НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр КОВАЛЬ

«__» _____2021 р.

Дипломна робота

На здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерне геометричне моделювання процесів та систем»

спеціальності 122 «Комп'ютерні науки»

на тему: «Моделювання причинно-наслідкової структури, враховуючи невимірний фактор впливу»

Виконав
студент IV курсу, групи ТР-71
Лісовенко Андрій Анатолійович _____

Керівник:
Асистент
Беспала Ольга Миколаївна _____

Консультант: _____

Рецензент: _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності: 122 «Комп'ютерні науки»

освітня програма: «Комп'ютерне геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр КОВАЛЬ
«___» _____ 2021 р.

ЗАВДАННЯ

на дипломну роботу студенту

Лісовенку Андрію Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання причинно-наслідкових структур, враховуючи невимірний фактор впливу

керівник роботи Беспала Ольга Миколаївна, асистент
(прізвище, ім'я, по батькові, наукова ступінь, вчене звання)

затверджено наказом вищого навчального закладу від "29" березня 2021р. №_

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи _мова програмування C#, фреймворк .NET, середовище розробки Visual Studio 2019

4. Зміст розрахунково пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі алгоритми пошуку причинно-наслідкових структур; розробити метод знаходження оцінки впливу між параметрами; розробити програмний продукт для візуалізації причинно-наслідкових структур на основі статистичних даних з урахуванням невимірного фактору.

5. Перелік ілюстративного матеріалу: Актуальність, Постановка задачі, Встановлення причинно-наслідкових структур, Алгоритм причинно-наслідкових структур, Приклади роботи на основі прогностичних моделей, Інтерфейс програмного продукту, Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "29" ____березня____ 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Терміни виконання етапів роботи	Примітки
1.	Затвердження теми роботи	30.03.2021	
2.	Вивчення та аналіз задачі	31.03.2021-09.042021	
3.	Розробка архітектури та загальної структури програмного продукту	12.04.2021-16.05.2021	
4.	Програмна реалізація програмного продукту	18.05.2021	
5.	Оформлення пояснювальної записки	12.03.2021-04.06.2021	
6.	Захист програмного продукту	12.05.2021	
7.	Передзахист	26.05.2021	
8.	Захист	17.06.2021	

Студент _____ _____Лісовенко А.А._____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____Беспала О.М._____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дана дипломна робота присвячена розробці програмного продукту, метою якого є моделювання причинно-наслідкових структур за допомогою статистичних даних математичної моделі, з урахуванням невимірного фактору впливу.

Програмний продукт розроблявся в середовищі Visual Studio 2019 мовою C# використовуючи технологію .NET.

Загальний об’єм роботи 50 аркушів, 29 рисунків, 12 таблиці, 2 блок-схкми, 3 додатки.

Ключові слова**:** причинно-наслідкова структура, моделювання причинно-наслідкових зв’язків, невимірний фактор впливу, граф причинності.

ABSTRACT

The aim of the diploma thesis is the development of a software product which aims to model causal structures using static data of a mathematical model, taking into account the immeasurable influence factor.

The software product was developed in Visual Studio 2019 in C # using .NET technology.

Total capacity: 50 pages, 29 figures, 12 tables, 2 flowchart, 3 appendices.

Keywords: causal structure, modeling of causal relationships, immeasurable factor of influence, causality graph.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПНС – причинно-наслідкова структура

ПП – програмний продукт

ШІ – штучний інтелект

PAG – Partially oriented graph (частково орієнтований граф)

DAG – Directed acyclic graph (спрямований ациклічний граф)

FCI – Fast Causal Inference (алгоритм швидкого причинно-наслідкового висновку)

GES – Greedy Equivalence Search (жадібний пошук еквівалентності)

LiNGAM – Linear non-Gaussian Acyclic Model (лінійна негаусова ациклічна модель)

ВСТУП

Встановлення причинно-наслідкових структур є одним з базових відносин, за допомогою яких будується уявлення знань в багатьох предметних областях, зокрема в слабоформалізованих. Інформація про причинно-наслідкові відносини дозволяє не тільки описувати і поповнювати структуру знань про об'єкти і явища в даній галузі, а й проводити планування і прогнозування, що особливо має значення в тих напрямках, для яких не створено загальних формальних моделей.

Більшість існуючих алгоритмів дають пояснення про те, як змінні причинно пов’язані між собою. Проте, щоб оцінити вплив однієї змінної на іншу необхідно виключити всі можливі впливи сторонніх факторів і на практиці це не завжди можливо. Саме тому розробка програмного продукту, алгоритм якого встановлює причинно-наслідкову структуру, враховуючи невимірний фактор впливу є актуальною задачею.

Метою даної роботи є розробка та створення програмного продукту, алгоритм якого встановлює причинно-наслідкові зв’язки між вхідними параметрами моделі, які задаються у вигляді набору даних з прихованими змінними.

1. ХАРАКТЕРИСТИКА ЗАДАЧ ВСТАНОВЛЕННЯ ПРИЧИННО-НАСЛІДКОВИХ СТРУКТУР, ВРАХОВУЮЧИ НЕВИМІРНИЙ ФАКТОР ВПЛИВУ

1.1. Причинно наслідкові структури, їх призначення

Причинно-наслідковий зв’язок є одним з небагатьох основних зв’язків, завдяки яким будується уявлення знань у будь-якій предметній області. Інформація про причинно-наслідкові відносини дозволяє не тільки описати і доповнити структуру знань про предмети та явища в даній предметній області, а також проводити планування та прогнозування, що особливо цінне в тих сферах, для яких не створено загальних формальних моделей, такі як охорона здоров'я, освіта, управління, фінанси, сільське господарство, психологія та медицина тощо [1].

Виявлення причинних наслідків є невід’ємною частиною дослідження, яке охоплює широкий спектр питань, таких як розуміння поведінки в Інтернет-системах, вплив соціальної політики або фактори ризику. Оскільки комп’ютери дедалі більше впливають на всі сфери життя, причинно-наслідковий зв'язок також має вирішальне значення для проектування та оцінки всіх комп'ютерних систем та додатків, які ми створюємо. Наприклад, як алгоритмічні рекомендації впливають на наші рішення про закупівлю? Як вони впливають на результати навчання студентів чи медичну ефективність? Це складні питання, що вимагають розгляду протилежного: що у світі може статися з іншою системою, політикою чи втручанням?

Задачі, які має виконувати програмний продукт:

- Приймати на вхід набір даних у вигляді .csv файлу
- Враховувати припущення щодо типів зв’язків між параметрами

- Встановлювати причинно-наслідкові структури між параметрами, враховуючи невимірний фактор впливу
- Візуалізувати у вигляді графа причинності встановлені причинно-наслідкові структури

1.2. Врахування припущень типу зв'язків між параметрами

Для пояснення причинно-наслідкових структур між змінними використовується аналіз шляху. Функцією аналізу шляху є посередництво, яке передбачає, що змінна може впливати на результат безпосередньо та опосередковано через іншу змінну. Тобто в ПП повинно бути припущення, що моделювання причинно-наслідкових структур відбувається без посередників – то такий зв'язок буде прямим (рис. 1.1.) [2],

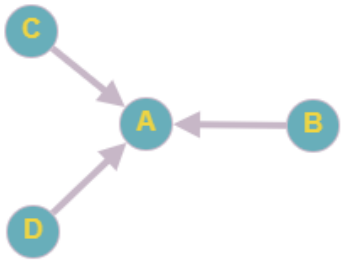


Рисунок 1.1. Модель з прямими причинно-наслідковими зв'язками

а якщо з посередником (рис. 1.2.К) то, відповіно, такий зв'язок буде непрямим (рис. 1.2.).

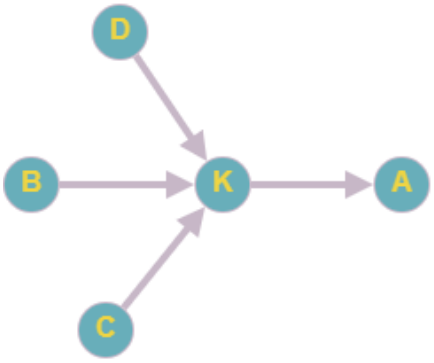


Рисунок 1.2. Модель з непрямими причинно-наслідковими зв'язками

1.3. Невимірний фактор

Невимірний фактор – це змінні, які не виявляються безпосередньо, а скоріше виявляються з інших змінних, які є явними. Математичні моделі що мають на меті пояснити спостережувані змінні з точки зору прихованих змінних, називаються моделями прихованих змінних. Моделі прихованих змінних використовуються в багатьох дисциплінах, включаючи психологію, економіку, медицину, фізику, машинне навчання, хімію та інших [2].

Для встановлення причинно-наслідкових структур між параметрами необхідно виключити всі можливі впливи сторонніх факторів, але на практиці це не завжди можливо. Так на практиці, зустрічається постійний систематичний вплив факторів на досліджуваний параметр, проте існує ймовірність випадкових факторів впливу, які неможливо виміряти, оскільки вони можуть мати тимчасовий та непередбачуваний характер.

Тому постає необхідність в розробці програмного продукту, алгоритм якого встановлює причинно-наслідкову структуру, враховуючи невимірний фактор впливу.

2. ІНСТРУМЕНТАРІЙ ВСТАНОВЛЕННЯ ПРИЧИННО-НАСЛІДКОВИХ СТРУКТУР

2.1. Алгоритми причинно-наслідкових структур

Для побудови причинно-наслідкової структури виникає необхідність у застосуванні різних алгоритмів для розв'язання поставлених задач [2].

Існують такі алгоритми:

PC-алгоритм – цей алгоритм виконує пошук за шаблоном, що передбачає, що основна причинно-наслідкова структура вхідних даних є ациклічною і те що змінні не спричинені прихованим параметром (невимірною змінною). Цей алгоритм є ефективним для будь-якого типу даних, для яких відомі факти умовної незалежності.

Робота алгоритму має наступну послідовність:

- Формує повний ненаправлений граф як на рисунку 2.1.B.
- Усуває ребра між змінними, які безумовно незалежні, в даному випадку це X і Y рисунок 2.1.C.
- Для кожної пари змінних (A, B), що мають ребро між собою, і для кожної змінної C з ребром, підключеним до будь-якої з них, усуває ребро між A і B, якщо $A \perp\!\!\!\perp B \mid C$, як на рисунку 2.1.D.
- Для кожної пари змінних A, B, що має ребро між собою, і для кожної пари змінних { C, D } з ребрами, приєднаними до A або обома підключеними до B , усуває ребро між A і B, якщо $A \perp\!\!\!\perp B \mid \{ C, D \}$.

Продовжує перевіряти незалежності, обумовлені підмножинами змінних зростаючого розміру n, поки більше не буде сусідніх пар (A, B), так що існує підмножина змінних розміром n, така що всі змінні в підмножині сусідні з A або все поруч з B . У розглянутому прикладі Z та W не є незалежними

від X, від Y або від X та Y , тому подальших статистичних рішень приймати не потрібно. Подібним чином для X і Z , а також для Y і Z.

- Для кожної трійки змінних (A, B, C) таких, що A і B суміжні, B і C суміжні, а A і C не суміжні, орієнтуйте ребра A - B - C як A → B ← C , якщо B не знаходився в наборі кондиціонування, на якому A і C ставали незалежними, і ребро між ними відповідно було усунене. Ми називаємо таку трійку змінних v-структурою .

  У цьому прикладі Z не був обумовлений усуненням ребра X - Y , тому орієнтуйте X - Z - Y як X → Z ← Y , а результат наведено на рисунку 2.1.E

- Для кожної трійки змінних , таких , що → B - C , і та C не є суміжними, орієнтувати кромку B - C , як B → C . Це називається поширенням орієнтації .

  На рисунку 2.1.F , Y → Z - W орієнтована , як Y → Z → W . У цьому прикладі справжня структура відновлюється однозначно.

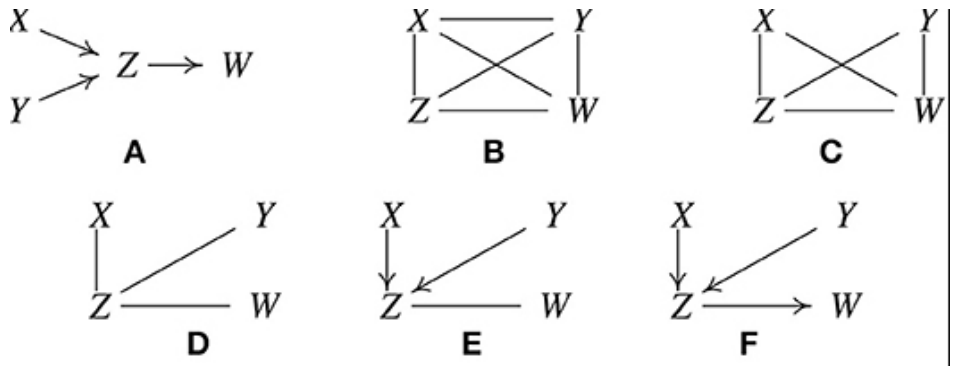


Рисунок 2.1. Ілюстрація роботи PC-алгоритму

Алгоритм швидкого причинно-наслідкового висновку (FCI) – цей алгоритм є модифікацією PC-алгоритму, який приймає як вхідні дані, вибіркові дані та необов'язкові фонові знання. Його результати є асимптотично правильними навіть при наявності невідомих факторів впливу. Рисунок 2.2.A ілюструє, де U – невідома змінна, справжній причинно-наслідковий граф. Але він має

значні недоліки. FCI-алгоритм обмежено кількома тисячами змінних, а на реальних розмірах вибірки він є неточним.

Як і у першому стані процедури PC, FCI вимагає суджень щодо статистичної незалежності, щоб обрізати ненаправлений графік, наведено на рисунку 2.2.A

Позначка «o» означає, що це може бути головка стріли або хвіст стрілки. Причина позначення "o" стане очевидною. FCI орієнтує краї за процедурою, подібною до ПК, але без припущення, що кожен край спрямований в ту чи іншу сторону. Край X ⟜ Z був усунутий без кондиціонування на Y, оскільки X і Z беззастережно незалежні, а X ⟜ Y ⟜ Z потрійний тому орієнтований як «прискорювач», X ⟜> Y <⟜ Z . Таким же чином виявляється , що Y ⟜ Z ⟜ W є колайдером, Y ⟜> Z <⟜ W , що дає рисунок 2.2.C.

Двонаправлене ребро між Y і Z указует на те, що існує принаймні один невиміряна змінна з Y і Z . Решта символи "o" на X та W вказують на те, що алгоритм не може визначити, чи є з'єднання X, Y напрямленим фронтом від X до Y , чи не виміряним конфудером, або тим і іншим; те саме для Z та W.

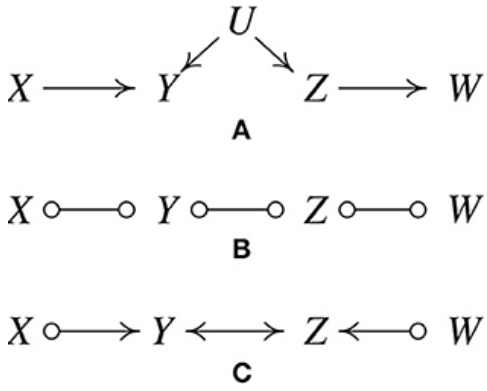


Рисунок 2.2. Ілюстрація алгоритму FCI

Жадібний пошук еквівалентності (GES) – замість того, щоб починати з повного ненаправленого графіка, як у PC та FCI, Жадібний еквівалентний пошук (GES) починається з порожнього графіка та додає потрібні на даний

момент ребра, а потім усуває непотрібні ребра у шаблоні. На кожному кроці в алгоритмі приймається рішення щодо того, чи додасть спрямований край до графіка, збільшить відповідність, виміряну деякою квазі-байєсовою оцінкою, такою як BIC, додається грань, яка найбільше покращує придатність.

Лінійна негаусова ациклічна модель (LiNGAM) – цей алгоритм базується на використанні негаусованості даних. Алгоритм ефективно видає оцінки моделей структурних рівнянь або лінійних баєсівських мереж.

Різницю між цими алгоритмами показано в таблиці 2.1.

Особливість алгоритму	PC	FCI	GES	LINGAM
Наявність припущень причинності	+	+	+	-
Наявність припущень щодо розподілу даних	-	-	+	+
Коректне відображення факторів впливу	-	+	-	-
Вихідні дані	Клас еквівалентності маркова	PAG	Клас еквівалентності маркова	DAG а також причинно-наслідкова модель

Таблиця 2.1. Порівняльна характеристика алгоритмів

У теорії ймовірностей та статистиці термін Марковська властивість відноситься до властивості відсутності пам'яті в стохастичному процесі. Його назвали на честь російського математика Андрія Маркова.

Подія має властивість Маркова, якщо умовний розподіл ймовірностей майбутніх станів цього процесу (як через минулий, так і поточний стан) залежить лише від поточного стану, а не від послідовності подій, які йому передували. Процес з цією властивістю називається Марковським процесом [3].

Partially oriented graph (PAG) – це частково орієнтований граф, де використовується три види ребер:

однонапрямлені ребра, стрілка на одному кінці ребра; двонаправлені ребра, на обох кінцях по стрілці; ненаправлені ребра, у яких немає стрілок.

Directed acyclic graph (DAG) – це спрямований ациклічний граф, де кожен вузол являє собою змінну, а кожна стрілка - пряму причину [4].

Як результат, в основу програмного продукту ліг PC-алгоритм оскільки в поставленій задачі є обмеження на моделі, а саме належність до класу Маркова, а PC – алгоритм якраз з ними працює [5].

2.2. Методи оцінки зв’язку між параметрами моделі

Більшість алгоритмів праналізованих вище встановлюють ПНС на основі оцінок. В цьому розділі проведено аналіз наступних методів такі, як критерій Фішера, Кореляція, інформаційний критерій Акаіке, інформаційний критерій Байєса та тест ксі-квадрат [6].

Інформаційний критерій Акаіке та інформаційний критерій Байєса є двома відносними показниками з точки зору вибору моделі, а не тесту нульової гіпотези. AIC пропонує відносну оцінку втраченої інформації, коли дана модель використовується для формування даних. BIC - це оцінка того, наскільки ощадливою є модель серед кількох кандидатів. AIC та BIC не є корисними для перевірки нульової гіпотези, але корисні для вибору моделі з найменшим переоснащенням.

Критерій Фішера – це будь-який статистичний критерій, тестова статистика якого при виконанні нульової гіпотези має розподіл Фішера.

Тест ксі-квадрат перевіряє гіпотезу про те, що існує невідповідність матриці коваріації, що має на увазі під моделлю, та вихідної матриці коваріації. Але цей метод дуже чутливий до обсягу вибірки і не порівнянний між різними оцінками зв'язку між параметрами.

Кореляція – це метод обробки статистичних даних, за допомогою якого вимірюється тіснота між двома або більше змінними. Кореляція відображає лише лінійну залежність величин, але не відображає їхньої функціональної зв'язності.

X1	E3	Кореляція	Критерій Фішера	Ксі квадрат
1,958	567,21254	1	0,5804224174	0
9,81	576,0853			
1,872	567,11536			
4,498	570,08274			
4,236	569,78668			
0,833	565,94129			
4,486	570,06918			
0,606	565,68478			
0,912	566,03056			
2,603	567,94139			
23,03	591,0239			
1,568	566,77184			
240,09	836,3017			
45,43	616,3359			
18,461	585,86093			
1,328	566,50064			
1,103	566,24639			
64,17	637,5121			
1,995	567,25435			
199,4	790,322			
230,3	825,239			
0,11	565,1243			

X1	E3	Невідома змінна	Кореляція	Критерій Фішера	Ксі квадрат
1,958	598,21254	31	0,8469651193	0,05366466747	0
9,81	483,0853	-93			
1,872	589,11536	22			
4,498	645,08274	75			
4,236	518,78668	-51			
0,833	651,94129	86			
4,486	479,06918	-91			
0,606	584,68478	19			
0,912	513,03056	-53			
2,603	632,94139	65			
23,03	631,0239	40			
1,568	621,77184	55			
240,09	924,3017	88			
45,43	640,3359	24			
18,461	667,86093	82			
1,328	654,50064	88			
1,103	540,24639	-26			
64,17	592,5121	-45			
1,995	474,25435	-93			
199,4	857,322	67			
230,3	806,239	-19			
0,11	501,1243	-64			

Таблиця 2.2. Порівняльна таблиця оцінка методів з урахуванням невідомого фактору впливу та без нього

Оскільки кореляційний аналіз показав один з кращих результатів оцінки між залежністю двох параметрів, як з урахуванням невимірного фактору так і без, тому доцільне використання саме цієї оцінки.

2.3. Програмні засоби для встановлення причинно-наслідкових структур

Tetrad – це настільний Java-додаток, який за необхідності може підключатися до зовнішніх супер обчислювальних ресурсів, що створює, моделює дані, оцінює, тестує, передбачає та шукає причинно-наслідкові та статистичні моделі. Метою програми є створення витончених методів, що вимагає дуже мало статистичної вишуканості користувача та відсутність знань програмування.

R-Causal – модуль R, який огортає алгоритми для проведення причинно-наслідкового виявлення великих даних.

Py Causal - модуль python, який огортає алгоритми для проведення причинно-наслідкового виявлення великих даних.

Causal Command – бібліотека Java та реалізація алгоритму командного рядка для виконання причинно-наслідкових структур в великих даних. Це програмне забезпечення можна використовувати, якщо користувача цікавить включення аналізу за допомогою сценарію оболонки або в програму на основі Java [7, 8].

Для розуміння наведу приклад роботи застосунку Tetrad, щоб показати в загальному які плюси цих програм та мінуси.

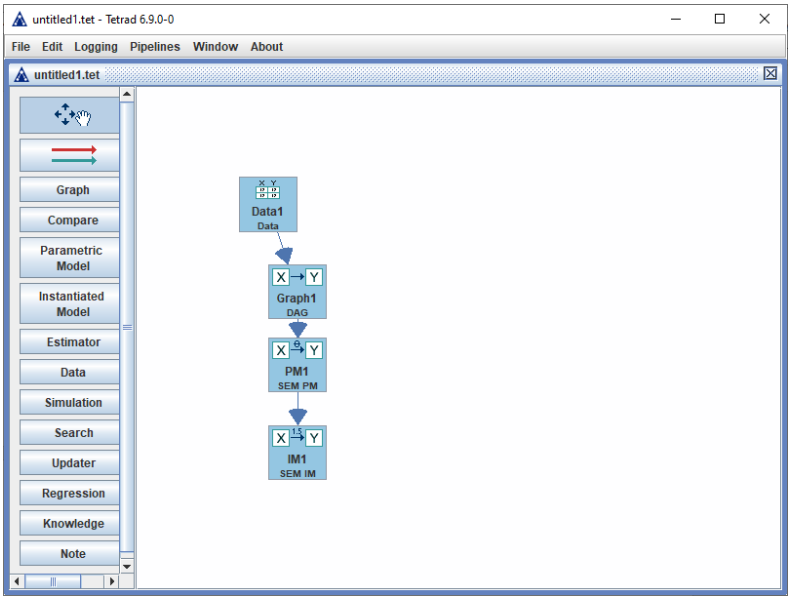
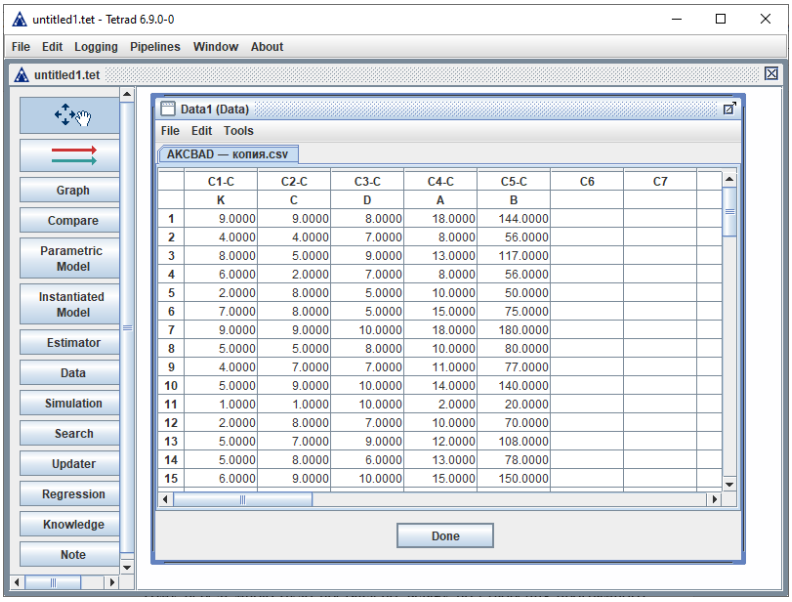


Рисунок 2.3. Інтерфейс програми Tetrad

1. Створюємо вкладку Data для завантаження даних
   1) Відкриваємо вкладку File
   2) Вказуємо шлях до файлу з даними
   3) Завантажуємо файл і натискаємо на кнопку Done



	C1-C	C2-C	C3-C	C4-C	C5-C	C6	C7
	K	C	D	A	B		
1	9.0000	9.0000	8.0000	18.0000	144.0000		
2	4.0000	4.0000	7.0000	8.0000	56.0000		
3	8.0000	5.0000	9.0000	13.0000	117.0000		
4	6.0000	2.0000	7.0000	8.0000	56.0000		
5	2.0000	8.0000	5.0000	10.0000	50.0000		
6	7.0000	8.0000	5.0000	15.0000	75.0000		
7	9.0000	9.0000	10.0000	18.0000	180.0000		
8	5.0000	5.0000	8.0000	10.0000	80.0000		
9	4.0000	7.0000	7.0000	11.0000	77.0000		
10	5.0000	9.0000	10.0000	14.0000	140.0000		
11	1.0000	1.0000	10.0000	2.0000	20.0000		
12	2.0000	8.0000	7.0000	10.0000	70.0000		
13	5.0000	7.0000	9.0000	12.0000	108.0000		
14	5.0000	8.0000	6.0000	13.0000	78.0000		
15	6.0000	9.0000	10.0000	15.0000	150.0000		

Рисунок 2.4. Вкладка Data

2. Створюємо вкладку Graph для візуалізації графа

1) З вкладки Data на основі завантажених даних програма візуалізувала вершини
2) Потрібно нажати на кнопку по встановленню зв’язків між вершинами
3) З’єднати вершини в потрібному напрямку (тобто програма сама не вміє малювати стрілки, що є мінусом даного ПП)

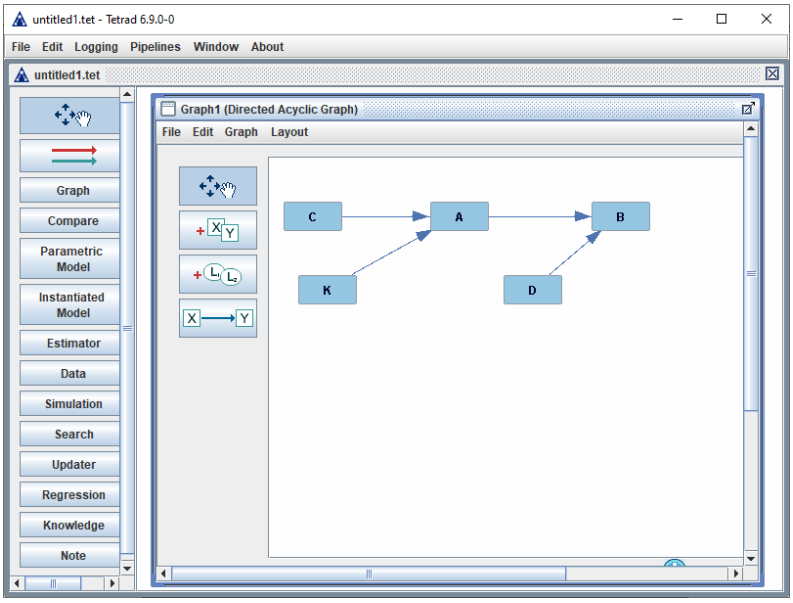


Рисунок 2.5. Вкладка Graph

3. Створюємо Parametric Model для регулювання параметрів

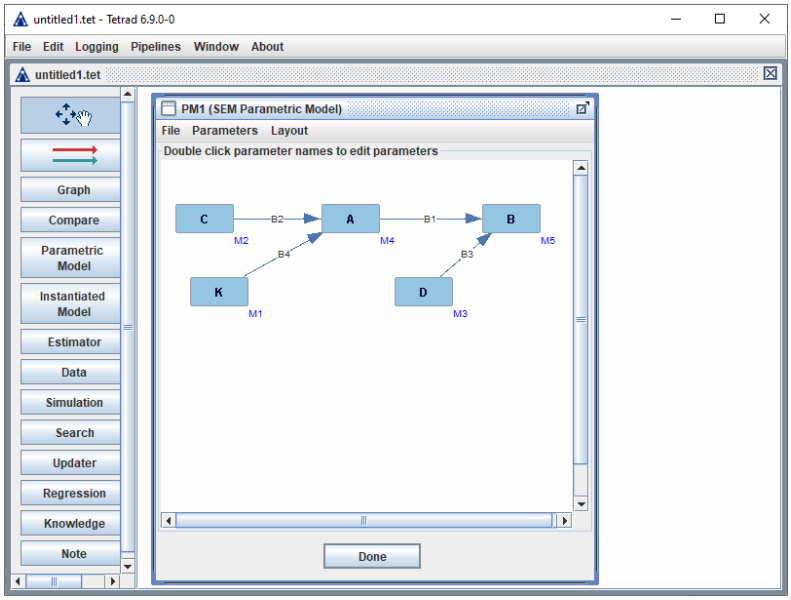


Рисунок 2.6. Вкладка Parametric Model

4. Створюємо Instantiated Model для перегляду всіх параметрів моделі

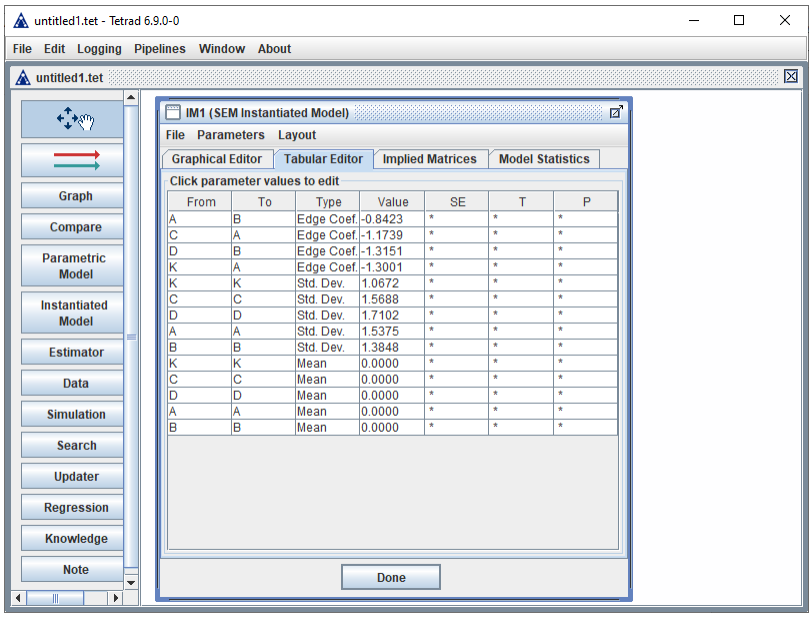


Рисунок 2.7. Вкладка Instantiated Model

Але ці програмні продукти в тій чи іншій мірі складні для розуміння, через що виникає складність з використанням користувачами даного програмного продукту. А також в цих програмних продуктах немає можливості врахування невимірних факторів, що є великим недоліком даних ПП.

Тому було поставлено задачу створити програмний продукт який:

- Знаходитиме причинно-наслідкові зв’язки між статистичними даними, враховуючи невимірний фактор впливу.
- Візуалізуватиме залежності між змінними за допомогою напрямленого графа.
- Зрозумілий користувачу для використання.

2.4. Висновок до розділу

У даному розділі показано опис існуючих програмних продуктів, що знаходять причинно-наслідкові структури. Показані їх плюси та мінуси в їх роботі.

Проаналізовано існуючі методи оцінки зв’язку між параметрами моделі.

Проаналізовано алгоритми побудови причинно-наслідкових структур для отримання розгорнутої картини залежностей всіх змінних і візуалізації цих зв’язків.

3. ОПИС АЛГОРИТМУ ТА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1. Опис алгоритму

При запуску програмного продукту користувачу потрібно вказати шлях до файлу з статистичними даними, які переписуються в масив, потім на основі цих даних робиться оцінка зв’язку між параметрами.

Оскільки використаний PC-алгоритм не повністю задовільнив умовам поставленої задачі, було прийнято рішення модифікувати даний алгоритм для врахування обмежень. Оскільки візуалізація відбувається за допомогою направленого графу, необхідно видалити петлі, або іншими словами виключити вплив параметра самого на себе і тим самим необхідно обнулити головну діагональ. Друге обмеження на вхідні дані, що всі параметри записані в хронологічному порядку, отже потрібно обнулити все що нижче головної діагоналі.

Блок-схема вищеописаного алгоритму наведена на рисунку 3.1.

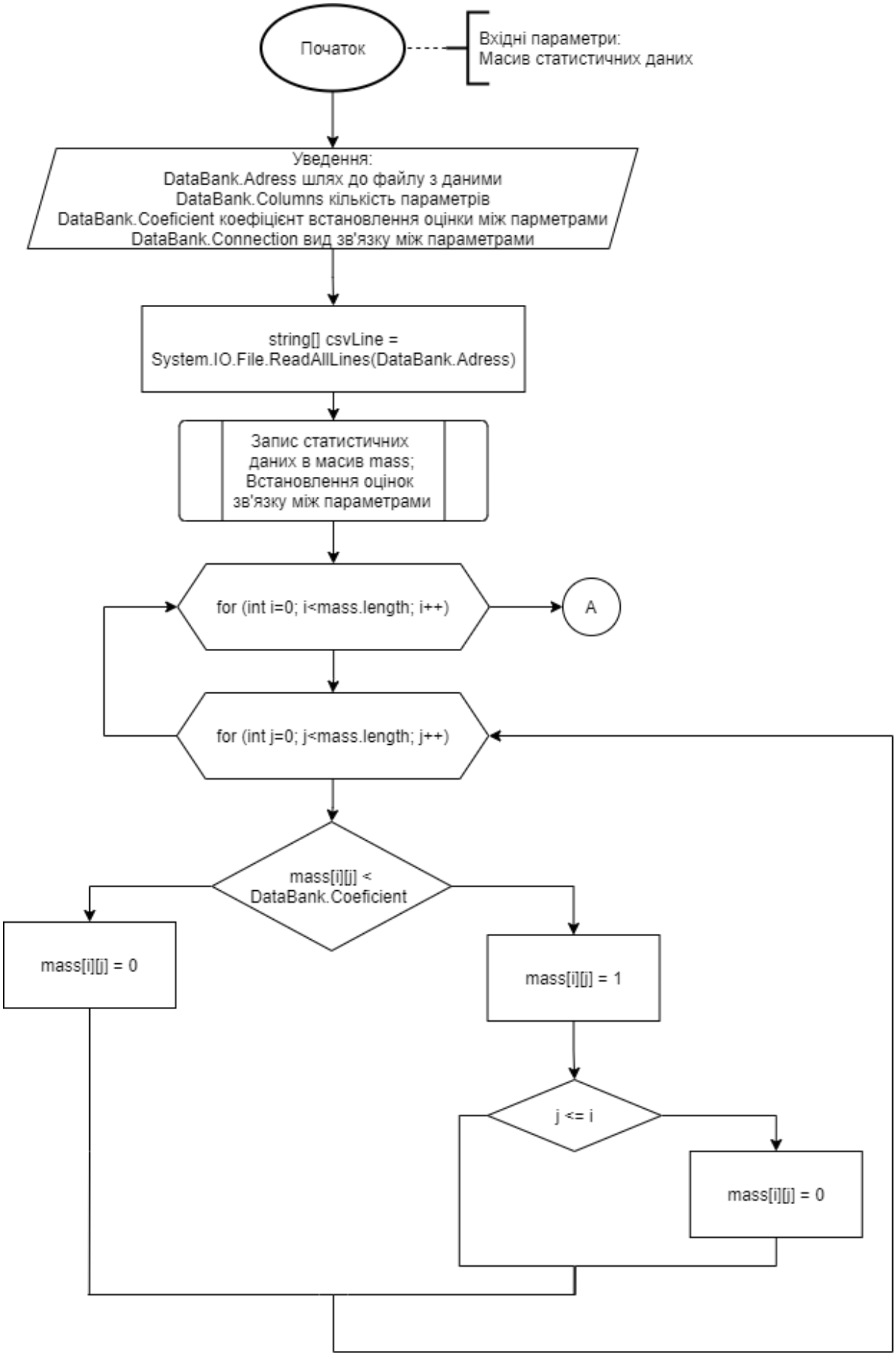


Рисунок 3.1. Блок-схема роботи алгоритму

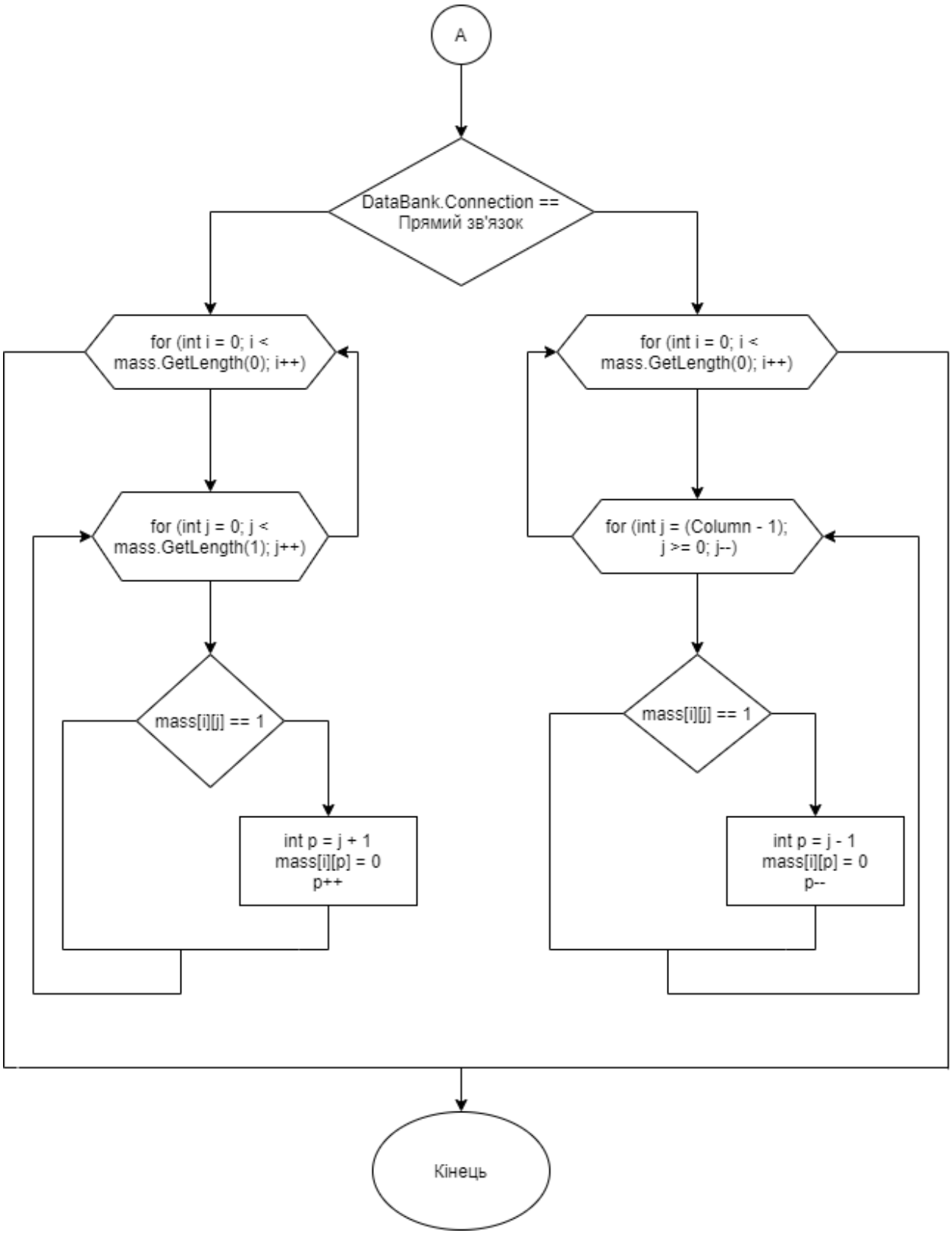


Рисунок 3.2. Блок-схема аналізу зв’язку між параметрами

Так як даний ПП повинен аналізувати зв’язок між параметрами було запрограмовано алгоритм встановлення прямих та непрямих зв’язків (рисунок 3.2.) [6].

Розглянемо алгоритм побудови зв’язків:

Вхідні дані: параметри системи та їх статичні дані, тобто матриця A: NxM, N – параметри системи; M – стан параметру системи.

1. Якщо розглядається модель з прямими причинно-наслідковими зв’язками.

Твердження 1. Якщо параметр Nn є умовно залежним з параметром Nn-i, то параметр Nn-i є умовно залежним лише з параметром Nn.

Алгоритм передбачає: припущення щодо причинності між параметрами, а саме наявність прямих причинних зв’язків; хронологічну послідовність параметрів Ni=1,n в матриці A.

Крок 1. Записати в матрицю A1: NxN оцінку умовної незалежності між параметрами системи.

Крок 2. Відповідно до твердження 1 послідовно встановити залежність між параметрами Nn…N1.

Крок 3. Результатом будуть елементи матриці, які вище головної діагоналі.

2. Якщо розглядається модель з непрямими причинно-наслідковими зв’язками.

Твердження 2. Якщо існують такі параметри Nn-i | Nn, Nn-k | Nn, Nn-k | Nn-i , то вважати, що Nn-k ⫫ Nn-i.

Алгоритм передбачає: припущення щодо належності причинно-наслідкових структур до класу еквівалентності Маркова; хронологічну послідовність параметрів Ni в матриці A.

Крок 1. В матриці A1: NxN міститься оцінка умовної незалежності між параметрами системи.

Крок 2. Відповідно до твердження 2 послідовно встановити залежність між параметрами Nn…N1.

Крок 3. Результатом будуть елементи матриці, які вище головної діагоналі.

3.2. Вхідні дані

Для коректної роботи програмного продукту потрібно правильно заповнений файл, з розширенням .csv, де в першому стовпчику потрібно написати найменування параметрів, а потім, відповідно, записати самі параметри, кожен параметр в окремий стовпчик, параметри потрібно записувати послідовно, параметрів не може бути більше чим їх найменувань, або навпаки. Нижче на таблиці 3.1. показано приклад заповненого .csv файла.

K	C	D	A	B
4	8	6	12	72
9	9	8	18	144
4	4	7	8	56
8	5	9	13	117
6	2	7	8	56
2	8	5	10	50
7	8	5	15	75
9	9	10	18	180
5	5	8	10	80
4	7	7	11	77
5	9	10	14	140
1	1	10	2	20
2	8	7	10	70
5	7	9	12	108
5	8	6	13	78
6	9	10	15	150
4	10	5	14	70
9	9	10	18	180

Таблиця 3.1. Приклад використання вхідних даних

3.3. Вихідні дані

Після успішно виконаної роботи по побудові ПНС, даний ПП візуалізує ці структури у вигляді направленого графа, що зображено на рисунку 3.4.

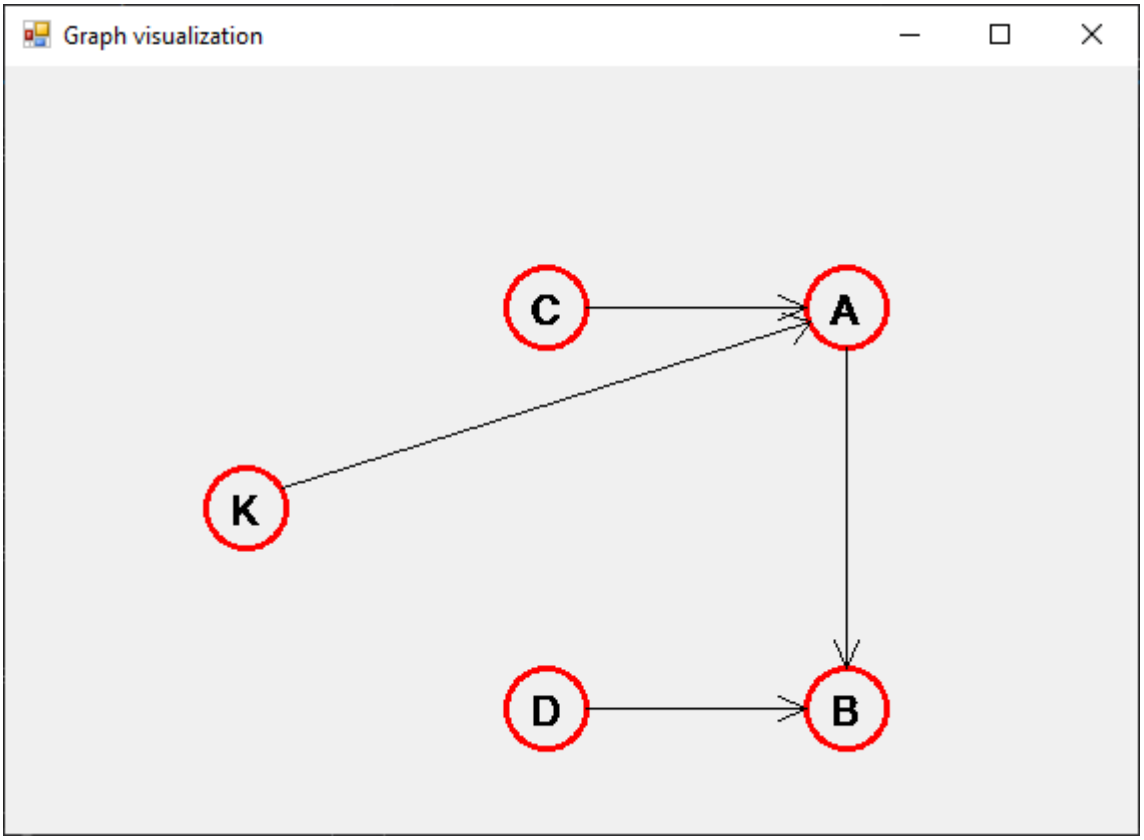


Рисунок 3.4. Приклад роботи програмного продукту

3.4. Висновок до розділу

У даному розділі було описано спосіб подання вхідних даних, їх обмеження, було наведено приклад.

Описано роботу PC-алгоритму, алгоритм встановлення зв’язків між параметрами на прикладі блок-схем.

Показано вихідні дані програмного продукту у виді направленого графа.

4. ПЕРЕВІРКА РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

4.1. Тестування моделей з прямими зв’язками

1. Прогностична модель захворюваності на гострий інфаркт міокарда [9]

$$ГІМ = 137,6 + 0,74(X_1) + 0,68(X_3) - 1,36(X_4) + 1,04(X_5),$$

де X_1 – об’єм скидання недостатньо очищених стічних вод у поверхневі водні об’єкти;

X_3 – викиди в атмосферне повітря хімічних речовин пересувними установками;

X_4 – викиди в атмосферне повітря хімічних речовин стаціонарними установками у поточному році;

X_5 – викиди в атмосферне повітря хімічних речовин стаціонарними установками на два роки раніше.

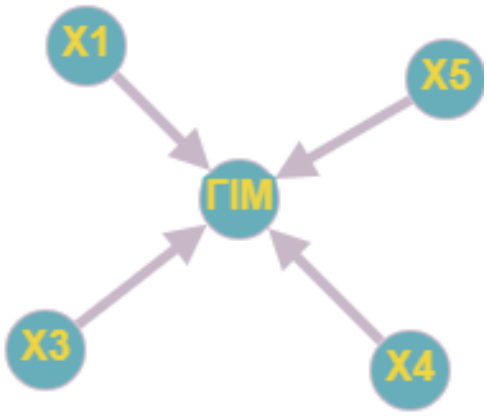


Рисунок 4.1. Граф побудований на основі математичної моделі

Нижче наведено таблицю 4.1. з статистичними даними по захворюванню на гострий інфаркт міокарда. А також приклад роботи ПП на основі цих статистичних даних.

X1	X3	X4	X5	R
1,869	3,358	84,4	48,188	76,59802
4,9	2,6	106,5	45	44,954
1,28	0,0525	2,4	3,3	138,7509
4,498	0,4578	9,9	9,6	137,7598
3,857	2,3	3,7	3,2	140,3142
0,82	3,2762	99,7	155,8	166,8746
3,317	1,9	51,839	48,318	121,0963
0,606	0,96	20,331	21,735	133,6555
0,372	0,5002	17,8	9,6	123,9914
1,991	2,078	9,4	10,6	138,7264
22,96	0,9778	21,68	20,33	146,9137
1,568	8,39	50,959	55,892	133,289
24,9	1,45	216,19	213,25	399,4264
43,9	6,922	109,1	88,87	118,8418
17,01	1,0312	37,3	75,1	178,2646
1,328	2,255	12,8	12,2	135,3961
1,075	4,548	205	198,3	68,92014
63,78	0,393	173,401	180,9	137,3751
1,897	0,8235	12,7	10,3	133,0038
199,1	0,626	773,5	784,8	49,59168
131,7	2,587	576,925	657,3	135,7912
1,1	0,2	5,3	5,1	135,906

Таблиця 4.1. Статистичні дані математичної моделі

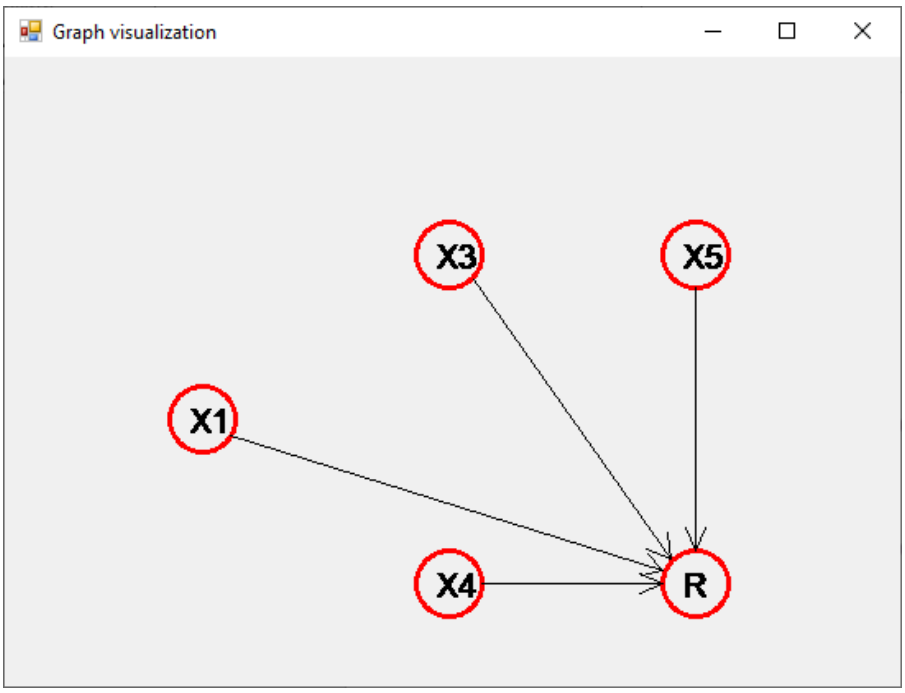


Рисунок 4.2. Направлений граф побудований на основі статистичних даних

2. Прогностична модель захворюваності на мозковий інсульт [9]

$MI = 592 + 1{,}38(X_1) - 0{,}79(X_2)$,

де X_1 – скидання забруднених стічних вод у поверхневі водні поточного року;

X_2 – викиди в атмосферне повітря хімічних речовин стаціонарними установками у поточному році.

Рисунок 4.3. Граф побудований на основі математичної моделі

Нижче наведено таблицю 4.2. з статистичними даними по захворюванню на мозковий інсульт. А також приклад роботи ПП на основі цих статистичних даних.

X1	X2	R
1,958	84,4	528,026
9,81	106,5	521,4028
1,872	2,4	592,6874
4,498	9,9	590,3862
4,236	3,7	594,9227
0,833	99,7	514,3865
4,486	51,839	557,2379
0,606	20,331	576,7748
0,912	17,8	579,1966
2,603	9,4	588,1661
23,03	21,68	606,6542
1,568	50,959	553,9062
24,09	216,9	784,577
45,43	109,1	568,5044
18,461	37,3	588,0092
1,328	12,8	583,7206
1,103	205	431,5721
64,17	173,401	543,5678
1,995	12,7	584,7201
199,4	773,5	256,107
230,3	576,925	454,0433
1,11	5,3	587,9648

Таблиця 4.2. Статистичні дані математичної моделі

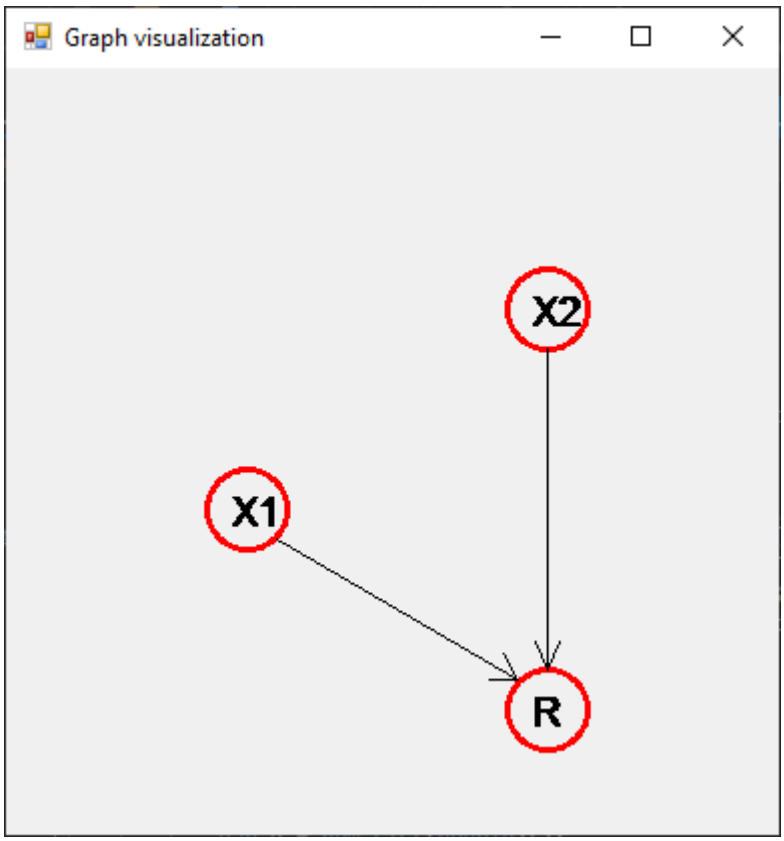


Рисунок 4.4. Направлений граф побудований на основі статистичних даних

3. Прогностична модель захворюваності на хронічну цереброваскулярну патологію [9]

ХЦП = 708 + 1,8(X_2) - 0,85(X_1),

де X_1 – викиди в атмосферне повітря хімічних речовин стаціонарними установками у поточному році;

X_2 – скидання недостатньо очищених стічних вод у поверхневі водні об’єкти на п’ять років раніше.

Рисунок 4.5. Граф побудований на основі математичної моделі

Нижче наведено таблицю 4.3. з статистичними даними по захворюванню на хронічну цереброваскулярну патологію. А також приклад роботи ПП на основі цих статистичних даних.

X1	X2	R
84,4	1,869	639,6242
106,5	4,9	626,295
2,4	1,28	708,264
9,9	4,498	707,6814
3,7	3,857	711,7976
99,7	0,82	624,731
51,839	3,317	669,9075
20,331	0,606	691,8095
17,8	0,372	693,5396
9,4	1,991	703,5938
21,68	22,96	730,9
50,959	1,568	667,5073
21,9	24,09	697,453
109,1	43,9	694,285
37,3	17,01	706,913
12,8	1,328	699,5104
205	1,075	535,685
173,401	63,78	675,4132
12,7	1,897	700,6196
773,5	199,1	408,905
576,925	131,7	454,6738
5,3	1,1	703,675

Таблиця 4.3. Статистичні дані математичної моделі

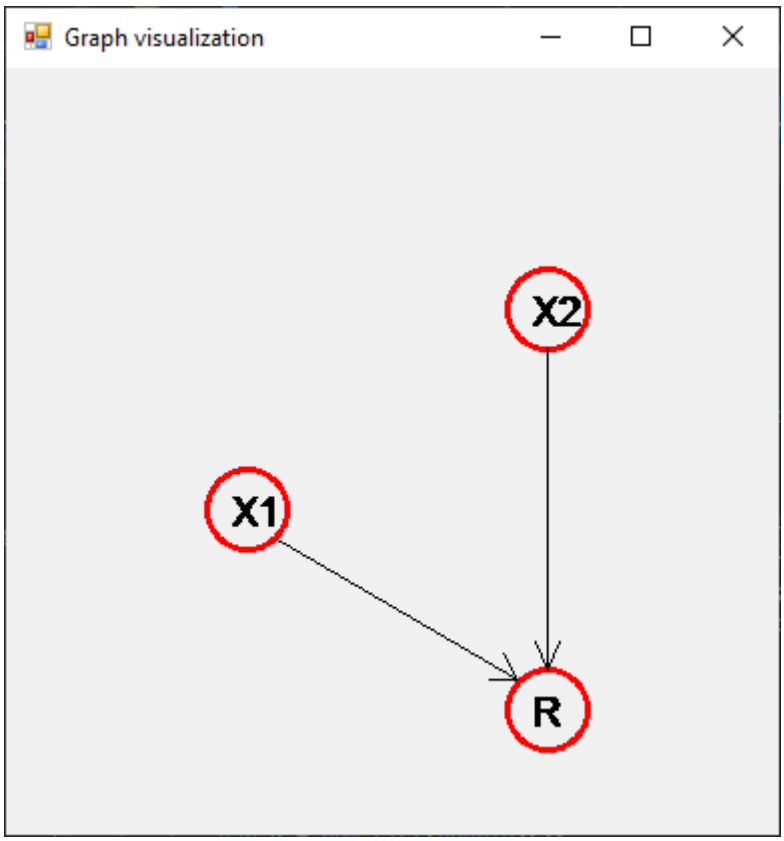


Рисунок 4.6. Направлений граф побудований на основі статистичних даних

4. Прогностична модель загального рівня ендокринної захворюваності [9]

$ЕЗ_{заг} = 565 + 1{,}13(X_1)$,

де X_1 – динаміка скидання забруднених стічних вод (за останні три роки).

Рисунок 4.7. Граф побудований на основі математичної моделі

Нижче наведено таблицю 4.4. з статистичними даними по рівню ендокринної захворюваності. А також приклад роботи ПП на основі цих статистичних даних.

X1	R
1,958	567,2125
9,81	576,0853
1,872	567,1154
4,498	570,0827
4,236	569,7867
0,833	565,9413
4,486	570,0692
0,606	565,6848
0,912	566,0306
2,603	567,9414
23,03	591,0239
1,568	566,7718
240,09	836,3017
45,43	616,3359
18,461	585,8609
1,328	566,5006
1,103	566,2464
64,17	637,5121
1,995	567,2544
199,4	790,322
230,3	825,239
0,11	565,1243

Таблиця 4.4. Статистичні дані математичної моделі

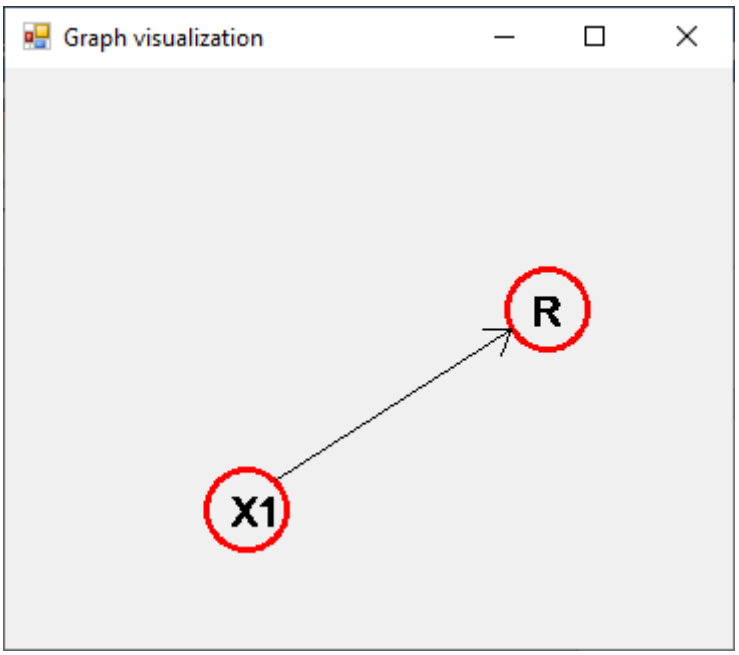


Рисунок 4.8. Направлений граф побудований на основі статистичних даних

5. Прогностична модель поширеності цукрового діабету [9]

ЦД = 4604 + 2,7(X_1),

де X_1 – динаміка скидання недостатньо очищених стічних вод, млн. $м^3$.

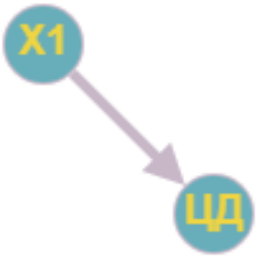


Рисунок 4.9. Граф побудований на основі математичної моделі

Нижче наведено таблицю 4.5. з статистичними даними по поширеності цукрового діабету. А також приклад роботи ПП на основі цих статистичних даних.

X1	R
1,869	4609,046
4,9	4617,23
1,28	4607,456
4,498	4616,145
3,857	4614,414
0,82	4606,214
3,317	4612,956
0,606	4605,636
0,372	4605,004
1,991	4609,376
22,96	4665,992
1,568	4608,234
24,09	4669,043
43,9	4722,53
17,01	4649,927
1,328	4607,586
1,075	4606,903
63,78	4776,206
1,897	4609,122
199,1	5141,57
131,7	4959,59
0,1	4604,27

Таблиця 4.5. Статистичні дані математичної моделі

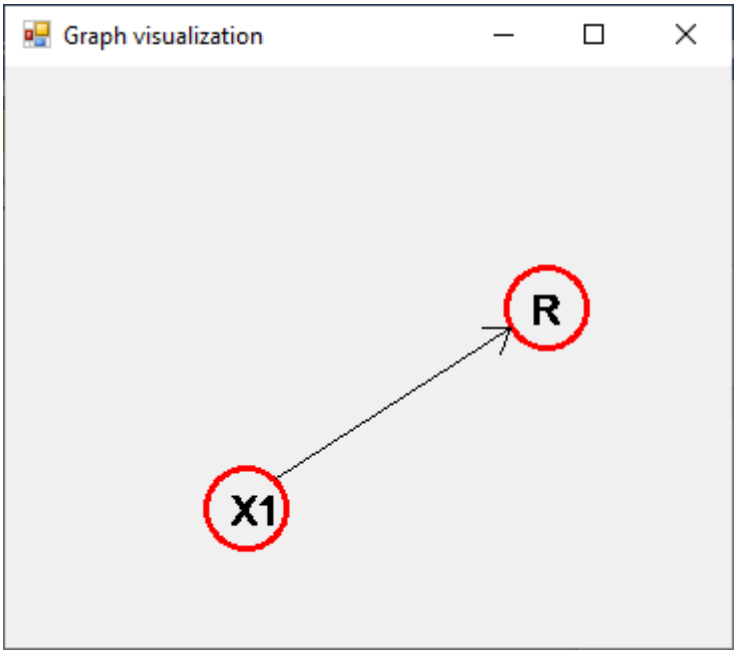


Рисунок 4.10. Направлений граф побудований на основі статистичних даних

4.2. Тестування моделей з непрямими зв’язками

1. Прогностична модель поширення корона вірусу [10]

$k = K / tD;$

$ND(ti) = NT(ti – tD)$

$NA(ti) = NT(ti – 1) – ND(ti – 1).$

де, k – наведена трансмісивність; виражає скільки один хворий в середньому заражає за день.

K – трансмісивність (передання) вірусу.

tD – середній час від зараження до ізоляції хворого.

ND(ti) – число виявлених заражених на дату ti.

NT(ti) – загальна кількість заражених на дату ti.

NA(ti) – кількість не виявлених заражених на дату ti.

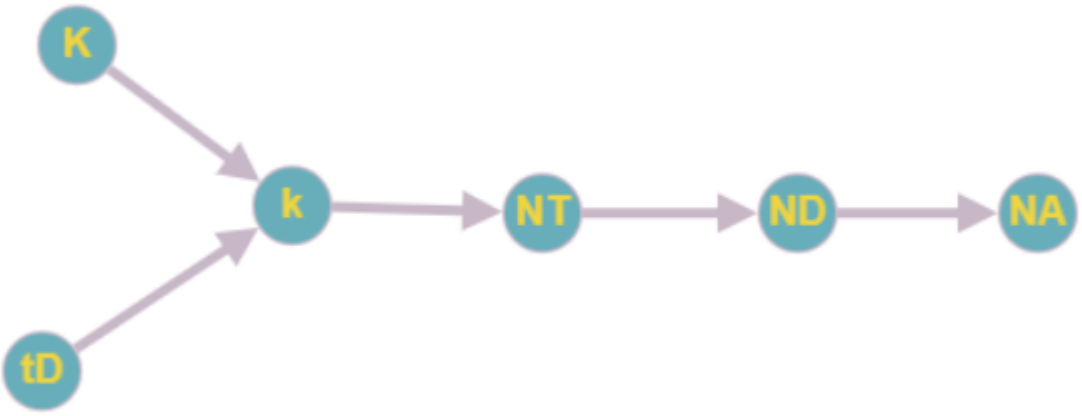


Рисунок 4.11. Граф побудований на основі математичної моделі

Нижче наведено таблицю 4.6. з статистичними даними по поширеності корона вірусу [10]. А також приклад роботи ПП на основі цих статистичних даних.

дата	кількість не виявлених заражених	загальна кількість хворих	кількість виявлених хворих	кількість загинувших
19трав.	423	723	199	1
20трав.	524	896	246	1
21трав.	650	1111	306	2
22трав.	806	1378	379	2
23трав.	999	1708	470	2
24трав.	1238	2118	583	3
25трав.	1535	2625	723	4
26трав.	1902	3254	896	5
27трав.	2358	4034	1111	6
28трав.	2923	4706	1378	6
29трав.	3328	5539	1708	6
30трав.	3831	6488	2118	10
31трав.	4370	7580	2625	12
01квіт.	4955	8825	3254	15
02квіт.	5571	10 237	4034	19
03квіт.	6203	11 825	4706	23
04квіт.	7119	13 539	5593	29
05квіт.	8054	15 622	6488	36
06квіт.	9134	17 917	7580	44
07квіт.	10 337	20 520	8825	55
08квіт.	11 695	23 466	10 237	68
09квіт.	13 229	26 799	11 825	85
10квіт.	14 974	30 569	13 593	105
11квіт.	16 977	34 837	15 622	130
12квіт.	19 216	39 676	17 917	161
13квіт.	21 759	45 152	20 520	205
14квіт.	24 632	51 353	23 466	205
15квіт.	27 887	58 373	26 799	281
16квіт.	31 574	66 321	30 569	328
17квіт.	35 752	75 320	34 837	381
18квіт.	40 482	85 509	39 676	441
19квіт.	45 833	97 046	45 152	508
20квіт.	51 894	110 109	51 353	584
21квіт.	58 756	124 899	58 373	671

Таблиця 4.6. Статистичні дані математичної моделі

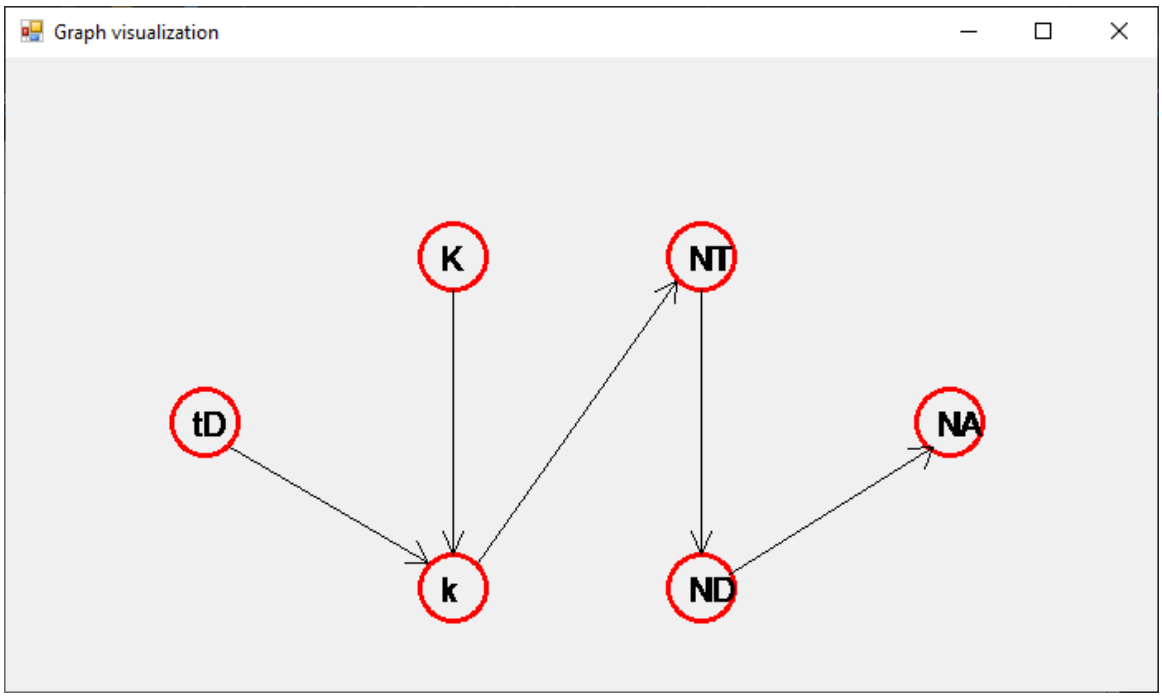


Рисунок 4.12. Направлений граф побудований на основі статистичних даних

2. Математична модель

A = K + C

B = A / D

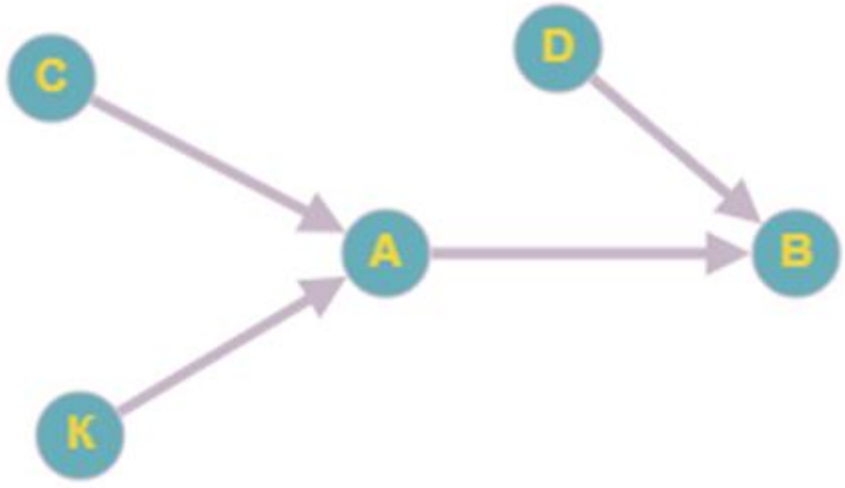


Рисунок 4.13. Граф побудований на основі математичної моделі

Нижче наведено таблицю 4.7 з статистичними даними. А також приклад роботи ПП на основі цих статистичних даних.

K	C	D	A	B
4	8	6	12	72
9	9	8	18	144
4	4	7	8	56
8	5	9	13	117
6	2	7	8	56
2	8	5	10	50
7	8	5	15	75
9	9	10	18	180
5	5	8	10	80
4	7	7	11	77
5	9	10	14	140
1	1	10	2	20
2	8	7	10	70
5	7	9	12	108
5	8	6	13	78
6	9	10	15	150
4	10	5	14	70
9	9	10	18	180
7	6	8	13	104
1	3	8	4	32
2	7	5	9	45
9	2	9	11	99
4	3	5	7	35
7	3	6	10	60
2	6	10	8	80
2	8	9	10	90
1	1	7	2	14
10	10	7	20	140

Таблиця 4.7. Статистичні дані математичної моделі

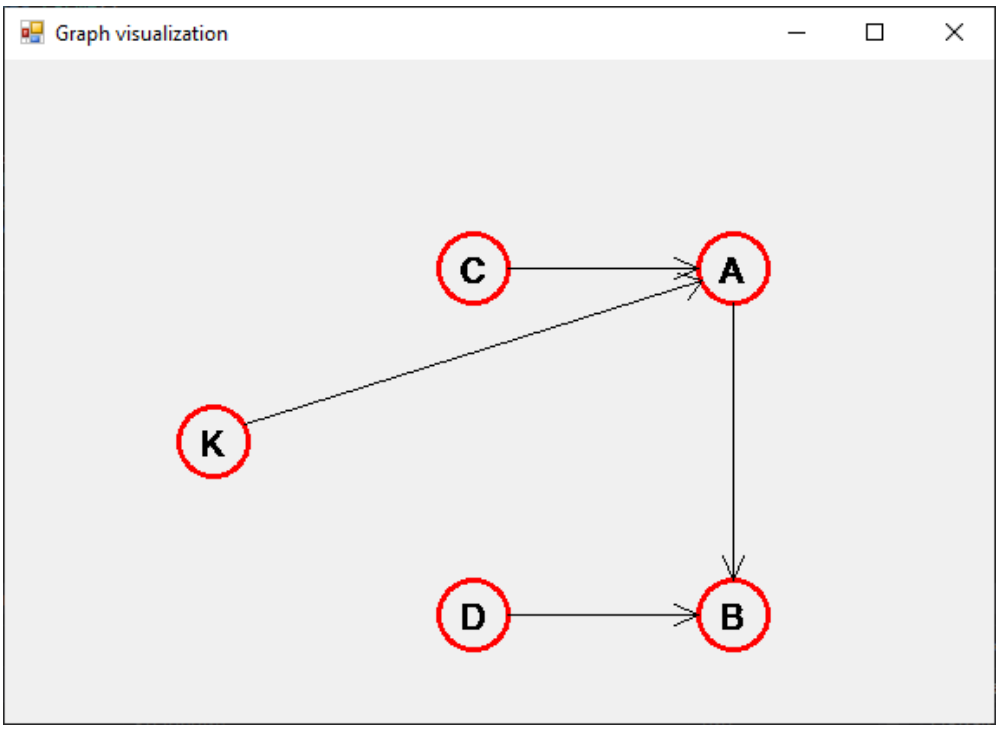


Рисунок 4.14. Направлений граф побудований на основі статистичних даних

В заключенні наведено таблицю 4.8. з статистикою роботи даного ПП прикладі прогностичних моделей

Назва прогностичної моделі	ГІМ	МІ	ХЦП	ЕЗ	ЦД	Covid	ABCDK
Зв'язок між параметрами моделі	Прямий	Прямий	Прямий	Прямий	Прямий	Непрямий	Непрямий
Кількість правильно встановлених причинно-наслідкових структур	100%	100%	100%	100%	100%	100%	100%
Максимальний доступний вплив приховано змінної	30%	50%	35%	50%	50%	15%	20%

Таблиця 4.8. Статистика роботи ПП

4.3. Робота з інтерфейсом

Запустивши програму перед користувачем відкривається вікно в якому зверху знаходиться текст з привітанням і короткою інформацією про програмний продукт.

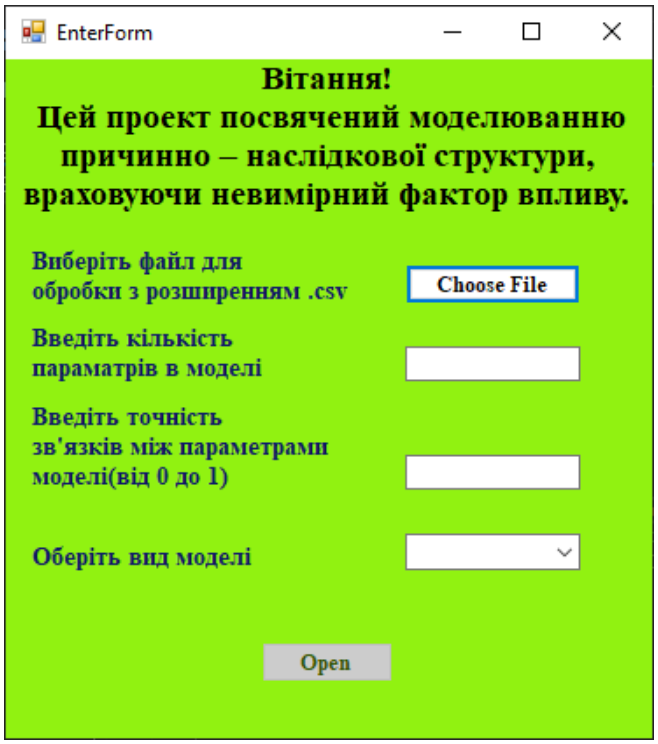


Рисунок 4.15. Коротка інформація про ПП

Нижче знаходиться кнопка для вибору файлу з вхідними даними. Важливо пам’ятати що файл повинен бути з розширенням .csv

Рисунок 4.16. Кнопка пошуку файлу з вхідними даними

Нижче знаходиться блок з полями які потрібно заповнити згідно з написами біля них:

- Кількість вершин графа (кількість змінних);
- Коефіцієнт точності (коефіцієнт оцінки між параметрами);
- Вказати прямий/непрямий зв’язок.

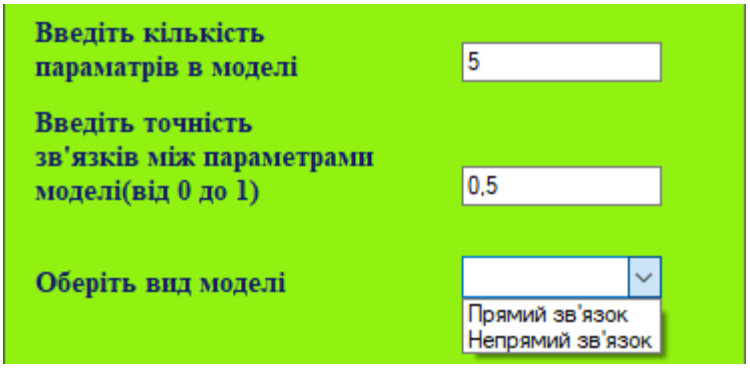


Рисунок 4.17. – Блок полів

А вже потім нажати кнопку Open,

Рисунок 4.18. – Кнопка відкриття

що переведе користувача в інше вікно, де вже буде продемонстровано візуалізацію ПНС, тобто напрямлений граф.

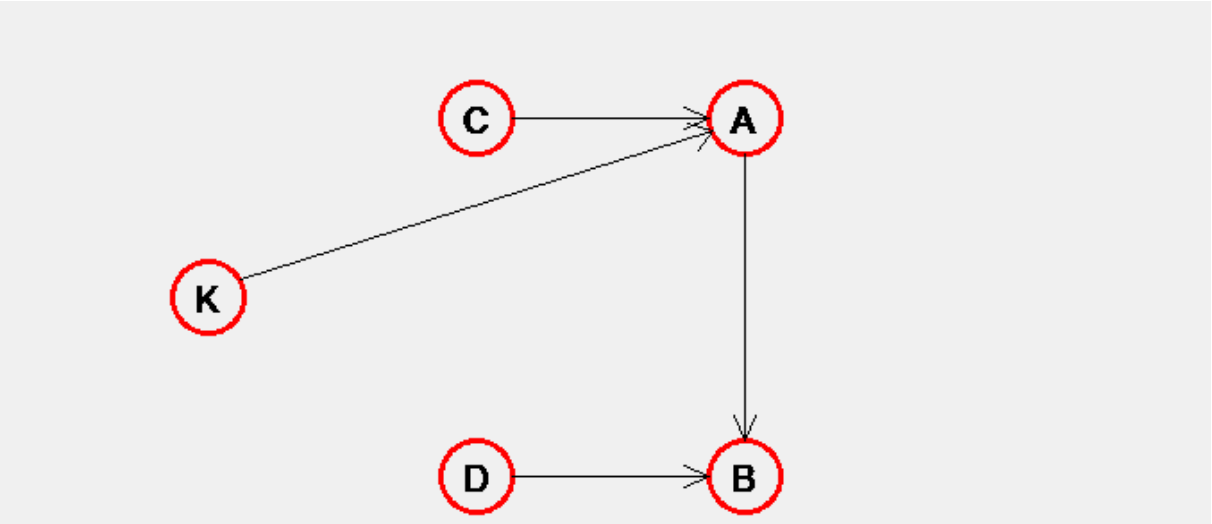


Рисунок 4.19. – Візуалізація графа

4.4. Висновок до розділу

В результаті було спроектовано та розроблено програмний продукт, який встановлює та візуалізує у вигляді причинного графа причинно-наслідкові структури моделі, що подається на вхід програми у вигляді набору даних.

Отже створений програмний продукт видає коректні результати і готовий до використання в практичних цілях.

5. ІНСТРУМЕНТИ ЯКІ БУЛИ ВИКОРИСТАНІ ДЛЯ РОЗРОБКИ

5.1. Розробка модуля причинно-наслідкових структур

Програмне забезпечення було створено в середовищі розробки Visual Studio 2019 в C # на основі технології .Net [11].

Microsoft Visual Studio – це серія продуктів Microsoft, які включають інтегроване середовище розробки програмного забезпечення та багато інших інструментів. Ці продукти дозволяють розробляти консольні та графічні програми, включаючи підтримку технології Windows Forms, а також веб-сайти, веб-програми та веб-служби у вихідному та керованому коді для всіх платформ. Підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

.Net Framework - це програмна технологія, запропонована корпорацією Майкрософт як платформа для створення як звичайних, так і веб-додатків. Багато в чому це продовження ідей та принципів, закладених у технології Java. Одна з ідей. NET – це сумісність служб, написаних різними мовами. Хоча Microsoft і визначає цю функцію як перевагу .NET, платформа Java має однакову продуктивність.

Кожна бібліотека (колекція) у .NET має власну версію, яка вирішує будь-які конфлікти між версіями колекцій.

C # – об'єктно-орієнтована мова програмування із захищеною системою набору тексту для програмної технології .NET. Розроблений Андерсом Галесбергом, Скоттом Вільтамутом та Пітером Голде під егідою Microsoft Research (належить Microsoft).

5.2. Розробка програмного інтерфейсу

Інтерфейс програми був створений за допомогою інструменту Windows Form [11].

Windows Forms – це інтерфейс прикладного програмування (API), відповідальний за графічний інтерфейс користувача, і є частиною Microsoft .NET Framework. Цей інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows, створюючи конверт для API Win32 в керованому коді.

У .NET Framework Windows Forms працює в просторі імен System.Windows.Forms.

ВИСНОВОК

В результаті було спроектовано та розроблено програмний продукт, який встановлює та візуалізує у вигляді направленого графа причинно-наслідкові структури моделі, що подається на вхід програми у вигляді набору статистичних даних. Також виявлено та встановлено обмеження, яким мають відповідати припущення щодо моделей та набору вхідних даних. Програмний продукт розроблений для виявлення як прямих так і для непрямих залежностей між параметрами, що дозволяє значно розширити спектр вхідних моделей.

Також було проведено контрольні розрахунки з використанням розроблених математичних моделей, які відповідають вищезазначеним обмеженням до вхідних даних та досліджуваних моделей, на основі реальних статистичних даних. За результатами вихідних даних програми було встановлено ідентичність між причинно-наслідковою структурою, яку відображає математична модель та між причинно-наслідковою структурою, яку відображає результат роботи програми на основі набору статистичних даних, що дає підстави для ствердження того, що підібрані методи, та запропонована модифікація PC-алгоритму відображає коректні результати.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Machine Learning Goes Causal I: Why Causality Matters [Електронний ресурс]: - Режим доступу: https://www.statworx.com/ch/blog/machine-learning-goes-causal-i-why-causality-matters/
2. Causality – The Next Most Important Thing in AL/ML [Електронний ресурс]: - Режим доступу: https://mobilemonitoringsolutions.com/causality-the-next-most-important-thing-in-ai-ml/
3. Spirtes, P., & Zhang, K. Causal discovery and inference: concepts and recent methodological advances. [Електронний ресурс] / Spirtes, P., & Zhang, K. — 2016. — Режим доступу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4841209/
4. Directed Acyclic Graph (DAGs). Chapter 4. More Basics [Електронний ресурс]: - Режим доступу: https://ericsink.com/vcbe/html/directed_acyclic_graphs.html
5. Беспала О. М. Інструментарій причинно-наслідкового висновку: огляд та перспективи [Електронний ресурс] / Беспала О. М. — 2020. — Режим доступу: http://usim.org.ua/?page_id=13164&lang=ru
6. Fan, Y., Chen, J., Shirkey, G., John, R., Wu, S. R., Park, H., & Shao, C. Applications of structural equation modeling (SEM) in ecological studies: an updated review [Електронний ресурс] / Fan, Y., Chen, J., Shirkey, G., John, R., Wu, S. R., Park, H., & Shao, C. — 2016. — Режим доступу: https://ecologicalprocesses.springeropen.com/articles/10.1186/s13717-016-0063-3
7. Causal CMD – CCD Docs [Електронний ресурс]: - Режим доступу: https://bd2kccd.github.io/docs/causal-cmd/
8. Data Science – Center For Causal Discovery [Електронний ресурс]: - Режим доступу: https://www.ccd.pitt.edu/data-science/

9. В. Г. Сліпченко, О. В. Коваль, Л. Г. Полягушко. Екологічний моніторинг / В. Г. Сліпченко, О. В. Коваль, Л. Г. Полягушко., 2018 – 304 с. ISBN 978-966-622-869-0
10. Коронавирусная инфекция: моделирование и прогноз [Електронний ресурс]: - Режим доступу: https://www.kommersant.ru/doc/4322667
11. Visual Studio [Електронний ресурс]: - Режим доступу: https://visualstudio.microsoft.com/ru/
12. Didelez, V., & Pigeot, I. Judea pearl: Causality: models, reasoning, and inference [Електронний ресурс] / Didelez, V., & Pigeot, I — 2016. — Режим доступу: https://books.google.com.ua/books?hl=uk&lr=&id=f4nuexsNVZIC&oi=fnd&pg=PR1&dq=causality&ots=y3DQWntync&sig=wnfrxg_wjDys5OnFmFtCCa2VBMo&redir_esc=y#v=onepage&q=causality&f=false
13. Shimizu, S., Inazumi, T., Sogawa, Y., Hyvärinen, A., Kawahara, Y., Washio, T., & Bollen, K. DirectLiNGAM: A direct method for learning a linear non-Gaussian structural equation model. [Електронний ресурс] / Shimizu, S., Inazumi, T., Sogawa, Y., Hyvärinen, A., Kawahara, Y., Washio, T., & Bollen, K. — 2011. — Режим доступу: https://www.jmlr.org/papers/volume12/shimizu11a/shimizu11a.pdf
14. Vineet K. Raghu, Allen Poon, Panayiotis V. Benos. Evaluation of Causal Structure Learning Methods on Mixed Data Tepes [Електронний ресурс] / Vineet K. Raghu, Allen Poon, Panayiotis V. — 2018. — Режим доступу: http://proceedings.mlr.press/v92/raghu18a.html
15. Yao, L., Chu, Z., Li, S., Li, Y., Gao, J., & Zhang, A. A survey on causal inference. [Електронний ресурс] / Yao, L., Chu, Z., Li, S., Li, Y., Gao, J., & Zhang, A. — 2020. — Режим доступу: https://arxiv.org/abs/2002.02770
16. Pearl, J. Causal inference in statistics: An overview. [Електронний ресурс] / Pearl, J. — 2009. — Режим доступу: https://projecteuclid.org/euclid.ssu/1255440554http://projecteuclid.org/euclid.ssu/1255440554

17. Sobel, M. E. An introduction to causal inference. [Електронний ресурс] / Sobel, M. E. — 2009. — Режим доступу: https://journals.sagepub.com/doi/abs/10.1177/0049124196024003004
18. Parafita, Á., & Vitrià, J. Causal Inference with Deep Causal Graphs. [Електронний ресурс] / Parafita, Á., & Vitrià, J. — 2020. — Режим доступу: https://arxiv.org/abs/2006.08380
19. Lee, K., Bargagli-Stoffi, F. J., & Dominici, F. Causal rule ensemble: Interpretable inference of heterogeneous treatment effects. [Електронний ресурс] / Lee, K., Bargagli-Stoffi, F. J., & Dominici, F. — 2020. — Режим доступу: https://arxiv.org/abs/2009.09036
20. Smith, M. J., Maringe, C., Rachet, B., Mansournia, M. A., Zivich, P. N., Cole, S. R., & Luque-Fernandez, M. A. Tutorial: Introduction to computational causal inference using reproducible Stata, R and Python code. [Електронний ресурс] / Smith, M. J., Maringe, C., Rachet, B., Mansournia, M. A., Zivich, P. N., Cole, S. R., & Luque-Fernandez, M. A. — 2020. — Режим доступу: https://arxiv.org/abs/2012.09920

ДОДАТОК А

Моделювання причинно-наслідкової структури, враховуючи невимірний фактор впливу

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР71229_21Б

Аркушів 2

Київ – 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР	LoginForm.cs	Компонент, в якому записуються всі праметри
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР	Form1.cs	Компонент обробки даних
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР	Data Bank.cs	Компонент зберігання даних

ДОДАТОК Б

Моделювання причинно-наслідкової структури, враховуючи невимірний фактор впливу

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР71229_21Б 12-1

Аркушів 30

Київ – 2021

LoginForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

//Запис параметрів для подальшої роботи
namespace Graph
{
   public partial class LoginForm : Form
   {
      public LoginForm()
      {
         InitializeComponent();
         button2.Enabled = false;
      }

      private void button1_Click(object sender, EventArgs e)
      {
         if (openFileDialog1.ShowDialog() == DialogResult.OK)
         {
            DataBank.Adress = openFileDialog1.FileName;
         }
      }

      private void textBox1_TextChanged(object sender, EventArgs e)
      {
         if ((textBox1.Text.Length != 0) && (textBox2.Text.Length != 0))
```

```
            button2.Enabled = true;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            DataBank.Columns = textBox1.Text;
            DataBank.Coefficient = textBox2.Text;
            DataBank.Connection = comboBox1.Text;

            this.Hide();
            Form1 graph = new Form1();
            graph.Show();
        }

    }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Graph
{
   public partial class Form1 : Form
   {
      Bitmap bmp;
      Graphics graph;
      Pen pen;

      public Form1()
      {
         InitializeComponent();
         Draw();
      }
//Функція оцінки змінних
      public static double Correl(List<double> x, List<double> y)
      {

         double[] array_xy = new double[x.Count];
         double[] array_xp2 = new double[x.Count];
         double[] array_yp2 = new double[x.Count];

         for (int i = 0; i < x.Count; i++)
            array_xy[i] = x[i] * y[i];
```

```
for (int i = 0; i < x.Count; i++)
    array_xp2[i] = Math.Pow(x[i], 2.0);

for (int i = 0; i < x.Count; i++)
    array_yp2[i] = Math.Pow(y[i], 2.0);

double sum_x = 0;
double sum_y = 0;

foreach (double n in x)
    sum_x += n;

foreach (double n in y)
    sum_y += n;

double sum_xy = 0;

foreach (double n in array_xy)
    sum_xy += n;

double sum_xpow2 = 0;

foreach (double n in array_xp2)
    sum_xpow2 += n;

double sum_ypow2 = 0;

foreach (double n in array_yp2)
    sum_ypow2 += n;

double Ex2 = Math.Pow(sum_x, 2.00);
double Ey2 = Math.Pow(sum_y, 2.00);

double Correl =
```

```
            (x.Count * sum_xy - sum_x * sum_y) /
            Math.Sqrt((x.Count * sum_xpow2 - Ex2) * (x.Count * sum_ypow2 - Ey2));

            return (Correl);
        }
//Функція по малюванню графа
        private void Draw()
        {
            string[] csvLine = System.IO.File.ReadAllLines(DataBank.Adress);
            int Column = Convert.ToInt32(DataBank.Columns);
            double[,] mass = new double[Column, Column];
            double t = Convert.ToDouble(DataBank.Coefficient);


            var X = new List<double>();
            var Y = new List<double>();
            var Z = new List<double>();
            var K = new List<double>();
            var P = new List<double>();
            var Z1 = new List<double>();
            var Name = new List<string>();
            int Counter = 0;

            if (Column == 2)
            {
                for (int i = 0; i < csvLine.Length; i++)
                {
                    string[] rawData = csvLine[i].Split(';');
                    Name.Add(rawData[0]);
                }

                for (int i = 0; i < csvLine.Length; i++)
                {
                    string[] rawData = csvLine[i].Split(';');
```

```
    double x = Convert.ToDouble(rawData[1]);
    X.Add(x);

    double y = Convert.ToDouble(rawData[2]);
    Y.Add(y);

};

mass[0, 0] = Correl(X, X);
mass[0, 1] = Correl(X, Y);

mass[1, 0] = Correl(Y, X);
mass[1, 1] = Correl(Y, Y);

for (int i = 0; i < mass.GetLength(0); i++)
{
    for (int j = 0; j < mass.GetLength(1); j++)
    {
        if (mass[i, j] < 0)
        {
            mass[i, j] = -mass[i, j];
        }
    }
}

for (int i = 0; i < mass.GetLength(0); i++)
{
    for (int j = 0; j < mass.GetLength(0); j++)
    {
        if (mass[i, j] > t)
        {
            mass[i, j] = 1;
        }
        if (j <= i)
        {
```

```
            mass[i, j] = 0;
        }
        if (mass[i, j] <= t)
        {
            mass[i, j] = 0;
        }
    }
}

if(DataBank.Connection == "Прямий зв\'язок")
{
    for (int i = 0; i < mass.GetLength(0); i++)
    {
        for (int j = (Column - 1); j >= 0; j--)
        {
            if (mass[i, j] == 1)
            {
                int p = j - 1;

                while (p >= 0)
                {
                    mass[i, p] = 0;
                    p--;
                }
            }
        }
    }
}

if (DataBank.Connection == "Непрямий зв\'язок")
{
    for (int i = 0; i < mass.GetLength(0); i++)
    {
        for (int j = 0; j < mass.GetLength(1); j++)
        {
```

```
            if (mass[i, j] == 1)
            {
                int p = j + 1;

                while (p < Column)
                {
                    mass[i, p] = 0;
                    p++;
                }
            }
        }
    }
}

bmp = new Bitmap(pictureBox1.Width, pictureBox1.Height);
graph = Graphics.FromImage(bmp);
pen = new Pen(Color.Red, 3f);
Font font = new Font("Arial", 15);

graph.DrawEllipse(pen, 100, 200, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 110, 210);
Counter++;
graph.DrawEllipse(pen, 250, 100, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 260, 110);

pen = new Pen(Color.Black, 1f);
for (int i = 0; i < mass.GetLength(0); i++)
{
    for (int j = 0; j < mass.GetLength(0); j++)
    {
        if (mass[i, j] == 1)
        {
            if (i == 0 && j == 1)
            {
                graph.DrawLine(pen, 135, 205, 252, 130);
```

```
                graph.DrawLine(pen, 252, 130, 238, 130);
                graph.DrawLine(pen, 252, 130, 247, 143);
            }

        }
    }
}
if (Column == 3)
{
    for (int i = 0; i < csvLine.Length; i++)
    {
        string[] rawData = csvLine[i].Split(';');
        Name.Add(rawData[0]);
    }

    for (int i = 0; i < csvLine.Length; i++)
    {
        string[] rawData = csvLine[i].Split(';');

        double x = Convert.ToDouble(rawData[1]);
        X.Add(x);

        double y = Convert.ToDouble(rawData[2]);
        Y.Add(y);

        double z = Convert.ToDouble(rawData[3]);
        Z.Add(z);
    };

    mass[0, 0] = Correl(X, X);
    mass[0, 1] = Correl(X, Y);
    mass[0, 2] = Correl(X, Z);

    mass[1, 0] = Correl(Y, X);
```

```
mass[1, 1] = Correl(Y, Y);
mass[1, 2] = Correl(Y, Z);

mass[2, 0] = Correl(Z, X);
mass[2, 1] = Correl(Z, Y);
mass[2, 2] = Correl(Z, Z);

for (int i = 0; i < mass.GetLength(0); i++)
{
   for (int j = 0; j < mass.GetLength(1); j++)
   {
      if (mass[i, j] < 0)
      {
         mass[i, j] = -mass[i, j];
      }
   }
}

for (int i = 0; i < mass.GetLength(0); i++)
{
   for (int j = 0; j < mass.GetLength(0); j++)
   {
      if (mass[i, j] > t)
      {
         mass[i, j] = 1;
      }
      if (j <= i)
      {
         mass[i, j] = 0;
      }
      if (mass[i, j] <= t)
      {
         mass[i, j] = 0;
      }
   }
```

```
}

if (DataBank.Connection == "Прямий зв\'язок")
{
   for (int i = 0; i < mass.GetLength(0); i++)
   {
      for (int j = (Column - 1); j >= 0; j--)
      {
         if (mass[i, j] == 1)
         {
            int p = j - 1;

            while (p >= 0)
            {
               mass[i, p] = 0;
               p--;
            }
         }
      }
   }
}

if (DataBank.Connection == "Непрямий зв\'язок")
{
   for (int i = 0; i < mass.GetLength(0); i++)
   {
      for (int j = 0; j < mass.GetLength(1); j++)
      {
         if (mass[i, j] == 1)
         {
            int p = j + 1;

            while (p < Column)
            {
               mass[i, p] = 0;
```

```
                p++;
            }
        }
    }
}

bmp = new Bitmap(pictureBox1.Width, pictureBox1.Height);
graph = Graphics.FromImage(bmp);
pen = new Pen(Color.Red, 3f);
Font font = new Font("Arial", 15);

graph.DrawEllipse(pen, 100, 200, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 110, 210);
Counter++;
graph.DrawEllipse(pen, 250, 100, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 260, 110);
Counter++;
graph.DrawEllipse(pen, 250, 300, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 260, 310);

pen = new Pen(Color.Black, 1f);
for (int i = 0; i < mass.GetLength(0); i++)
{
    for (int j = 0; j < mass.GetLength(0); j++)
    {
        if (mass[i, j] == 1)
        {
            if (i == 0 && j == 1)
            {
                graph.DrawLine(pen, 135, 205, 252, 130);
                graph.DrawLine(pen, 135, 205, 139, 191);
                graph.DrawLine(pen, 135, 205, 149, 205);
            }
            if (i == 0 && j == 2)
```

```
                {
                    graph.DrawLine(pen, 135, 235, 255, 305);
                    graph.DrawLine(pen, 255, 305, 241, 305);
                    graph.DrawLine(pen, 255, 305, 248, 292);
                }
                if (i == 1 && j == 2)
                {
                    graph.DrawLine(pen, 270, 140, 270, 300);
                    graph.DrawLine(pen, 270, 300, 264, 286);
                    graph.DrawLine(pen, 270, 300, 276, 286);
                }
            }
        }
    }
}
if (Column == 4)
{

    for (int i = 0; i < csvLine.Length; i++)
    {
        string[] rawData = csvLine[i].Split(';');
        Name.Add(rawData[0]);
    }

    for (int i = 0; i < csvLine.Length; i++)
    {
        string[] rawData = csvLine[i].Split(';');

        double x = Convert.ToDouble(rawData[1]);
        X.Add(x);

        double y = Convert.ToDouble(rawData[2]);
        Y.Add(y);

        double z = Convert.ToDouble(rawData[3]);
```

```
    Z.Add(z);

    double k = Convert.ToDouble(rawData[4]);
    K.Add(k);

};

mass[0, 0] = Correl(X, X);
mass[0, 1] = Correl(X, Y);
mass[0, 2] = Correl(X, Z);
mass[0, 3] = Correl(X, K);

mass[1, 0] = Correl(Y, X);
mass[1, 1] = Correl(Y, Y);
mass[1, 2] = Correl(Y, Z);
mass[1, 3] = Correl(Y, K);

mass[2, 0] = Correl(Z, X);
mass[2, 1] = Correl(Z, Y);
mass[2, 2] = Correl(Z, Z);
mass[2, 3] = Correl(Z, K);

mass[3, 0] = Correl(K, X);
mass[3, 1] = Correl(K, Y);
mass[3, 2] = Correl(K, Z);
mass[3, 3] = Correl(K, K);

for (int i = 0; i < mass.GetLength(0); i++)
{
    for (int j = 0; j < mass.GetLength(1); j++)
    {
        if (mass[i, j] < 0)
        {
            mass[i, j] = -mass[i, j];
        }
```

```
    }
}

for (int i = 0; i < mass.GetLength(0); i++)
{
    for (int j = 0; j < mass.GetLength(0); j++)
    {
        if (mass[i, j] > t)
        {
            mass[i, j] = 1;
        }
        if (j <= i)
        {
            mass[i, j] = 0;
        }
        if (mass[i, j] <= t)
        {
            mass[i, j] = 0;
        }
    }
}

if (DataBank.Connection == "Прямий зв\'язок")
{
    for (int i = 0; i < mass.GetLength(0); i++)
    {
        for (int j = (Column - 1); j >= 0; j--)
        {
            if (mass[i, j] == 1)
            {
                int p = j - 1;

                while (p >= 0)
                {
                    mass[i, p] = 0;
```

```
                p--;
            }
        }
    }
}

if (DataBank.Connection == "Непрямий зв\'язок")
{
    for (int i = 0; i < mass.GetLength(0); i++)
    {
        for (int j = 0; j < mass.GetLength(1); j++)
        {
            if (mass[i, j] == 1)
            {
                int p = j + 1;

                while (p < Column)
                {
                    mass[i, p] = 0;
                    p++;
                }
            }
        }
    }
}

bmp = new Bitmap(pictureBox1.Width, pictureBox1.Height);
graph = Graphics.FromImage(bmp);
pen = new Pen(Color.Red, 3f);
Font font = new Font("Arial", 15);

graph.DrawEllipse(pen, 100, 200, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 110, 210);
Counter++;
```

```
graph.DrawEllipse(pen, 250, 100, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 260, 110);
Counter++;
graph.DrawEllipse(pen, 250, 300, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 260, 310);
Counter++;
graph.DrawEllipse(pen, 400, 100, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 410, 110);

pen = new Pen(Color.Black, 1f);
for (int i = 0; i < mass.GetLength(0); i++)
{
   for (int j = 0; j < mass.GetLength(0); j++)
   {
      if (mass[i, j] == 1)
      {
         if (i == 0 && j == 1)
         {
            graph.DrawLine(pen, 135, 205, 252, 130);
            graph.DrawLine(pen, 135, 205, 139, 191);
            graph.DrawLine(pen, 135, 205, 149, 205);
         }
         if (i == 0 && j == 2)
         {
            graph.DrawLine(pen, 135, 235, 255, 305);
            graph.DrawLine(pen, 255, 305, 241, 305);
            graph.DrawLine(pen, 255, 305, 248, 292);
         }
         if (i == 0 && j == 3)
         {
            graph.DrawLine(pen, 138, 210, 402, 127);
            graph.DrawLine(pen, 402, 127, 388, 123);
            graph.DrawLine(pen, 402, 127, 394, 138);
         }
         if (i == 1 && j == 2)
```

```
                {
                    graph.DrawLine(pen, 270, 140, 270, 300);
                    graph.DrawLine(pen, 270, 300, 264, 286);
                    graph.DrawLine(pen, 270, 300, 276, 286);
                }
                if (i == 1 && j == 3)
                {
                    graph.DrawLine(pen, 290, 120, 400, 120);
                    graph.DrawLine(pen, 400, 120, 386, 114);
                    graph.DrawLine(pen, 400, 120, 386, 126);
                }
                if (i == 2 && j == 3)
                {
                    graph.DrawLine(pen, 285, 305, 405, 135);
                    graph.DrawLine(pen, 405, 135, 392, 141);
                    graph.DrawLine(pen, 405, 135, 404, 148);
                }
            }
        }
    }
}
if (Column == 5)
{
    for (int i = 0; i < csvLine.Length; i++)
    {
        string[] rawData = csvLine[i].Split(';');
        Name.Add(rawData[0]);
    }

    for (int i = 0; i < csvLine.Length; i++)
    {
        string[] rawData = csvLine[i].Split(';');

        double x = Convert.ToDouble(rawData[1]);
        X.Add(x);
```

```
    double y = Convert.ToDouble(rawData[2]);
    Y.Add(y);

    double z = Convert.ToDouble(rawData[3]);
    Z.Add(z);

    double k = Convert.ToDouble(rawData[4]);
    K.Add(k);

    double p = Convert.ToDouble(rawData[5]);
    P.Add(p);
};

mass[0, 0] = Correl(X, X);
mass[0, 1] = Correl(X, Y);
mass[0, 2] = Correl(X, Z);
mass[0, 3] = Correl(X, K);
mass[0, 4] = Correl(X, P);

mass[1, 0] = Correl(Y, X);
mass[1, 1] = Correl(Y, Y);
mass[1, 2] = Correl(Y, Z);
mass[1, 3] = Correl(Y, K);
mass[1, 4] = Correl(Y, P);

mass[2, 0] = Correl(Z, X);
mass[2, 1] = Correl(Z, Y);
mass[2, 2] = Correl(Z, Z);
mass[2, 3] = Correl(Z, K);
mass[2, 4] = Correl(Z, P);

mass[3, 0] = Correl(K, X);
mass[3, 1] = Correl(K, Y);
mass[3, 2] = Correl(K, Z);
```

```
mass[3, 3] = Correl(K, K);
mass[3, 4] = Correl(K, P);

mass[4, 0] = Correl(P, X);
mass[4, 1] = Correl(P, Y);
mass[4, 2] = Correl(P, Z);
mass[4, 3] = Correl(P, K);
mass[4, 4] = Correl(P, P);

for (int i = 0; i < mass.GetLength(0); i++)
{
   for (int j = 0; j < mass.GetLength(1); j++)
   {
      if (mass[i, j] < 0)
      {
         mass[i, j] = -mass[i, j];
      }
   }
}

for (int i = 0; i < mass.GetLength(0); i++)
{
   for (int j = 0; j < mass.GetLength(0); j++)
   {
      if (mass[i, j] > t)
      {
         mass[i, j] = 1;
      }
      if (j <= i)
      {
         mass[i, j] = 0;
      }
      if (mass[i, j] <= t)
      {
         mass[i, j] = 0;
```

```
        }
    }
}

if (DataBank.Connection == "Прямий зв\'язок")
{
    for (int i = 0; i < mass.GetLength(0); i++)
    {
        for (int j = (Column - 1); j >= 0; j--)
        {
            if (mass[i, j] == 1)
            {
                int p = j - 1;

                while (p >= 0)
                {
                    mass[i, p] = 0;
                    p--;
                }
            }
        }
    }
}

if (DataBank.Connection == "Непрямий зв\'язок")
{
    for (int i = 0; i < mass.GetLength(0); i++)
    {
        for (int j = 0; j < mass.GetLength(1); j++)
        {
            if (mass[i, j] == 1)
            {
                int p = j + 1;

                while (p < Column)
```

```
            {
                mass[i, p] = 0;
                p++;
            }
        }
    }
}

bmp = new Bitmap(pictureBox1.Width, pictureBox1.Height);
graph = Graphics.FromImage(bmp);
pen = new Pen(Color.Red, 3f);
Font font = new Font("Arial", 15);

graph.DrawEllipse(pen, 100, 200, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 110, 210);
Counter++;
graph.DrawEllipse(pen, 250, 100, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 260, 110);
Counter++;
graph.DrawEllipse(pen, 250, 300, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 260, 310);
Counter++;
graph.DrawEllipse(pen, 400, 100, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 410, 110);
Counter++;
graph.DrawEllipse(pen, 400, 300, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 410, 310);

pen = new Pen(Color.Black, 1f);
for (int i = 0; i < mass.GetLength(0); i++)
{
    for (int j = 0; j < mass.GetLength(0); j++)
    {
        if (mass[i, j] == 1)
```

```
{
    if (i == 0 && j == 1)
    {
        graph.DrawLine(pen, 135, 205, 252, 130);
        graph.DrawLine(pen, 135, 205, 139, 191);
        graph.DrawLine(pen, 135, 205, 149, 205);
    }
    if (i == 0 && j == 2)
    {
        graph.DrawLine(pen, 135, 235, 255, 305);
        graph.DrawLine(pen, 255, 305, 241, 305);
        graph.DrawLine(pen, 255, 305, 248, 292);
    }
    if (i == 0 && j == 3)
    {
        graph.DrawLine(pen, 138, 210, 402, 127);
        graph.DrawLine(pen, 402, 127, 388, 123);
        graph.DrawLine(pen, 402, 127, 394, 138);
    }
    if (i == 0 && j == 4)
    {
        graph.DrawLine(pen, 137, 230, 400, 312);
        graph.DrawLine(pen, 400, 312, 391, 302);
        graph.DrawLine(pen, 400, 312, 387, 315);
    }
    if (i == 1 && j == 2)
    {
        graph.DrawLine(pen, 270, 140, 270, 300);
        graph.DrawLine(pen, 270, 300, 264, 286);
        graph.DrawLine(pen, 270, 300, 276, 286);
    }
    if (i == 1 && j == 3)
    {
        graph.DrawLine(pen, 290, 120, 400, 120);
        graph.DrawLine(pen, 400, 120, 386, 114);
```

```
                graph.DrawLine(pen, 400, 120, 386, 126);
            }
            if (i == 1 && j == 4)
            {
                graph.DrawLine(pen, 285, 135, 405, 305);
                graph.DrawLine(pen, 405, 305, 390, 299);
                graph.DrawLine(pen, 405, 305, 403, 289);
            }
            if (i == 2 && j == 3)
            {
                graph.DrawLine(pen, 285, 305, 405, 135);
                graph.DrawLine(pen, 405, 135, 392, 141);
                graph.DrawLine(pen, 405, 135, 404, 148);
            }
            if (i == 2 && j == 4)
            {
                graph.DrawLine(pen, 290, 320, 400, 320);
                graph.DrawLine(pen, 400, 320, 386, 314);
                graph.DrawLine(pen, 400, 320, 386, 326);
            }
            if (i == 3 && j == 4)
            {
                graph.DrawLine(pen, 420, 140, 420, 300);
                graph.DrawLine(pen, 420, 300, 414, 286);
                graph.DrawLine(pen, 420, 300, 426, 286);
            }
        }
    }
}
if (Column == 6)
{
    for (int i = 0; i < csvLine.Length; i++)
    {
        string[] rawData = csvLine[i].Split(';');
```

```
    Name.Add(rawData[0]);
}

for (int i = 0; i < csvLine.Length; i++)
{
    string[] rawData = csvLine[i].Split(';');

    double x = Convert.ToDouble(rawData[1]);
    X.Add(x);

    double y = Convert.ToDouble(rawData[2]);
    Y.Add(y);

    double z = Convert.ToDouble(rawData[3]);
    Z.Add(z);

    double k = Convert.ToDouble(rawData[4]);
    K.Add(k);

    double p = Convert.ToDouble(rawData[5]);
    P.Add(p);

    double z1 = Convert.ToDouble(rawData[6]);
    Z1.Add(z1);

};

mass[0, 0] = Correl(X, X);
mass[0, 1] = Correl(X, Y);
mass[0, 2] = Correl(X, Z);
mass[0, 3] = Correl(X, K);
mass[0, 4] = Correl(X, P);
mass[0, 5] = Correl(X, Z1);

mass[1, 0] = Correl(Y, X);
```

```
mass[1, 1] = Correl(Y, Y);
mass[1, 2] = Correl(Y, Z);
mass[1, 3] = Correl(Y, K);
mass[1, 4] = Correl(Y, P);
mass[1, 5] = Correl(Y, Z1);

mass[2, 0] = Correl(Z, X);
mass[2, 1] = Correl(Z, Y);
mass[2, 2] = Correl(Z, Z);
mass[2, 3] = Correl(Z, K);
mass[2, 4] = Correl(Z, P);
mass[2, 5] = Correl(Z, Z1);

mass[3, 0] = Correl(K, X);
mass[3, 1] = Correl(K, Y);
mass[3, 2] = Correl(K, Z);
mass[3, 3] = Correl(K, K);
mass[3, 4] = Correl(K, P);
mass[3, 5] = Correl(K, Z1);

mass[4, 0] = Correl(P, X);
mass[4, 1] = Correl(P, Y);
mass[4, 2] = Correl(P, Z);
mass[4, 3] = Correl(P, K);
mass[4, 4] = Correl(P, P);
mass[4, 5] = Correl(P, Z1);

mass[5, 0] = Correl(Z1, X);
mass[5, 1] = Correl(Z1, Y);
mass[5, 2] = Correl(Z1, Z);
mass[5, 3] = Correl(Z1, K);
mass[5, 4] = Correl(Z1, P);
mass[5, 5] = Correl(Z1, Z1);

for (int i = 0; i < mass.GetLength(0); i++)
```

```
{
    for (int j = 0; j < mass.GetLength(1); j++)
    {
        if (mass[i, j] < 0)
        {
            mass[i, j] = -mass[i, j];
        }
    }
}

for (int i = 0; i < mass.GetLength(0); i++)
{
    for (int j = 0; j < mass.GetLength(0); j++)
    {
        if (mass[i, j] > t)
        {
            mass[i, j] = 1;
        }
        if (j <= i)
        {
            mass[i, j] = 0;
        }
        if (mass[i, j] <= t)
        {
            mass[i, j] = 0;
        }
    }
}

if (DataBank.Connection == "Прямий зв\'язок")
{
    for (int i = 0; i < mass.GetLength(0); i++)
    {
        for (int j = (Column - 1); j >= 0; j--)
        {
```

```
            if (mass[i, j] == 1)
            {
                int p = j - 1;

                while (p >= 0)
                {
                    mass[i, p] = 0;
                    p--;
                }
            }
        }
    }
}

if (DataBank.Connection == "Непрямий зв\'язок")
{
    for (int i = 0; i < mass.GetLength(0); i++)
    {
        for (int j = 0; j < mass.GetLength(1); j++)
        {
            if (mass[i, j] == 1)
            {
                int p = j + 1;

                while (p < Column)
                {
                    mass[i, p] = 0;
                    p++;
                }
            }
        }
    }
}

bmp = new Bitmap(pictureBox1.Width, pictureBox1.Height);
```

```
graph = Graphics.FromImage(bmp);
pen = new Pen(Color.Red, 3f);
Font font = new Font("Arial", 15);

graph.DrawEllipse(pen, 100, 200, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 110, 210);
Counter++;
graph.DrawEllipse(pen, 250, 100, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 260, 110);
Counter++;
graph.DrawEllipse(pen, 250, 300, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 260, 310);
Counter++;
graph.DrawEllipse(pen, 400, 100, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 410, 110);
Counter++;
graph.DrawEllipse(pen, 400, 300, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 410, 310);
Counter++;
graph.DrawEllipse(pen, 550, 200, 40, 40);
graph.DrawString(Name[Counter], font, Brushes.Black, 560, 210);

pen = new Pen(Color.Black, 1f);
for (int i = 0; i < mass.GetLength(0); i++)
{
   for (int j = 0; j < mass.GetLength(0); j++)
   {
      if (mass[i, j] == 1)
      {
         if (i == 0 && j == 1)
         {
            graph.DrawLine(pen, 135, 205, 252, 130);
            graph.DrawLine(pen, 252, 130, 238, 130);
            graph.DrawLine(pen, 252, 130, 247, 143);
         }
```

```
if (i == 0 && j == 2)
{
   graph.DrawLine(pen, 135, 235, 255, 305);
   graph.DrawLine(pen, 255, 305, 241, 305);
   graph.DrawLine(pen, 255, 305, 248, 292);
}
if (i == 0 && j == 3)
{
   graph.DrawLine(pen, 138, 210, 402, 127);
   graph.DrawLine(pen, 402, 127, 388, 123);
   graph.DrawLine(pen, 402, 127, 394, 138);
}
if (i == 0 && j == 4)
{
   graph.DrawLine(pen, 137, 230, 400, 312);
   graph.DrawLine(pen, 400, 312, 391, 302);
   graph.DrawLine(pen, 400, 312, 387, 315);
}
if (i == 0 && j == 5)
{
   graph.DrawLine(pen, 142, 220, 550, 220);
   graph.DrawLine(pen, 550, 220, 536, 214);
   graph.DrawLine(pen, 550, 220, 536, 226);
}
if (i == 1 && j == 2)
{
   graph.DrawLine(pen, 270, 140, 270, 300);
   graph.DrawLine(pen, 270, 300, 264, 286);
   graph.DrawLine(pen, 270, 300, 276, 286);
}
if (i == 1 && j == 3)
{
   graph.DrawLine(pen, 290, 120, 400, 120);
   graph.DrawLine(pen, 400, 120, 386, 114);
   graph.DrawLine(pen, 400, 120, 386, 126);
```

```
}
if (i == 1 && j == 4)
{
   graph.DrawLine(pen, 285, 135, 405, 305);
   graph.DrawLine(pen, 405, 305, 390, 299);
   graph.DrawLine(pen, 405, 305, 403, 289);
}
if (i == 1 && j == 5)
{
   graph.DrawLine(pen, 288, 130, 555, 210);
   graph.DrawLine(pen, 555, 210, 545, 201);
   graph.DrawLine(pen, 555, 210, 542, 214);
}
if (i == 2 && j == 3)
{
   graph.DrawLine(pen, 285, 305, 405, 135);
   graph.DrawLine(pen, 405, 135, 392, 141);
   graph.DrawLine(pen, 405, 135, 404, 148);
}
if (i == 2 && j == 4)
{
   graph.DrawLine(pen, 290, 320, 400, 320);
   graph.DrawLine(pen, 400, 320, 386, 314);
   graph.DrawLine(pen, 400, 320, 386, 326);
}
if (i == 2 && j == 5)
{
   graph.DrawLine(pen, 288, 310, 555, 230);
   graph.DrawLine(pen, 555, 230, 541, 227);
   graph.DrawLine(pen, 555, 230, 545, 240);
}
if (i == 3 && j == 4)
{
   graph.DrawLine(pen, 420, 140, 420, 300);
   graph.DrawLine(pen, 420, 300, 414, 286);
```

```
                    graph.DrawLine(pen, 420, 300, 426, 286);
                }
                if (i == 3 && j == 5)
                {
                    graph.DrawLine(pen, 435, 132, 560, 205);
                    graph.DrawLine(pen, 560, 205, 553, 193);
                    graph.DrawLine(pen, 560, 205, 545, 204);
                }
                if (i == 4 && j == 5)
                {
                    graph.DrawLine(pen, 437, 312, 560, 235);
                    graph.DrawLine(pen, 560, 235, 545, 236);
                    graph.DrawLine(pen, 560, 235, 555, 248);
                }
            }
        }
    }
    pictureBox1.Image = bmp;
  }
 }
}
```

DataBank.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Graph
{
   static class DataBank
   {
      public static string Adress;
      public static string Columns;
      public static string Coefficient;
      public static string Connection;
   }
}
```

ДОДАТОК В

Моделювання причинно-наслідкової структури, враховуючи невимірний фактор впливу

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР71229_21Б 13-1

Аркушів

Київ – 2021

АНОТАЦІЯ

Додаток містить опис програми для моделювання причинно-наслідкових структур, враховуючи невимірний фактор впливу, що виконує поставлені завдання:

- Проведення оцінки зв’язків між параметрами
- Встановлення причинно-наслідкових структур
- Встановлення виду зв’язку між параметрами

Програмний продукт створено мовою програмування C# з використанням технології .NET Framework в середовищі Visual Studio 2019.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку міститься опис основних компонентів програми для моделювання причинно-наслідкових структур, що виконує деякі завдання, визначені в розділі 2. Додаток Б містить програмний код цих компонентів.

Для роботи ПП потрібно мати встановлене середовище Visual Studio.

Програмний продукт створено мовою програмування C# з використанням технології .NET Framework в середовищі Visual Studio 2019.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблена програма надає користувачу наступний функціонал:

- Вибір файлу з даними
- Вибір кількості параметрів
- Вибір оцінки параметрів
- Вибір зв’язку між параметрами

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програма складається з трьох частин: форми з полями, які потрібно заповнити; функції, яка зберігає дані; функції, яка обрахує дані та на основі результатів будує направлений граф.

ЗАВАНТАЖЕННЯ І ЗАПУСК

Для користування програмним продуктом на ПК користувача повинна бути встановлена операційна система Windows 7, Windows 8 або Windows 10. Також потрібно встановити Visual Studio 2019 із розширенням .Net.

Для коректної роботи програмного продукту вхідні дані потрібно помістити в файл з розширення .csv і зберегти на ПК користувача.

Якщо в критерії до ПК користувача підходять для запуску ПП, то для запуска програми потрібно знайти файл Graph.exe і запустити його.

ВХІДНІ ДАНІ

Для коректної роботи програмного продукту вхідні дані потрібно помістити в файл з розширення .csv і зберегти на ПК користувача.

Дані в файлі повинні бути записані послідовно.

ВИХІДНІ ДАНІ

В результаті виконання програми користувач отримає візуалізацію статистичних даних у вигляді направленого графа.