

Використання CAD систем є недоцільним, якщо виконуються такі умови:

1. Креслення готового виробу не існує в електронному вигляді.

2. Операції обробки є простими, код яких можна створити навіть без CAM-системи.

До простих операцій можна віднести, наприклад, свердління та розточку отворів, стругання шпоночних пазів, фрезерування по прямолінійним траєкторіям, токарну обробку циліндрів та нарізання різьби, тощо. Фактично, до таких операцій можна віднести більшість операцій, які виконуються на універсальних верстатах з ручним керуванням. В цих випадках вигідніше замість повноцінної CAD-системи використовувати вбудований в CAM-систему модуль створення креслень. Детальне креслення для створення цих траєкторій обробки недоцільне тому, що воно буде надлишковим (наприклад в випадку з різьбою), або містити тільки невелику кількість параметрів, які доцільніше задавати під час створення траєкторії (наприклад, для свердління на кресленні задаються лише центр отвору та його глибина).

Від оптимальності коду керування верстатом з ЧПК залежить коректна робота обладнання і, як наслідок, його довговічність і якість вихідної продукції. Існуючі CAM-системи або вузькоспеціалізовані та містять недостатній функціонал, або складні і дорогі в використані та мають надлишковий функціонал в контексті використання на невеликих виробництвах.

Характеристики CAM-систем і критерії їх оптимізації

За результатами досліджень засобів створення керуючої програми було виокремлено характеристики CAM-систем. В таблиці 1 наведено приклади таких систем та відповідних значень характеристик. Для

представлення обрано характерні приклади складної системи Master-CAM X6, простої – ArtCAM Pro 9, а також розробленої PowerCAM.

Таблиця 1

Характеристики CAM-систем

№	CAM-система / Характеристика	Зміст характеристики	MasterCAM	ArtCAM	PowerCAM
1	Можливість створення траєкторій обробки для верстатів різних типів	Підтримка фрезерних, токарних, свердлильних, розточних верстатів	Так	Ні	Так
2	Наявність простих траєкторій	Фрезерування площини та торця, проточка конуса, свердління, розточка, тощо	Так	Так	Так
3	Наявність складних траєкторій	Траєкторії багато осьової обробки	Так	Ні	Ні
4	Можливість параметричного стилю написання G-коду	Використання підпрограм, циклів, параметрів	Так	Ні	Так
5	Можливість ручного редагування коду	Можливість ручного редагування коду	Так	Ні	Так
6	Швидкість введення параметрів, пар./с.	Значення середнього арифметичного введених параметрів за 1 секунду	0.05	0.2	0.3

Швидкість введення параметрів тестувалися для простих траєкторій, які подібні для всіх систем. Покращення цього показника зумовлено тим, що:

1. Відсутня необхідність введення параметрів, які не впливають на генерацію коду.

2. Використовується командний рядок для введення/виведення даних за аналогом до AutoCAD.

 ISSN 1560-8956

Таким чином, можна сформувати вимоги до вдосконаленої CAM-системи, які втілені в PowerCAM:

1. Мінімальний набір траєкторій: фрезерування площини та торця, проточка конуса, свердління, розточка.

2. Типи верстатів, для яких можливе створення керуючих програм: фрезерні, токарні, свердлильні та розточні.

3. Можливість створення простих креслень безпосередньо в CAM-системі.

4. Швидке і зрозуміле для користувача створення керуючих програм.

5. Інтуїтивно-зрозумілий інтерфейс з подібним шаблоном для всіх операцій.

6. Відкриття креслень формату .dxf.

7. Створення керуючих програм на мові G-/M-коду.

Крім цього система PowerCAM розроблювалась з врахуванням таких критеріїв оптимальності побудови коду керуючої програми:

1. Мінімальна кількість команд для побудови траєкторії.

2. Заміщення багаторазового задання коду ідентичних операцій командою виконання циклу.

В рамках проекту автоматизовано такі процеси:

1. Створення креслень.

2. Створення траєкторій обробки.

3. Генерація коду керуючої програми.

Система PowerCAM реалізована на мові C#, платформа .Net Framework 4.6.

Основною ціллю PowerCAM є створення оптимального коду керуючих програм.

Структура PowerCAM

PowerCAM містить підсистему роботи з кресленнями, підсистему створення траєкторій обробки, постпроцесор.

Підсистема роботи з кресленнями реалізує такі функції:

1. Створення плоских графічних об'єктів, таких як:

1.1. Точка. Задається координатами X, Y.

1.2. Відрізок. Задається початок і кінець відрізку (дві точки).

1.3. Коло. Задається центр кола (точка) і радіус (число).

1.4. Дуга. Задається твірним колом, початковим і кінцевим кутами (2 числа).

2. Відкриття існуючих креслень формату .dxf.

Підсистема створення траєкторій обробки охоплює такі траєкторії:

1. Свердління. Задаються координати свердління, глибина свердління, необхідність стружколомної зупинки та виведення свердла, швидкість свердління.

2. Розточка. Задаються ті самі параметри, що і для свердління, але обробка повинна бути циклічною та мати зупинку для виміряння та уточнення розміру.

3. Фрезерування площини. Найбільш широко при цьому використовується метод фрезерування «Зигзаг», при якому фрезерування йде вздовж однієї з осей при подальшому зміщенні в напрямі іншої, або метод «Зміщення», при якому обробка йде вздовж периметру зі зміщенням від/до периметру. Для цього задається контур, глибина фрезерування, крок поперечного зміщення, глибина обробки, крок глибини, швидкість обробки.

4. Прямолінійне фрезерування, яке часто використовується для відрізки деталей, прорізки пазів, тощо. Задаються ті самі параметри, що і для фрезерування площини, окрім зміщення.

5. Фрезерування по контуру. При цьому задаються такі ж параметри, як і для прямолінійного фрезерування, а також контур обробки.

6. Токарна обробка циліндру. Задаються подача, швидкість різання, діаметр обробки, крок обробки.

7. Токарна обробка канавок. Задаються ті самі параметри, що і для обробки циліндру, окрім подачі.

8. Нарізання різьби. Задаються швидкість різання, діаметр обробки, крок обробки та крок різьби, напрямок різьби.

9. Проточка конусу. Задаються подача, швидкість різання, діаметри конусу та його довжина (або ж конусність), крок обробки.

Для всіх цих операцій також можуть бути включені додаткові параметри, такі як увімкнення/вимкнення подачі змазуючо-охолоджуючої рідини, налаштування обертів шпинделя.

Постпроцесор для верстата з ЧПК – це програмний модуль, який призначений для перетворення керуючої траєкторії обробки в керуючу програму. Постпроцесор системи PowerCAM генерує код керуючої програми мовами G-/M-коду. При цьому він містить вбудовані операції обробки коду керуючої програми, такі як цикл, створення змінних, додавання блоку коду, написаного вручну.

 ISSN 1560-8956

Реалізація системи PowerCAM

На рисунку 1 представлено вікно PowerCAM в момент створення програми обробки деталі.

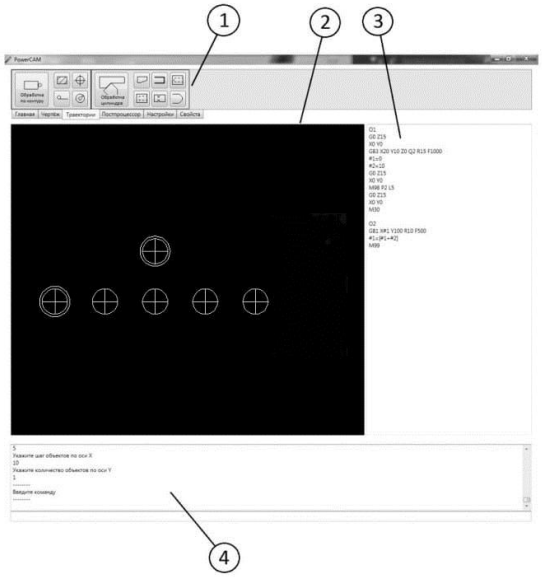

Рис. 1. Головне вікно системи PowerCAM

Виноски відповідають базовим компонентам графічного інтерфейсу користувача:

1. Програмне меню реалізоване у вигляді стрічки за аналогією до AutoCAD, SolidWorks. Стрічкове меню базується на панелях інструментів, розділених вкладками. Воно займає мало місця, не створює проблеми багаторівневого вкладення, надає швидкий доступ до внутрішніх елементів.

2. Зона креслення призначена для перегляду та вибору графічних примітивів, а також траєкторій обробки, що задаються в програмі.

3. Панель G-/M-коду відображає код створених траєкторій обробки, а також надає можливість створювати цикли і дописувати код вручну.

4. Командна панель реалізована за аналогією до відповідного компоненту у AutoCAD і є більш високорівневим аналогом консолі. Вона дозволяє виконувати будь-яку з функцій налаштування системи, створення графічних об'єктів або траєкторій, редагування коду керуючої програми. При цьому командна панель надає однотипний простий інтерфейс для введення/виведення даних, що пришвидшує роботу та зменшує ймовірність введення некоректних даних.

Такий набір графічних компонентів дозволяє:

1. Забезпечити максимальний розмір робочого простору.

2. Прискорити роботу розробника за рахунок ефективного доступу до функцій системи через командний рядок.

3. Співставляти код з траєкторією обробки під час його створення.

На рисунку 2 винесено фрагменти коду керуючих програм, створені системами ArtCAM (рис. 2а) та запропонованою PowerCAM (рис. 2б). Код задає обробку деталі з одним отвором глибиною 15 мм, що обробляється з виведенням свердла, і 5 отворів глибиною 10 мм, розташованих з кроком 10 мм по вісі X.

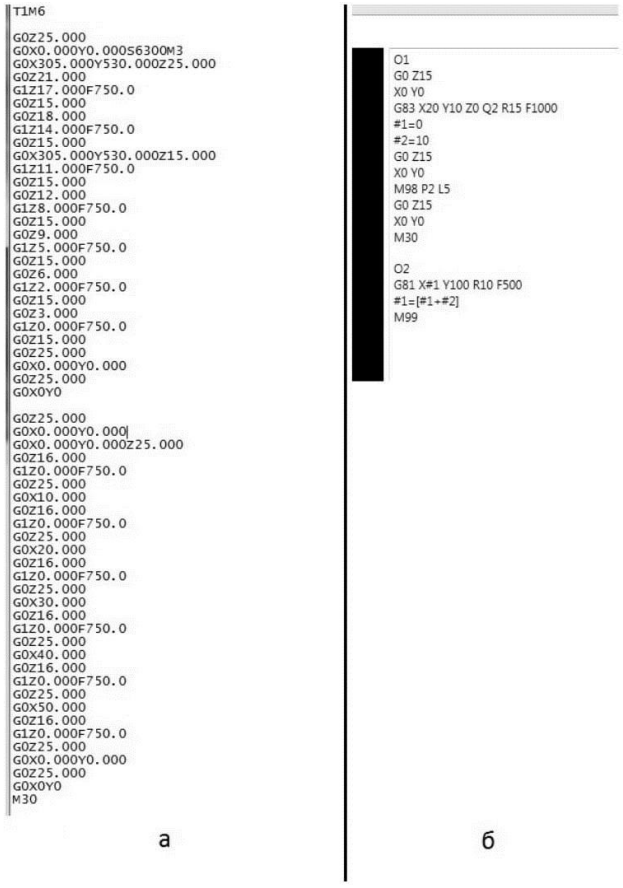

Рис. 2. Фрагменти коду керуючих програм: а) ArtCAM, б) PowerCAM

Тестування системи проводилось під керуванням операційної системи Microsoft Windows 7 Ultimate 64bit, та системою керування Mach3. Отримані результати доводять, що код створений PowerCAM коротший і простіший для використання та редагування.

 ISSN 1560-8956

Висновки

1. Проведено дослідження існуючих засобів створення керуючих програм верстатів з ЧПК. Виокремлено основні характеристики CAM-систем і критерії їх оптимізації.

2. Запропоновано систему PowerCAM, вдосконалену набором функцій, представленням графічного інтерфейсу користувача та оптимізацією генерації керуючого коду.

3. Наведено приклад роботи PowerCAM для порівняння з аналогічними системами.

Література

1. Пивоваров В. И. Современные цифровые технологии изготовления зубных протезов / В. И. Пивоваров, Е. С. Бондарь, И. П. Рыжова. // Саратовский научно-медицинский журнал. – 2011. – №1.

2. Маданов А. В. Анализ технологической подготовки производства авиационных деталей сложной геометрии на станках с числовым программным управлением / А. В. Маданов. // Известия Самарского научного центра Российской академии наук. – 2014. – №1.

3. Разработка технологии изготовления изделий ракетно-космической техники с применением CAD/CAM-систем / [П. С. Попов, А. Ю. Литвинчук, К. Г. Пасечник та ін.]. // Актуальные проблемы авиации и космонавтики. – 2010.

4. Кузнецов Є. Ю. Использование CAM-систем для составления управляющих программ многопроходного нарезания витков червяков / Є. Ю. Кузнєцов, А. С. Ямников. // Известия Тульского государственного университета. Технические науки. – 2013. – №8.

5. Подготовка управляющих программ с помощью cad/cam систем / И. А.Бондарев, А. В. Никитин, С. Ю. Сыроежко, Н. А. Амельченко. // Актуальные проблемы авиации и космонавтики. – 2011. – №7.

6. Кашуба Л. А. CAD/CAM Adem наилучшая отечественная система для мелкосерийного производства / Л. А. Кашуба, О. В. Чекалкин. // Известия Южного федерального университета. Технические науки. – 1998. – №2.

7. Merser T. CAD/CAM selection for small manufacturing companies / Tim Merser. // The Graduate College University of Wisconsin. – 2000.

8. Солкин А. Ю. Проблема оптимизации NC программ для механообрабатывающих станков с ЧПУ / А. Ю. Солкин, П. К. Кузнецов. // Вестник Волжского университета им. В.Н. Татищева. – 2011. – №17. – С. 90–94.

ДОДАТОК В

Апробації

Автоматичне розв’язання логічних задач

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ3269_19М

Аркушів 16

2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XVI Міжнародної науково-практичної конференції аспірантів, магістрантів і студентів м. Київ, 24-27 квітня 2018 року,

ТОМ 2

Київ-2018

Доц., к.т.н.

РОЗВ'ЯЗАННЯ ЛОГІЧНИХ ЗАДАЧ НЕЙРОННИМИ МЕРЕЖАМИ

При реалізації штучного інтелекту все частіше використовуються нейронні мережі. Вони апробовані в прикладних програмних системах і показали задовільні результати для прийняття рішень в задачах, що добре формалізуються, наприклад, в іграх з повною інформацією. Однак проблема розв'язку задач, що потребують логічних висновків, і досі є актуальною і в подальшому може мати практичне значення в багатьох сферах науки та техніки.

В роботі [1] наведено класифікацію підходів розв'язання логічних задач без застосування нейронних мереж. Загальним недоліком цих підходів є низька швидкість виведення висновку. В останні роки виокремився новий напрям досліджень для розв'язання таких задач нейронними мережами. Komendantskaya та Zhang запропонували систему Sherlock [2], яка дозволяє транслювати правила логічної програми (Prolog) в характеристики нейросимвольної мережі (neuro-symbolic networks) або об'єднаної мережі індуктивного навчання та логічного програмування (Connectionist Inductive Learning and Logic Programming – CILP). Робота [3] присвячена розв'язанню логічних головоломок за допомогою вдосконалених нейронних мереж з пам'яттю (Memory Neural Network – MemNN). Автори розв'язують прості комбінаторні головоломки з ресурсу The bAbI project [4]. Однак наявні вдосконалення не забезпечують розв'язання задач, що потребують послідовних логічних висновків (наприклад, задачі Positional Reasoning, Path Finding). Тому необхідним є вдосконалення нейронних мереж для розв'язання саме таких задач.

Висновки:

1. Проведено огляд архітектур існуючих нейронних мереж, застосування яких можливе для розв'язання логічних задач.
2. Проведено аналіз публікацій з результатами ефективності виведення логічних висновків нейронними мережами.
3. Визначено загальноприйняті тестові приклади логічних задач (benchmarks) та отримано навчальні вибірки до них.
4. Визначено напрям досліджень для вдосконалення нейронних мереж з пам'яттю для забезпечення задовільного розв'язання задач, що потребують послідовних логічних висновків.

Перелік посилань:

1. Лейсан С. Методы решения логических задач [Електронний ресурс] / Сафиуллина Лейсан. – 2012. – Режим доступу до ресурсу: https://www.mindmeister.com/ru/161070700/_.

2. Kommendskaya E. SHERLOCK — A Neural Network Software for Automated Problem Solving [Electronic resource] / E. Kommendskaya, Z. Qiming // School of Computing, University of Dundee, UK. – 2013. – Mode of access: http://staff.computing.dundee.ac.uk/katya/QK.pdf.

3. Weston J. Towards AI-complete questions answering : a set of prerequisite toy asks[Text] / J.Weston, A. Bordes, S. Chopra, A.M. Rush, B. van Merrienboer, A. Joulin, T. Mikolov// - Conference paper at ICLR, 2016.-P.14.

4. The bAbI project [Electronic resource] / [H. Miller, S. Chopra, M. Ranzato та ін.] // Facebook research. – 2015. – Mode of access: https://research.fb.com/downloads/babi/.

Сучасні проблеми наукового забезпечення енергетики: Матеріали XVI Міжнародної науково-практичної конференції аспірантів, магістрантів і студентів, м. Київ, 24–27 квітня 2018 р. У 2 т. – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2018. – Т. 2. – 298 с.

ISBN 978-966-622-886-7

ISBN 978-966-622-888-1 (Т. 2)

Подано тези доповідей XVI Міжнародної науково-практичної конференції аспірантів, магістрантів і студентів «Сучасні проблеми наукового забезпечення енергетики» за напрямами: атомна енергетика, теплообмін і гідродинаміка в теплопередаючих пристроях і енергетичних установках, сучасні технології в тепловій енергетиці, проблеми теоретичної і промислової теплотехніки.

Друкується в авторській редакції за рішенням Вченої ради теплоенергетичного факультету Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (протокол № 8 від 26 березня 2018 р.)

ISBN 978-966-622-886-7
ISBN 978-966-622-888-1 (Т. 2)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XVII Міжнародної науково-практичної конференції молодих вчених та студентів м. Київ, 23-26 квітня 2019 року,

ТОМ 2

Київ- 2019

УДК 620.9(062)+621.311(062)
С91

Сучасні проблеми наукового забезпечення енергетики: Матеріали XVII Міжнародної науково-практичної конференції молодих вчених та студентів, м. Київ, 23–26 квітня 2019 р. У 2 т. – К. : КПІ ім. Ігоря Сікорського, 2019. – Т. 2. – 195 с.

ISBN 978-966-622-937-6
ISBN 978-966-622-939-0 (Т.2)

Подано тези доповідей XVII Міжнародної науково-практичної конфе-ренції молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики» за напрямами: автоматизація теплоенергетичних процесів, геометричне моделювання та проблеми візуалізації, програмне забезпечення інформаційних систем та мережних комплексів, моделювання та аналіз теплоенергетичних процесів, сучасні проблеми сталого розвитку енергетики.

Для викладачів вищих навчальних закладів, наукових працівників, аспірантів та студентів технічних спеціальностей.

Друкується в авторській редакції за рішенням Вченої ради теплоенергетичного факультету Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського»
(протокол № 8 від 25 березня 2019 р.)

ISBN 978-966-622-937-6
ISBN 978-966-622-939-0 (Т.2)

УДК 004.032.26:004.832

Магістрант 6 курсу, гр. ТВ-71мн Бараніченко О.М.
Доц., к.т.н. Шаповалова С.І.

МАШИННЕ НАВЧАННЯ ДЛЯ РОЗВ'ЯЗАННЯ ЛОГІЧНИХ ГОЛОВОЛОМОК

При випробуванні механізмів логічного висновування найбільш розповсюдженими тестами є комбінаторні головоломки типу «Задача Ейнштейна». Однак на сьогоднішній день відсутні програмні рішення розв'язання таких задач, які є простими у створенні, використанні та підтримці, мають високу ефективність, забезпечують можливість розширення або часткової заміни умов задачі. Саме тому створення такого алгоритму та його комп'ютерна реалізація є актуальним, має наукове та прикладне значення і потребує ґрунтовного дослідження.

Для розв'язання таких задач насамперед використовуються мови логічного програмування, наприклад, Prolog. Інший підхід базується на машинному навчанні. Прикладами є робота [1], в якій представлено мережу «Sherlock» для вирішення логічних задач на основі парадигми об'єднання індуктивного навчання і логічного програмування (CILP – Connectionist Inductive Learning and Logic Programming); робота [2], призначена для розв'язання логічних задач методами булевої алгебри; робота [3], в якій здійснена спроба створення уніфікованого алгоритму розв'язання логічних задач шляхом використання лінійної алгебри.

В даній роботі було запропоновано обчислювальний засіб розв'язання задач на основі навчання мережевої структури. Створена структура являє собою n-мірну мережу вузлів зв'язків, де кожен вузол відповідає за зв'язок відповідних йому значень параметрів. Перед навчанням кожний вузол ініціюється початковим значенням. Після цього відбувається певна кількість епох навчання, кожна з яких полягає у послідовній подачі навчальних прикладів (умов та запитань задачі), які встановлюють зв'язок значень вузлів мережі (параметрів задачі). Після цього відбувається уточнення значень вузлів мережі шляхом обрахування сум (n-1)-мірних шарів та визначення вже навчених вузлів. Наприкінці кожної епохи визначається сума мережі. Незмінність сум поточної та попередньої епох є критерієм завершення навчання.

Запропонований засіб апробовано на головоломках «Neighbors», «Meeting», «Ships», «Gardens» з ресурсу [4]. Кожна задача потребувала створення своєї n-мірної мережі, де n відповідно дорівнювало: 6, 5, 5, 4. Отримані результати довели коректність розв'язання логічних задач.

Перелік посилань:

1. Ekaterina K. Sherlock –A Neural Network Software for Automated Problem Solving / K. Ekaterina, Z. Qiming. // School of Computing. – 2012. – P. 152–164.

2. Wladston F. Solving the Zebra Puzzle with Boolean Algebra [Electronic resource] / Filho Wladston. – 2017. – Access mode: https://code.energy/solving-zebra-puzzle/.

3. Application of "Einstein's riddle" in solving construction machine allocation problems / D.Borna, G. Mario, Č. Marko, K. Uroš. // e-gfos. – 2016. – C. 12–22.

4. Einstein's riddle and grid puzzles [Electronic resource]. – 2012. – Access mode: http://brainden.com/einsteins-riddles.htm.

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»
Теплоенергетичний факультет

Київська державна академія водного транспорту
імені П.Конашевича-Сагайдачного

Інститут кібернетики ім.В.М.Глушкова НАН

V науково-практична дистанційна конференція
молодих вчених і фахівців
з розробки програмного забезпечення

«СУЧАСНІ АСПЕКТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

15 травня 2018

м. Київ

УДК 004.42:519.7
ББК 973.20-018.2я7

Рекомендовано Вченою радою Теплоенергетичного факультету Національного технічного університету України «Київський політехнічний інститут» (протокол № від року)

Сучасні аспекти розробки програмного забезпечення: Збірник наукових праць V науково-практичної дистанційної конференції молодих вчених і фахівців з розробки програмного забезпечення, 15 травня 2018 р. – Черкаси: видавець Чабаненко Ю.А., 2018. – 216 с.

У збірнику наукових праць V науково-практичної дистанційної конференції молодих вчених і фахівців з розробки програмного забезпечення «Сучасні аспекти розробки програмного забезпечення» наведено результати наукових досліджень та практичних розробок за наступними напрямами: інженерія програмного забезпечення, комп'ютерний еколого-економічний моніторинг, системи автоматизованого проектування.

Матеріали друкуються в авторській редакції.

ISBN

ЗМІСТ

УДК 004.032.26:004.832.3

Бараніченко О.М., магістрант

Шаповалова С.І., ктн, доцент

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Україна, Київ, вул. Політехнічна, 6. 0977580417, ljoschenka@gmail.com

Встановлення зв’язків між об’єктами логічної задачі

Існує широкий клас логічних задач, які необхідно розв’язувати у різноманітних сферах науки і техніки. Однак існуючі методи їх розв’язання або не задовольняють прийнятній швидкості, або не дають гарантовано точного результату. Використання нейронних мереж – найновіший підхід для розв’язання логічних задач. Через новизну цього підходу, існує мало результатів досліджень в цій області і досі не досягнуто рівня нейронних мереж, який задовольняє їх використанню в промислових системах. Тому створення концепції нейронних мереж для розв’язання логічних задач є актуальною задачею та має практичне застосування.

В роботі [1] наукова група корпорації Facebook надала класифікацію найбільш розповсюджених лінгвістичних задач, довела необхідність їх розв’язання та привела результати роботи системи, яка базується на використанні нейронних мереж з пам’яттю. При цьому було запропоновано набір прикладів [2], на яких проводилося навчання та тестування системи. В статті [3] було показано подібність логічних задач до задач типу XOR, приведено їх формалізацію та показано можливість їх розв’язання за допомогою елементів булевої алгебри. Однак недоліком такого підходу є більш складний, ніж у нейронних мереж, механізм модифікації постановки задачі, який полягає у додаванні нових логічних термів, їх властивостей та значень. Тому необхідне вдосконалення нейронних мереж для розв’язання логічних задач.

Логічну задачу можна представити у вигляді об’єктів, які мають набір властивостей. Нехай, X – множина всіх властивостей об’єкта логічної задачі:

$$X = \{X_1, X_2, X_3, \dots, X_n\}, \qquad (1)$$

де X_i (i=1..n) – і-та властивість об’єкта задачі.

В свою чергу, X_i має набір значень:

$$X_i \in V_i, \quad (2)$$

$$V_i = \{V_{i1}, V_{i2}, V_{i3}, \dots, V_{ik}\}, \quad (3)$$

де V_{ij} (j=1..k) – можливі значення властивості X_i.

Початкові умови задаються у вигляді відношень між значеннями властивостей.

Тоді розв’язок задачі для кожного об’єкту можна представити у вигляді

$$\{\langle X_1, \tilde{V}_1\rangle, \langle X_2, \tilde{V}_2\rangle, \dots, \langle X_n, \tilde{V}_n\rangle\}, \quad (4)$$

де $\tilde{V}_i$ – значення і-ї властивості, яке не суперечить всім умовам.

Запропонована для розв’язання логічних задач нейронна мережа, за класифікацією [4], є неповною багатошаровою мережею прямого розповсюдження.

Наведемо приклад скороченої задачі Ейнштейна: X_1 – колір будику, X_2 – напій, відповідно V_1 = {синій, білий, жовтий}, V_2 = {кава, чай, вода}. Мережа для розв’язання такої задачі з 2 характеристиками, кожна з яких має по 3 значення наведена на рисунку 1. Товщина ліній відповідає силам міжнейронних зв’язків. Для даної задачі це означає, що мешканець будинку білого кольору п’є каву.

Нейронна мережа складається з 3 шарів:

1) Шар вхідних нейронів.

Призначення – передача сигналу до відповідних нейронів наступного шару.

2) Шар логічних нейронів.

Призначення – здійснення процесу навчання. Кожен нейрон цього шару зв’язує два нейрони з першого шару, що фактично означає зв’язок між двома властивостями різних класів.

3) Шар вихідних обчислювальних нейронів.

Призначення – визначення нейронів-переможців, які визначають зв’язки між властивостями логічної задачі. Кожен вихід Y_{ij} цього шару означає індекс k, який позначає значення властивості V_{mk}, що має найбільшу силу зв’язку з відповідним йому значенням V_{ij} властивості.

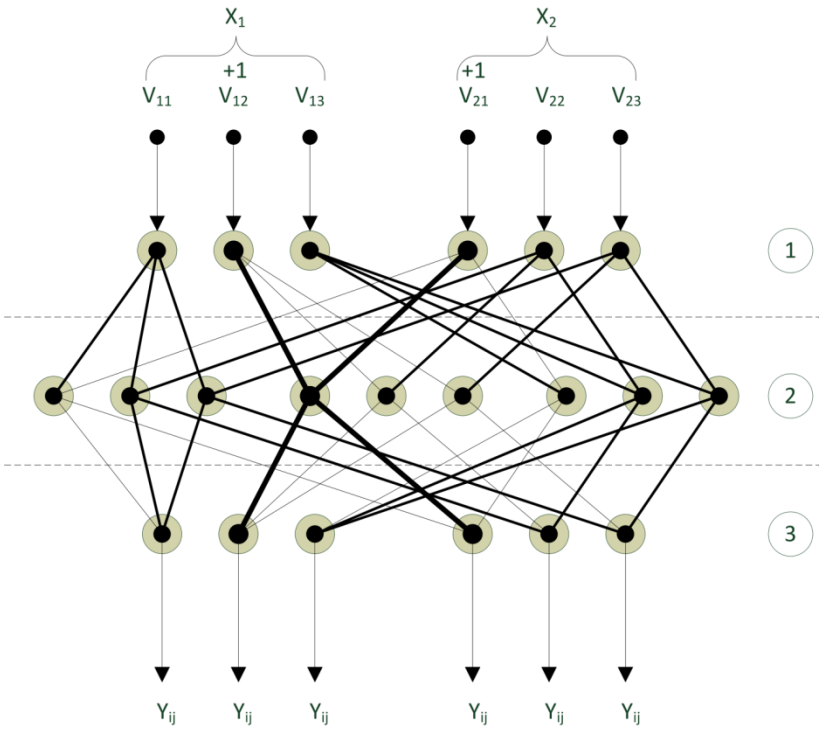

Рисунок 1. Архітектура нейронної мережі

Навчання нейронної мережі має такий алгоритм:

1) Після того, як на вхідний шар поступили сигнали з розрідженого вектора значень, відбувається активізація тих нейронів, на які поступили сигнали високого рівня.

2) Активізовані нейрони подають сигнал на шар логічних нейронів. Для активізації нейрону цього шару необхідна подача на нього двох сигналів з високими рівнями.

3) Після спрацювання логічних нейронів, активізовані нейрони збільшують вагові коефіцієнти відповідних зв’язків, і таким чином укріплюють їх. Ті нейрони, які не були активізовані, послаблюють свої зв’язки з нейронами вхідного шару, імітуючи при цьому процес відмирання зв’язку.

Для виведення результатів після навчання лінійна комбінація векторів вхідних сигналів та встановлених сил зв’язків подається на шар вихідних обчислювальних нейронів. Вони, в свою чергу, визначають найсильнішу групу зв’язків та, в залежності від постановки задачі, видають певний результат, наприклад індекс елементу з відповідної групи.

Висновки:

1. Проведено формалізацію логічних задач.

2. Запропоновано архітектуру нейронної мережі, яка здатна розв’язувати логічні задачі.

3. Показано алгоритми навчання та виведення результатів мережею.

Список літератури:

1. Weston J. Towards AI-complete questions answering : a set of prerequisite toy asks[Text] / J.Weston, A. Bordes, S. Chopra, A.M. Rush, B. van Merrienboer, A. Joulin, T. Mikolov// - Conference paper at ICLR, 2016.-P.14.

2. The bAbI project [Electronic resource] / [H. Miller, S. Chopra, M. Ranzato та ін.] // Facebook research. – 2015. – Mode of access: https://research.fb.com/downloads/babi/.

3. Filho W. Solving the Zebra Puzzle with Boolean Algebra [Електронний ресурс] / Wladston Filho. – 2017. – Режим доступу до ресурсу: https://code.energy/solving-zebra-puzzle/.

4. Хайкин С. Нейронные сети. Полный курс. / Саймон Хайкин. – Москва: Издательский дом "Вильямс", 2006. – 1104 с.