МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ
ФАКУЛЬТЕТ №2
Кафедра інформаційних технологій

РОЗРОБКА ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ ДЛЯ ПЛАНУВАННЯ ПАТРУЛЮВАННЯ ТЕРИТОРІЙ У QGIS

кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Олег МРАЧКОВСЬКИЙ

Науковий керівник:
доцент, кандидат технічних наук
Андрій ДЯКОВ

Рецензент:
_____
_____

Кваліфікаційна робота допущена до захисту
«___» _____ 2026 р., протокол № ____

Завідувач кафедри інформаційних технологій
_____ **Олег ЗАЧЕК**
(підпис)

Львів

АНОТАЦІЯ

МРАЧКОВСЬКИЙ О. Розробка інформаційно-аналітичної системи для планування патрулювання територій у QGIS. – Рукопис.

Дослідження на здобуття освітнього ступеня «бакалавр» за спеціальністю 126 «Інформаційні системи та технології». – Львівський державний університет внутрішніх справ, МВС України, Львів, 2026.

У даній роботі здійснено розробку інформаційно-аналітичної системи для планування патрулювання територій із використанням геоінформаційних технологій. У ході виконання роботи проведено аналіз сучасних підходів до організації патрульної діяльності та застосування геоінформаційних систем у діяльності правоохоронних органів. Розглянуто існуючі програмні рішення у сфері аналізу криміногенної ситуації, зокрема комерційні та відкриті платформи, та визначено їх переваги і недоліки.

У роботі обґрунтовано доцільність використання просторово-часового підходу до аналізу подій, який дозволяє враховувати не лише географічне розташування інцидентів, а й їхню часову актуальність. Запропоновано методику визначення зон підвищеного ризику на основі просторового узагальнення даних та вагового врахування подій залежно від часу їх виникнення.

Практична частина роботи полягає у розробці програмного модуля в середовищі QGIS із використанням мови програмування Python та бібліотеки PyQGIS. Реалізований модуль автоматизує процес обробки просторових даних, включаючи фільтрацію подій, розрахунок показників інтенсивності, визначення зон ризику, їх кластеризацію та побудову маршрутів патрулювання. Особливістю розробленого рішення є інтеграція всіх етапів аналізу в єдиний технологічний процес без необхідності використання сторонніх комерційних систем.

Ключові слова: геоінформаційна система, QGIS, просторовий аналіз, планування патрулювання, маршрутизація.

tory Patrol Planning in QGIS. – Manuscript.

Research for obtaining the bachelor's degree in specialty 126 "Information Systems and Technologies". – Lviv State University of Internal Affairs, Ministry of

ЗМІСТ

ВСТУП

Обґрунтування актуальності теми. Традиційні підходи до планування маршрутів патрулювання часто ґрунтуються на суб’єктивному досвіді або статистиці в розрізі районів, що не дозволяє врахувати точну просторову локалізацію правопорушень. Це призводить до неоптимального розподілу ресурсів та збільшення часу реагування поліції.

Аналіз останніх досліджень і публікацій. Теоретичні засади побудови геоінформаційних систем і баз даних ґрунтовно викладені у працях В. І. Зацерковного, В. Г. Бурачека, О. О. Железняка та А. О. Терещенка [1]. Питання загальної теорії ГІС також досліджувала Л. А. Павленко [3], а фундаментальні принципи їх функціонування деталізовані у зарубіжних джерелах, зокрема у роботах R. A. de By та T. Sutton .

Специфіку застосування геоінформаційних технологій у правоохоронній сфері та міжнародний досвід їх використання для запобігання правопорушенням розкрито у дослідженнях В. Л. Костюка [2]. Окремі аспекти

картування злочинності (Crime Mapping) та просторового аналізу криміногенної ситуації висвітлено у працях S. Chainey.

Практичні аспекти використання відкритого програмного забезпечення QGIS, що є інструментальною основою даної роботи, розглядаються у публікаціях О. Часовського, Ю. Андрейчука та Т. Ямелинця [4].

Метою є розробка програмного модуля та методики просторового аналізу в середовищі QGIS для автоматизованого виявлення зон ризику та побудови оптимальних маршрутів патрулювання.

Для досягнення поставленої мети було визначено наступні **завдання**:

- Проаналізувати методи просторового аналізу для оцінки криміногенної обстановки.
- Обґрунтувати архітектуру рішення на базі QGIS та PostGIS.
- Реалізувати алгоритми кластеризації та оцінки щільності для виявлення зон концентрації подій.

Об’єктом дослідження є процес планування маршрутів патрулювання територій.

Предметом дослідження є методи та інструментальні засоби геоінформаційного аналізу для оптимізації патрулювання у середовищі QGIS.

Методи дослідження.

- геостатистичний метод (Kernel Density Estimation) — для побудови теплових карт та візуалізації щільності подій;
- методи мережевого аналізу (Network Analysis) — для побудови оптимальних маршрутів з урахуванням дорожнього графу;

- об’єктно-орієнтоване програмування (Python/PyQGIS) — для автоматизації процесів обробки даних у середовищі QGIS.

Практичне значення одержаних результатів. Розроблено прикладний інструментарій у середовищі QGIS, який дозволяє аналітикам поліції автоматизувати процес виявлення небезпечних зон та формування маршрутів патрулювання. Запропоноване рішення, на відміну від комерційних аналогів, базується на відкритому програмному забезпеченні та даних OpenStreetMap, що спрощує його впровадження.

Структура та обсяг роботи. Дипломна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 87 сторінок.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ АНАЛІЗУ ТЕРИТОРІАЛЬНОГО ПАТРУЛЮВАННЯ ТА ГЕОІНФОРМАЦІЙНОГО МОДЕЛЮВАННЯ

Патрулювання території як об’єкт просторово-часового аналізу

Патрулювання території у межах цієї роботи розглядається не лише як утворююча діяльність правоохоронних органів, а як формалізований процес просторового розподілу в умовах обмежених ресурсів та нерівномірного просторового розподілу подій. З позиції інформаційної системи патрулювання постає задачею аналітичної обробки історичних даних з метою визначення зон, де концентрація подій є статистично високою, і відповідно направлення туди більшої кількості патрулів[2].

Основою для моделювання є просторово локалізовані події, представлені у вигляді точкових об’єктів із координатами та моментом коли вони трапилися. Саме ці два параметри — місце та момент виникнення правопорушення — дозволяє нам перейти від описового аналізу до кількісної

оцінки ризику, що створює основу для обґрунтованого розподілу патрульних ресурсів. Координати визначають територіальну прив’язку до порушення, тоді як дата та час дають можливість врахувати її актуальність направлення ресурсів[3].

На відміну від традиційного підходу, де кожне порушення розглядається окремо, у межах просторового аналізу важливим є не сам факт інциденту, а його розміщення в близькій частині території. Повторюваність подій у межах обмеженої ділянки свідчить про підвищену концентрацію інцидентів виникнення нових правопорушень, що потребує посиленої присутності патрульних нарядів[7].

Окремого значення набуває часовий аспект. Події, що сталися відносно недавно, мають більший вплив на поточний оперативний стан, ніж ті, що відбулися давно. Тому для коректного створення карти ризику ми мусимо враховувати давність подій та зменшувати їхній внесок у загальну оцінку залежно від часу, що минув з моменту вчинення правопорушення[6].

Таким чином, патрулювання як об’єкт автоматизації може бути представлене у вигляді задачі просторово-часового аналізу, де:

- вхідними даними є точкові події з атрибутами координат та часу;
- аналітичною метою є визначення територій із підвищеною концентрацією порушень;
- результатом є виділення зон ризику, що підлягають пріоритетному патрулюванню.

Зазначене формалізоване представлення створює основу для подальшої реалізації алгоритмів просторового узагальнення та розподілу патрульних можливостей у зонах.

Оцінка рівня ризику на території повинна враховувати як кількість подій, так і їхню актуальність, що потребує використання методів просторового узагальнення та вагового агрегування даних[1].

Методичні підходи до визначення зон концентрації подій

Перехід від аналізу окремих взятих точкових подій до формування усієї картини ризику потребує застосування методів просторового узагальнення. Наявність координат та часових міток дозволяє не лише показати події на карті, але й здійснити їх оцінку з урахуванням просторового розміщення та давності вчинення[3]. У цьому контексті ключовим завданням є визначення територій, де концентрація порушень перевищує середній рівень та потребує більшої у

в Для задачі планування патрулювання важливо забезпечити поєднання к ількох вимог: керованість масштабу аналізу, відтворюваність результату при о днакових параметрах, можливість урахування давності подій та формування и тко окреслених зон ризику. З огляду на зазначене доцільним є використання регулярної сіткової моделі просторового узагальнення[11]. Такий підхід передбачає поділ досліджуваної території на систему однакових комірок ф іксованого розміру, у межах яких здійснюється агрегування інформації про п одії[9].

в У практиці геоінформаційного аналізу для визначення зон підвищеної н небезпеки застосовуються різні методи. Одним із найпоширеніших методів є оцінка ядерної щільності, яка дозволяє побудувати згладжену поверхню р озподілу інтенсивності подій. Такий підхід формує безперервне поле точок, щ о відображає ступінь концентрації порушень у містах. Разом з тим, результат р методу має характер плавної поверхні без чітко визначених меж, що ускладнює в иділення конкретних ділянок для оперативного реагування. Цей метод більше нідходить для тактичного аналізу. Для систематизації переваг і обмежень р озглянутих методів просторового узагальнення доцільно здійснити їх и орівняльний аналіз у таблиці 1.1.[9]

х Таблиця 1.1

Порівняльна характеристика методів просторового узагальнення

о р	**Критерій**			**Гексагональна сітка**

г

а

н

Тип результату	Растрова поверхня	Кластери довільної форми	Регулярні комірки
Чіткість меж	Нечіткі	Залежить від параметрів	Фіксовані
Придатність для маршрутизації	Обмежена	Середня	Висока
Придатність для автоматизації	Середня	Складна	Висока

Джерело: складено автором за матеріалами [2]

У процесі експериментального аналізу було встановлено, що хоча алгоритм DBSCAN добре виділяє шари довільної форми, його параметрична чутливість до радіуса ускладнює стабільне використання в умовах різної кількості інформації. З огляду на необхідність забезпечення відтворюваності та керованості масштабу аналізу для практичної реалізації було обрано метод гексагонального бінінгу, який дозволить чітко формалізувати межі зон небезпеки та забезпечує стабільність результатів при повторному запуску алгоритму[3].

У межах всіх комірок визначається загальний показник, що показує рівень накопиченої небезпеки. На відміну від простого підрахунку кількості порушень, оцінка може враховувати часовий аспект через зменшення ваги подій залежно від часу, що минув з моменту їх вчинення[6]. Такий механізм дозволяє надати більшого значення актуальним інцидентам та зменшити вплив застарілих подій на поточну оцінку оперативної ситуації.

Подальший етап аналізу полягає у відборі частини комірок з найвищими значеннями кількості подій. Використання відносного критерію, наприклад визначення певного відсотка зон із найбільшою кількість порушень, дозволяє адаптувати модель до різних масштабів території та забезпечує гнучкість у прийнятті управлінських рішень[12]. Таким чином формується множина просторово визначених зон підвищеної небезпеки, які можуть бути використані для розподілу наших ресурсів.

Зазначений підхід поєднує просторову структурованість, часову чутливість та можливість параметричного налаштування, що створює підґрунтя для подальшої алгоритмічної реалізації системи підтримки прийняття рішень у сфері патрулювання.

Застосування геоінформаційних систем для аналітичного забезпечення патрулювання

Ефективність планування патрулювання територій безпосередньо залежить від можливості аналізувати події з урахуванням їх просторової локалізації. Традиційні інформаційні системи, що оперують лише табличними даними, дозволяють зберігати відомості про правопорушення, однак не забезпечують інструментів для оцінки їх взаємного розташування, просторової концентрації та доступності. У зв’язку з цим виникає потреба у використанні спеціалізованого програмного середовища, здатного працювати з географічно прив’язаними даними[9].

Геоінформаційна система є комплексним інструментом для збирання, зберігання, обробки та аналізу інформації[3]. Її принципова відмінність від звичайних інформаційних систем полягає у можливості одночасної роботи з атрибутивними та геометричними характеристиками об’єктів. Це дозволяє виконувати просторові запити, визначати належність подій до певних територій, розраховувати відстані, аналізувати щільність розподілу інцидентів та моделювати маршрути пересування[7].

У задачах патрулювання територій ГІС виступає не лише засобом візуалізації, а аналітичною платформою для підтримки прийняття рішень. Просторова інтерпретація даних дозволяє виявляти зони концентрації правопорушень.

Для задач аналітичного забезпечення патрулювання особливе значення мають такі функціональні можливості геоінформаційних систем[11]:

Таблиця 1.2

Функціональні можливості ГІС у задачах планування патрулювання поліції

Функція ГІС	Практичне значення для патрулювання
Просторові запити	Визначення належності подій до конкретних районів або зон відповідальності
Просторове узагальнення	Агрегування подій у межах комірок або кластерів для оцінки рівня ризику
Аналіз щільності	Виявлення зон підвищеної концентрації інцидентів
Мережевий аналіз	Побудова оптимальних маршрутів з урахуванням дорожньої інфраструктури
Візуалізація результатів	Наглядне представлення зон ризику та маршрутів для прийняття рішень

Джерело: складено автором за матеріалами [2]

Застосування цих можливостей у єдиному програмному середовищі забезпечує комплексність аналізу та дозволяє інтегрувати різні дані: інформацію про правопорушення, адміністративні межі, дорожню графіку та об’єкти інфраструктури. Це створює основу для формування об’єктивної оцінки оперативної обстановки на території обслуговування.

Важливою перевагою геоінформаційних систем є можливість автоматизації аналітичних процедур. Повторювані операції — зокрема агрегація подій, визначення зон ризику та побудова маршрутів — можуть бути формалізовані у вигляді алгоритмів, що забезпечує відтворюваність результатів та зменшує вплив суб’єктивного фактору[6]. У контексті планування патрулювання це дозволяє оперативно оновлювати аналітичну модель у разі зміни вхідних даних.

Таким чином, геоінформаційна система у межах даної роботи розглядається як базова аналітична платформа, що забезпечує реалізацію просторово-часового аналізу та створює інструментальні передумови для оптимізації маршрутів патрулювання[5].

Отже, використання геоінформаційної системи створює технічні та аналітичні передумови для застосування спеціалізованих методів просторового аналізу, які будуть розглянуті у наступному підрозділі.

1.4. Методи просторового аналізу у плануванні патрулювання

Просторовий аналіз у діяльності поліції не є самоціллю, а виступає інструментом підтримки прийняття управлінських рішень. Його застосування дозволяє обґрунтовано визначати ділянки підвищеної концентрації подій, оцінювати просторові закономірності їх виникнення та формувати інформаційну основу для оптимізації маршрутів патрулювання[2]. Таким чином, методи просторового аналізу є ключовим елементом переходу від статистичного узагальнення до геоаналітичного моделювання[7].

У сучасних умовах забезпечення публічної безпеки планування патрулювання територій потребує використання формалізованих методів аналізу, що дозволяють перейти від описового сприйняття оперативної ситуації до її кількісної оцінки[3]. Просторовий аналіз виступає інструментом трансформації розрізнених даних про події у структуровану модель ризику, придатну для подальшої оптимізації маршрутів пересування патрульних підрозділів.

На відміну від традиційних підходів, де рішення щодо дислокації нарядів приймаються на основі статистичних звітів або суб’єктивного досвіду, геоінформаційний підхід дозволяє враховувати взаємне розташування подій, їх щільність, просторову концентрацію та часову динаміку. Це створює можливість обґрунтованого розподілу ресурсів відповідно до фактичного рівня криміногенної активності[2].

Формалізація задачі просторового узагальнення

Множину подій можна представити як сукупність точкових об’єктів:

=

де кожна подія характеризується координатами (x, y) та часовою міткою,

Задача просторового аналізу полягає у переході від множини точок до множини зон[1]:

=

кожна з яких має інтегральний показник інтенсивності. ,

Такий перехід потребує застосування методу просторового агрегування.

У межах роботи використано метод регулярної гексагональної сітки. Територія дослідження поділяється на систему рівновеликих шестикутних комірок, у межах яких здійснюється підрахунок подій[6].

Вибір саме гексагональної структури обумовлений такими причинами:
інімізація просторових спотворень порівняно з квадратною сіткою.
івновіддаленість центрів суміжних комірок.
раща апроксимація природних зон впливу.
ідсутність домінування напрямків (на відміну від квадратної моделі).

Для систематизації переваг різних типів регулярних сіток доцільно навести порівняння[6]:

Таблиця 1.3

Порівняння типів регулярних сіткових моделей

Критерій	Квадратна сітка	Гексагональна сітка
Рівновіддаленість центрів	Часткова	Повна
Кількість сусідів	4 або 8	
Просторові спотворення	Вищі	Менші
Придатність для маршрутного аналізу	Середня	Висока
Симетричність моделі	Обмежена	Висока

Джерело: складено автором за матеріалами [6]

Алгоритм визначення зон ризику

Алгоритмічна реалізація просторового аналізу включає такі етапи:
енерація сітки — створення регулярної структури комірок фіксованого розміру.

грегування подій — підрахунок кількості інцидентів у межах кожної комірки.
озрахунок показника інтенсивності:

$$I_i = \Sigma w_j, \quad (1.3.)$$

де w_j — ваговий коефіцієнт події.
асове зважування подій

Для підвищення актуальності аналізу використовується функція затухання[4]:

=

де:)

Δt — час, що минув з моменту події,

λ — коефіцієнт згасання.

Таким чином, нові події мають більший вплив на показник ризику.

Комірки з $I_i \geq I_ih$ визначаються як зони підвищеної концентрації.

Інтеграція з маршрутизацією

Отримані зони трансформуються у центроїди, які виступають орієнтирами для планування маршрутів. Далі задача набуває вигляду оптимізаційної моделі на графі дорожньої мережі[10].

Таким чином, просторовий аналіз виступає підготовчим етапом задачі маршрутизації, забезпечуючи:

- зменшення області пошуку,
- концентрацію уваги на пріоритетних точках,
- підвищення ефективності покриття території.

Висновок до першого розділу

У першому розділі здійснено теоретичний аналіз організації патрулювання територій та можливостей використання геоінформаційних технологій для його аналітичної підтримки. Показано, що традиційні методи до планування маршрутів патрулювання, які ґрунтуються на адміністративному поділі місцевості або суб’єктивному досвіді посадовців, не

дозволяють у повній мірі враховувати просторову концентрацію правопорушень та часову динаміку подій.

Обґрунтовано доцільність розгляду патрулювання як задачі просторово-часового аналізу, у межах якої велика кількість зареєстрованих подій інтерпретується як сукупність геоприв'язаних об'єктів із часовою характеристикою. Такий підхід дає можливість перейти від описового аналізу статистики до формалізованої моделі оцінювання рівня ризику на окремих ділянках території.

У процесі порівняльного аналізу методів просторового узагальнення (KDE, DBSCAN, регулярні сіткові моделі) встановлено, що хоча методи ядерної оцінки щільності та кластеризації ефективні для виявлення зон концентрації подій, їх використання у задачах оперативного планування потребує додаткової формалізації меж та сталі параметри. З огляду на вимоги відтворюваності результатів, керованості масштабу аналізу та подальшої інтеграції з алгоритмами маршрутизації обґрунтовано застосування гексагональної сіткової моделі просторового реагування.

Крім того, важливим результатом теоретичного аналізу є визначення ролі часової складової у процесі планування патрулювання. Встановлено, що інтенсивність правопорушень має виражену часову нерівномірність, яка проявляється у вигляді пікових періодів активності. Урахування цих закономірностей дозволяє не лише точніше визначати зони ризику, але й адаптувати графіки патрулювання відповідно до прогнозованих змін ситуації.

Також обґрунтовано необхідність інтеграції просторово-часового аналізу з методами оптимізації, що забезпечують ефективний розподіл ресурсів. Зокрема, поєднання геоінформаційних технологій із алгоритмами кластеризації та маршрутизації дозволяє формувати узгоджену систему підтримки прийняття рішень. Такий підхід сприяє переходу від реактивного до проактивного управління патрулюванням.

Окремо слід відзначити, що впровадження запропонованих підходів створює передумови для підвищення прозорості та обґрунтованості

управлінських рішень. Використання формалізованих методів аналізу зменшує вплив суб'єктивних факторів та забезпечує можливість відтворення результатів дослідження. Це є важливим кроком у напрямі цифровізації процесів забезпечення громадської безпеки.

Встановлено, що використання геоінформаційних систем створює технічні передумови для автоматизації процесів аналізу, кластеризації та формування зон високого ризику.

Отримані результати теоретичного дослідження формують методологічну основу для системного аналізу та проєктування інформаційно-аналітичної системи планування патрулювання, що розглядається у наступному розділі.

РОЗДІЛ 2. СИСТЕМНИЙ АНАЛІЗ І ПРОЄКТУВАННЯ ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ

2.1. Постановка задачі та визначення вимог до системи

У межах даної роботи розглядається задача автоматизованого формування маршрутів патрулювання території на основі просторово-часового аналізу зареєстрованих подій. На відміну від традиційного підходу, коли маршрути визначаються на основі адміністративного поділу або суб'єктивної оцінки оперативної обстановки, у даній системі рішення приймається на основі формалізованої обробки геоприв'язаних даних[7].

Задача полягає у перетворенні множини точкових подій, що мають координати та часову характеристику, у множину рекомендованих маршрутів патрулювання, які забезпечують охоплення зон підвищеної концентрації інцидентів[6].

Таким чином, система повинна виконати наступне логічне перетворення:

Просторово-часові події → оцінка ризику території → визначення пріоритетних зон → формування маршрутів патрулювання.

Вхідною інформацією для функціонування системи є просторово прив’язані дані про інциденти, представлені у вигляді точкового шару. Кожен об’єкт містить координати, часову характеристику та набір атрибутів, що дозволяють здійснювати фільтрацію за типом події. Окрім цього, користувач задає параметри аналізу, зокрема часовий горизонт, правила зважування подій, розмір комірки просторового узагальнення та кількість патрулів, для яких формується маршрут[8].

Обробка даних у системі

Процес перетворення вхідних даних включає такі етапи:

ільтрація подій за часовим вікном з метою врахування лише актуальних інцидентів[6].

ризначення ваг подіям залежно від їх давності[6].

росторове узагальнення подій шляхом поділу території на регулярну сітку[1].

озрахунок показника інтенсивності подій у межах кожної комірки.

ідбір частини комірок із найвищими значеннями інтенсивності[3].

ормування центроїдів пріоритетних зон[6].

озподіл зон між визначеною кількістю патрулів.

обудова рекомендованих маршрутів обходу зон ризику[8].

Вихідні результати

Результатом функціонування інформаційно-аналітичної системи є формування набору просторових шарів, що відображають аналітичні висновки. Зокрема, у процесі обробки даних генерується шар зон підвищеного ризику, сформований на основі агрегування подій, а також шар їх центроїдів, які можуть розглядатися як потенційні точки дислокації патрульних нарядів. Кінцевим результатом роботи системи є побудовані маршрути патрулювання, оптимізовані з урахуванням просторового розподілу подій та структури дорожньої мережі[9].

На рис. 2.1 подано узагальнену контекстну модель функціонування інформаційно-аналітичної системи. Схема відображає послідовність перетворення вхідних просторово-часових даних у набір аналітичних

результатів, зокрема зон підвищеного ризику та рекомендованих маршрутів патрулювання.

Представлена модель демонструє логічний зв’язок між етапами фільтрації даних, їх просторового узагальнення, оцінювання інтенсивності та подальшої маршрутизації, що дозволяє розглядати систему як цілісний аналітичний механізм підтримки прийняття рішень.

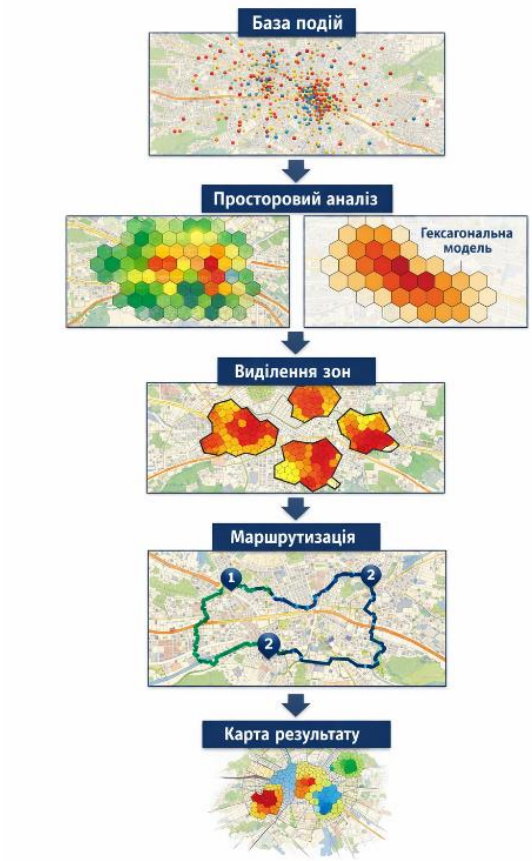

Рис. 2.1. Контекстна схема перетворення даних у системі планування патрулювання

Визначення вимог до системи

Функціональні вимоги до системи визначають перелік операцій, які вона повинна виконувати для забезпечення автоматизованого планування патрулювання. Система має забезпечувати імпорт та обробку точкових геопросторових даних, виконання просторового аналізу з урахуванням заданих параметрів, формування зон підвищеного ризику та автоматизовану побудову маршрутів патрулювання на основі дорожнього графу. Крім цього, повинна бути реалізована можливість зміни параметрів аналізу (розміру комірки,

часових обмежень, порогу концентрації) та забезпечена відтворюваність результатів при повторному запуску алгоритму[6].

Нефункціональні вимоги визначають якісні характеристики розроблюваної системи. Програмний модуль повинен функціонувати у середовищі QGIS без використання комерційного програмного забезпечення та базуватися на відкритих технологіях Важливою вимогою є інтуїтивність інтерфейсу та можливість запуску алгоритмів через графічну оболонку без необхідності ручного редагування коду[4].

2.2. Аналіз існуючих ІТ-рішень у сфері патрулювання

У сучасній практиці правоохоронної діяльності активно застосовуються геоінформаційні системи та спеціалізовані програмні платформи, що забезпечують аналіз криміногенної ситуації, моніторинг подій та підтримку прийняття управлінських рішень. Розвиток цифрових технологій сприяв появі як комерційних професійних рішень, так і муніципальних систем ситуаційного моніторингу. Проте рівень їх відповідності задачам оперативного планування патрулювання є різним[1].

З метою визначення доцільності розроблення власного програмного модуля було проведено аналіз трьох груп рішень: комерційних аналітичних платформ (на прикладі ArcGIS Crime Analysis), муніципальних систем відеомоніторингу (SafeCity) та базових можливостей відкритої ГІС-платформи

Аналіз систем класу SafeCity

Системи типу SafeCity, що впроваджуються у низці міст України, орієнтовані передусім на оперативне реагування та централізований відеомоніторинг. Їх функціональність зосереджена на інтеграції камер спостереження, відображенні інцидентів у реальному часі та підтримці диспетчерських служб.

Основною перевагою таких систем є можливість швидкого виявлення подій та координації реагування. Водночас аналітичні можливості SafeCity обмежені: система не передбачає повноцінного інструментарію для історичного аналізу щільності подій, автоматизованого виявлення зон концентрації правопорушень або математичної оптимізації маршрутів патрулювання на майбутній період[12].

Таким чином, SafeCity виконує функцію оперативного моніторингу, але не забезпечує комплексного аналітичного циклу планування патрулювання. З метою наочного представлення функціональних можливостей розглянутої системи на Рис. 2.2. наведено приклад її інтерфейсу.

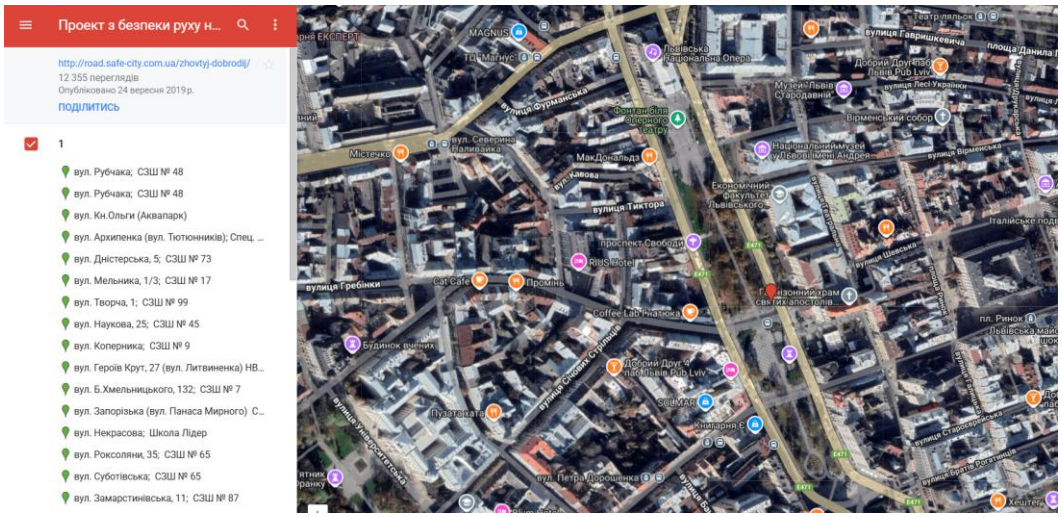

Рис. 2.2. Приклад інтерфейсу системи SafeCity

Аналіз платформи ArcGIS Crime Analysis

Професійні рішення на базі платформи ArcGIS (зокрема модулі Crime Analysis та ArcGIS Tracker) є світовим стандартом у сфері геоаналітики. Вони дозволяють виконувати просторову кластеризацію, аналіз «гарячих точок» (Hot Spot Analysis), прогнозування серійних правопорушень, а також інтегрувати дані з CAD-систем і мобільних додатків[9].

Перевагою ArcGIS є наявність розвинутого аналітичного інструментарію та готових шаблонів для потреб поліції. Система підтримує складні методи статистичного аналізу та мережеву маршрутизацію[4].

Разом з тим, її впровадження пов’язане з істотними фінансовими витратами, оскільки програмне забезпечення має комерційну ліцензію та

потребує додаткових модулів. Закритість вихідного коду обмежує можливості адаптації алгоритмів під специфічні вимоги окремих територіальних підрозділів. Крім того, для повноцінного використання системи необхідна підготовка персоналу та відповідна технічна інфраструктура[5].

Отже, ArcGIS є потужним інструментом стратегічного аналізу, однак його використання на рівні районних підрозділів може бути економічно та організаційно недоцільним. Для ілюстрації особливостей реалізації аналітичних функцій у системі на Рис. 2.3. представлено приклад робочого інтерфейсу.

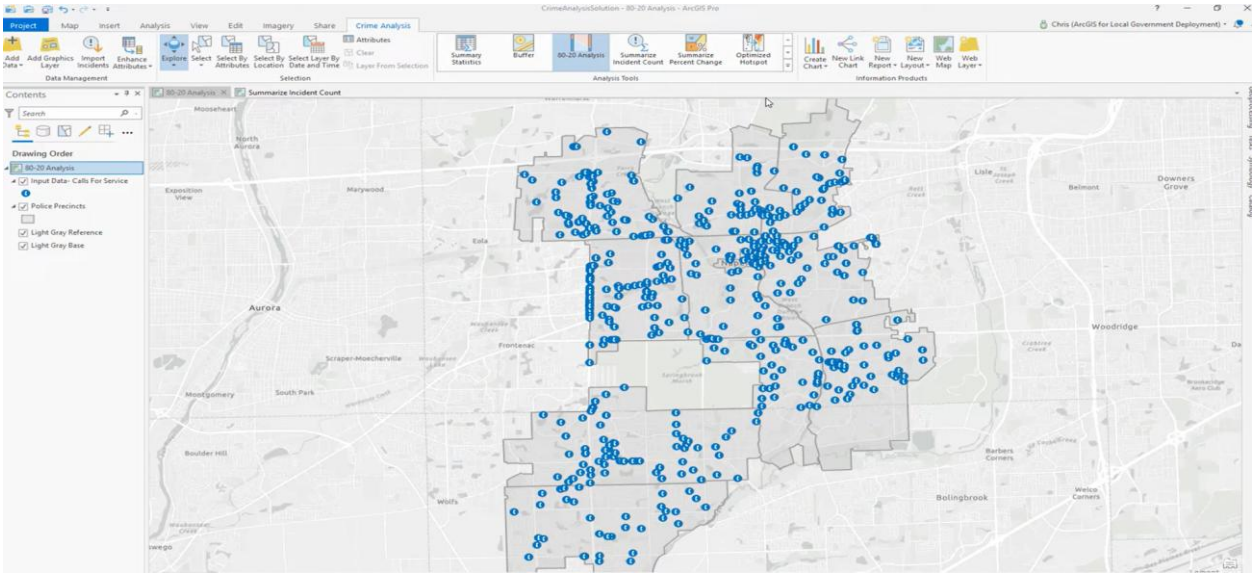

Рис.2.3. Приклад аналізу "гарячих точок" у середовищі ArcGIS

Аналіз можливостей QGIS

QGIS як відкрита геоінформаційна платформа забезпечує широкий набір базових інструментів просторового аналізу. Серед них - побудова теплових карт (Heatmap), виконання кластеризації, використання плагінів для мережевого аналізу (Road Graph) та інтеграція з просторовими базами даних[3].

Перевагами QGIS є безкоштовність, відкритий код та підтримка великої кількості форматів даних. Однак базові можливості системи не інтегровані в єдиний автоматизований технологічний процес. Для формування маршруту патрулювання аналітик змушений виконувати послідовність ручних операцій:

завантаження шару подій, перепроєктування, виконання кластеризації, побудову центроїдів, запуск мережевого аналізу тощо.

Відсутність автоматизації призводить до збільшення часу обробки даних та підвищує ризик помилок. Таким чином, QGIS має достатній інструментарій, проте потребує створення спеціалізованого модуля, що інтегрує всі етапи аналізу в єдину систему[5]. На Рис. 2.4. продемонстровано типовий вигляд інтерфейсу системи, що дозволяє оцінити її функціональну спрямованість та виявити обмеження щодо автоматизації процесів планування патрулювання.

Рис.2.4. Використання плагіна Road Graph у QGIS

Порівняльний аналіз систем.

Для систематизації результатів дослідження доцільно узагальнити ключові характеристики розглянутих систем (табл. 2.1).

Таблиця 2.1

Порівняльна характеристика існуючих рішень

Критерій			QGIS (базовий)	Розроблювана система
Тип ліцензії	Комерційна	Комерційна		

Історичний аналіз подій	Так	Обмежений	Частково	Так
Кластеризація	Так	Ні	Частково	Так
Теплові карти	Так	Ні	Так	Так

Джерело: складено автором за матеріалами [7]

Проведене дослідження показало, що жодне з розглянутих рішень не забезпечує одночасно доступність, відкритість архітектури, автоматизований просторовий аналіз та інтегровану маршрутизацію в межах єдиного технологічного процесу[6].

Комерційні платформи характеризуються потужним аналітичним інструментарієм, але потребують значних фінансових ресурсів та мають обмеження щодо модифікації алгоритмів. Муніципальні системи відеомоніторингу орієнтовані на оперативне реагування і не підтримують глибокий просторовий аналіз. Водночас QGIS, попри відкритість та функціональність, не забезпечує автоматизованого циклу планування патрулювання без додаткової розробкиУ роботі розробляється програмний модуль (алгоритм) для QGIS, який реалізовано у вигляді Processing Algorithm на Python..

Таким чином, доцільним є створення спеціалізованого програмного модуля для QGIS, який інтегрує просторовий аналіз, визначення зон ризику та оптимізовану маршрутизацію в єдину систему підтримки прийняття рішень.

2.3. Архітектура системи

Архітектура розробленого програмного модуля реалізована у вигляді алгоритму обробки просторових даних у середовищі QGIS із використанням фреймворку QGIS Processing. Модуль створено мовою Python як клас, що наслідує QgsProcessingAlgorithm, що дозволяє інтегрувати його безпосередньо

до системи просторового аналізу QGIS та забезпечити виконання повного циклу обробки даних у межах настільного середовища[4].

На відміну від клієнт–серверних архітектур, розроблювана система функціонує локально та не потребує окремої серверної частини. Усі обчислення виконуються засобами QGIS Processing та внутрішніх алгоритмів платформи. Вхідні дані можуть зберігатися у будь-якому підтримуваному QGIS форматі (Shapefile, GeoPackage тощо), що забезпечує гнучкість використання системи у практичних умовах[8].

Загальна архітектурна модель базується на послідовному перетворенні просторових даних, яке реалізується у функції processAlgorithm. Логіка роботи модуля полягає у поетапному виконанні операцій фільтрації, агрегування, кластеризації та маршрутизації, результатом чого є формування аналітичних ш

а Для формалізації структури програмного модуля на Рис. 2.5 подано у узагальнену схему архітектури, що відображає послідовність обробки даних і від моменту завдання параметрів користувачем до формування вихідних к картографічних шарів.

д

л

я

п

і

д

т

р

и

м

к

и

п

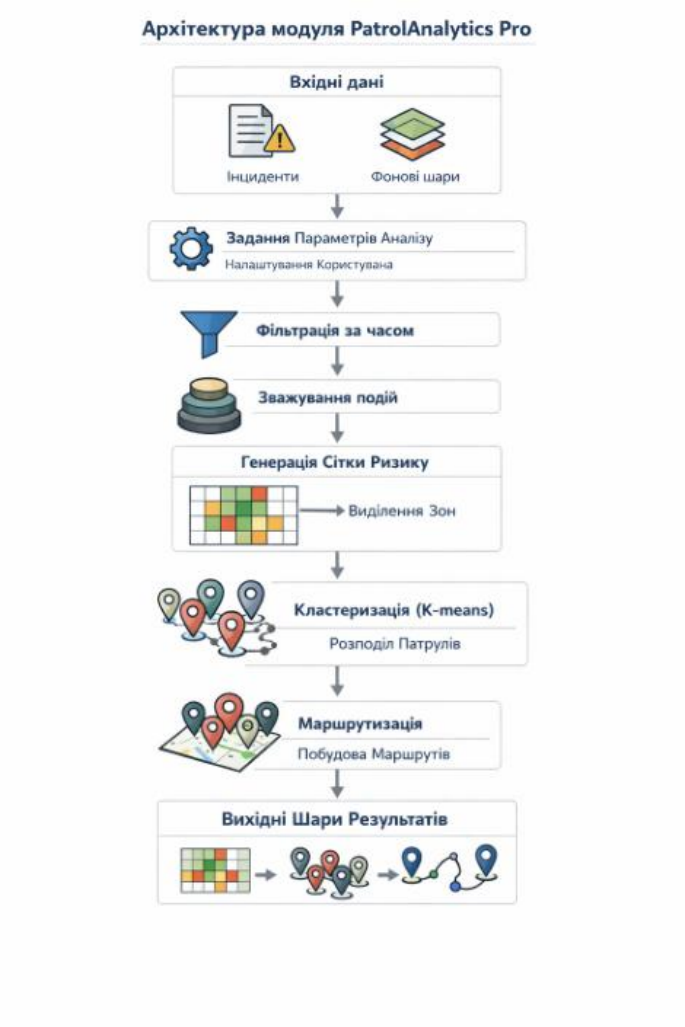

Рис. 2.5. Архітектура модуля PatrolAnalytics

Логіка функціонування модуля

Після запуску алгоритму користувач задає параметри аналізу, зокрема часовий горизонт дослідження, розмір комірки просторової сітки, вагові коефіцієнти для подій різної давності, відсоток відбору зон підвищеного ризику та кількість патрулів. Надалі модуль здійснює фільтрацію вхідних подій відповідно до заданого часового вікна. Це дозволяє враховувати лише актуальні інциденти та виключати застарілі записи[6].

Наступним етапом є врахування давності подій шляхом присвоєння їм вагових коефіцієнтів. Події, що сталися нещодавно, отримують більшу вагу, тоді як події більш давнього періоду мають зменшений вплив на загальну оцінку ризику. Такий підхід дозволяє моделювати поточну оперативну ситуацію з урахуванням часової динаміки[1].

Для просторового узагальнення території формується регулярна прямокутна сітка, що покриває досліджувану область. У межах кожної комірки обчислюється сумарна вага подій, а також показник інтенсивності, який визначається як відношення накопиченої ваги до площі комірки. Отриманий показник використовується для ранжування територій за рівнем ризику.

Після сортування комірок за показником інтенсивності здійснюється відбір верхнього відсотка зон, що характеризуються найбільшою концентрацією подій. Саме ці ділянки розглядаються як пріоритетні для патрулювання. Для кожної з них обчислюється центроїд, який виступає потенційною точкою контролю території.

Розподіл визначених точок між патрулями здійснюється із застосуванням алгоритму K-means, що дозволяє сформувати просторово збалансовані групи об’єктів. Кожній точці присвоюється ідентифікатор патруля, що створює основу для подальшого формування маршрутів.

Події представляються у вигляді точкових об'єктів, зони ризику формуються у вигляді полігонів або комірок просторової сітки, а маршрути патрулювання описуються лінійними об'єктами дорожнього графа[1, 3].

Порядок відвідування точок визначається за евристичним підходом «найближчого сусіда», який полягає у послідовному виборі найближчої невідвіданої точки. Хоча такий метод не гарантує глобальної оптимальності маршруту, він є обчислювально ефективним та придатним для оперативного планування в умовах реального часу.

Формування результатів

У результаті роботи алгоритму створюється набір векторних шарів, що відображають різні аспекти аналітичної обробки: сітка ризику з відібраними зонами підвищеної інтенсивності, центроїди цих зон, рекомендовані точки дислокації патрулів, а також лінійні сегменти маршрутів із зазначенням порядку відвідування. Усі результати формуються як окремі шари у середовищі QGIS та можуть бути збережені або використані для подальшого аналізу[4].

Такий підхід дозволяє інтегрувати методи просторової статистики, кластеризації та евристичної маршрутизації в межах одного програмного модуля без необхідності залучення зовнішніх серверних компонентів. Архітектура модуля є компактною, прозорою та повністю узгодженою з його програмною реалізацією[6].

Таким чином, запропонована архітектура системи передбачає використання стандартного функціоналу геоінформаційної системи QGIS для роботи з просторовими даними та візуалізації результатів, а також авторського програмного модуля, який реалізує алгоритм просторово-часового аналізу. PyQGIS. Він забезпечує автоматизовану обробку подій, обчислення ваг подій залежно від їх актуальності, формування просторової сітки, визначення зон ризику та подальшу генерацію маршрутів патрулювання.

Реалізація такого програмного модуля дозволяє інтегрувати аналітичні методи безпосередньо у робоче середовище QGIS та забезпечує автоматизацію процесу планування патрулювання. Запропонована архітектура створює основу для подальшої програмної реалізації алгоритму, що буде детально розглянуто у третьому розділі дипломної роботи.

2.4. Моделювання процесів планування патрулювання

Процес планування патрулювання у розробленій системі базується на послідовній обробці просторових та часових даних про події. Основою аналізу є точкові дані інцидентів, що містять координати місця події та інформацію про час її виникнення[6]. Такі дані дозволяють виконувати просторово-часовий аналіз та визначати території з підвищеним рівнем ризику виникнення подій. Метою моделювання процесів планування патрулювання є формалізація етапів обробки даних, що забезпечують перетворення початкової інформації про події у результати, придатні для підтримки прийняття рішень щодо розміщення патрулів та побудови маршрутів їхнього руху.

У даній роботі для моделювання процесів використано два типи діаграм: блок-схему алгоритму та UML-діаграму активностей. Використання блок-схеми дозволяє відобразити загальну послідовність обробки даних та основні етапи роботи алгоритму. Такий тип діаграм є зручним для представлення логіки виконання обчислювальних процедур та демонструє, як дані послідовно проходять через різні етапи аналізу. UML-діаграма активностей використовується для відображення взаємодії окремих процесів системи та показує логіку переходів між етапами обробки даних. Поєднання цих двох типів діаграм дозволяє представити як загальну структуру алгоритму, так і детальніше описати процеси взаємодії компонентів системи[7].

Загальна логіка роботи програмного модуля полягає у поетапному перетворенні даних про події у просторові результати, які можуть бути використані для планування патрулювання. На початковому етапі система отримує точкові дані про події, що містять координати та часові атрибути. Подальша обробка даних передбачає фільтрацію подій за часовим інтервалом, призначення ваг подіям залежно від їхньої давності, просторове узагальнення даних, визначення зон підвищеного ризику та формування маршрутів патрулювання. Загальна структура алгоритму роботи системи представлена на рисунку 2.6.

На першому етапі алгоритму виконується часова фільтрація подій. Для кожної події визначається її вік відносно поточного моменту часу. Події, що відбулися поза межами заданого інтервалу, виключаються з подальшого аналізу. Такий підхід дозволяє зосередити аналіз лише на актуальних даних та зменшити вплив застарілої інформації на результати моделювання[6].

Рис. 2.6. Блок-схема алгоритму планування патрулювання

Запропонована модель безпосередньо відповідає структурі алгоритму PatrolAnalytics Pro, реалізованого у середовищі QGIS за допомогою Python та PyQGIS.

Після цього кожній події призначається вага залежно від її давності. Нові події мають більший вплив на формування карти ризику, тоді як більш давні події враховуються з меншою вагою. Такий підхід дозволяє враховувати динаміку змін ситуації та підвищувати значущість останніх подій. У програмному модулі для цього використовується ступінчаста модель зменшення ваги подій залежно від часових інтервалів[4].

Наступним етапом є просторове узагальнення даних. Для цього територія дослідження поділяється на регулярну сітку, де кожна комірка використовується як окрема одиниця аналізу. У межах кожної комірки обчислюється сумарний показник ризику, який формується на основі ваг подій, що потрапляють у відповідну комірку. Такий підхід дозволяє перейти від аналізу окремих точок до оцінювання просторових закономірностей розподілу подій.

На основі отриманих значень ризику виконується визначення зон підвищеного ризику. Для цього комірки сітки впорядковуються за рівнем інтенсивності подій, після чого відбирається певна частка комірок з найвищими значеннями показників. Такі комірки розглядаються як зони підвищеної ймовірності виникнення подій.

Для подальшого аналізу кожна зона ризику представляється у вигляді окремої точки, що відповідає геометричному центру відповідної комірки. Використання центроїдів дозволяє представити зони ризику у спрощеному вигляді та використовувати їх для побудови маршрутів патрулювання.

На наступному етапі точки контролю розподіляються між патрулями. Для цього використовується алгоритм кластеризації, який дозволяє поділити множину точок на кілька груп. Кожна група відповідає зоні відповідальності окремого патруля. Такий підхід забезпечує більш рівномірний розподіл навантаження між патрулями та дозволяє оптимізувати процес патрулювання.

Після визначення зон відповідальності виконується побудова маршрутів патрулювання. Маршрут формується у вигляді послідовності точок контролю, які повинен відвідати патруль. При цьому враховується просторове розташування точок, що дозволяє зменшити загальну довжину маршруту та підвищити ефективність патрулювання.

Процеси взаємодії етапів алгоритму та переходи між ними представлені у вигляді UML-діаграми активностей, яка наведена на рисунку 2.7.

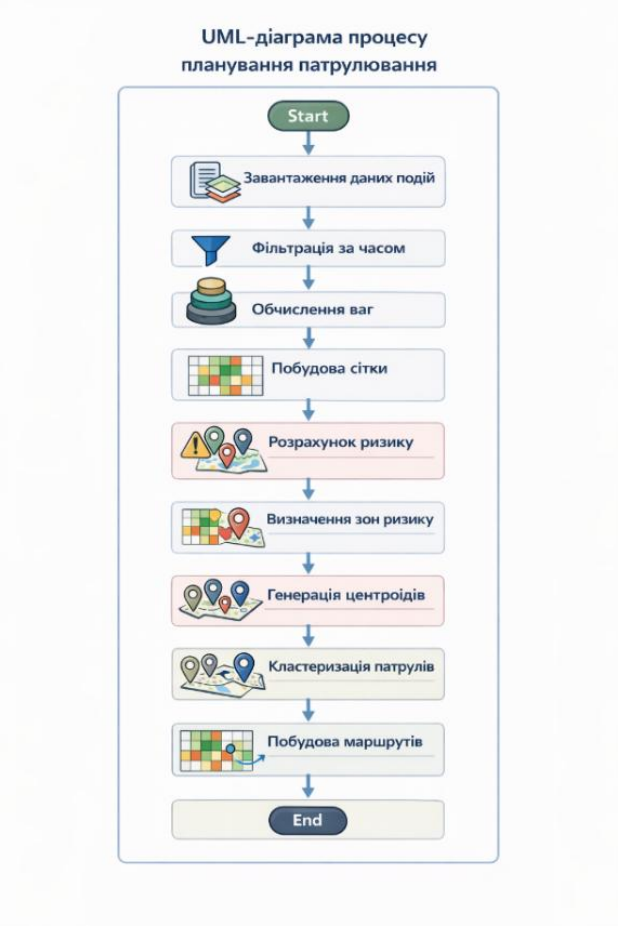

Рис. 2.7. Блок-схема алгоритму планування патрулювання

Таким чином, запропонована модель процесу планування патрулювання дозволяє формалізувати послідовність обробки даних та забезпечує перехід від початкових даних про події до формування маршрутів патрулювання. Реалізація описаних процесів виконана у вигляді програмного модуля на мові Python для середовища QGIS[6]. Фрагмент програмного коду програмного модуля наведено у додатку А.

2.5. Формування технічного завдання та критеріїв ефективності

Розроблення технічного завдання (ТЗ) є невід’ємним етапом створення інформаційно-аналітичної системи, оскільки саме на цьому етапі визначається повний перелік функціональних і нефункціональних вимог, описуються очікувані результати роботи системи та встановлюються критерії оцінювання її ефективності. ТЗ слугує основою для подальшого проєктування, програмної

реалізації, тестування і впровадження. У контексті системи планування патрулювання воно визначає набір можливостей, які повинні забезпечити автоматизоване опрацювання просторових даних, побудову маршрутів та підтримку прийняття рішень на оперативному рівні[1].

Система повинна забезпечувати завантаження просторових даних подій із векторного шару QGIS або бази просторових даних (PostGIS), їх візуалізацію у QGIS, виконання просторового аналізу, створення кластерів, генерацію теплових карт, побудову маршрутів за допомогою алгоритмів мережевого аналізу та формування звітів. Важливою вимогою є можливість інтерактивної роботи з картою, включаючи застосування фільтрів, зміну параметрів алгоритмів і ручне корегування маршруту[3].

Не менш важливим компонентом технічного завдання є нефункціональні вимоги. Вони враховують продуктивність системи, надійність, масштабованість, безпеку та зручність використання. Система повинна коректно працювати з великими наборами просторових даних. Очікується, що обробка даних (кластеризація або побудова теплової карти) буде здійснюватися протягом часу, прийнятного для оперативної роботи - не більше кількох секунд[6].

З точки зору безпеки система має підтримувати контроль доступу, авторизацію користувачів та захищене з’єднання з базою даних. Це особливо важливо у правоохоронній сфері, де передається інформація службового характеру. Зручність використання передбачає інтуїтивний інтерфейс, у якому всі основні операції (завантаження даних, вибір параметрів аналізу, побудова маршруту) виконуються через зрозумілі графічні елементи.

Наступним важливим елементом є визначення критеріїв ефективності системи. Оскільки вона призначена для підтримки діяльності підрозділів поліції, оцінювання повинно враховувати практичні аспекти. Основним критерієм є точність та адекватність зон ризику, визначених на основі просторового аналізу[1]. Система повинна коректно виявляти місця найбільшої концентрації подій, що дозволяє забезпечити ефективніше

використання патрульних нарядів. Іншим критерієм є оптимальність побудованих маршрутів, тобто відповідність маршруту дорожньому графу, мінімальна довжина або час проходження, а також охоплення визначених зон ризику.

Важливо також оцінювати час реакції системи. Умови роботи аналітичних підрозділів вимагають швидкої обробки даних - інструмент повинен виконувати просторовий аналіз і побудову маршруту в межах 1-5 секунд залежно від обсягу даних[4]. Окремим критерієм є стабільність системи при оновленні великих наборів подій або одночасній роботі кількох користувачів. У таблиці 2.1. наведені критерії ефективності роботи інформаційно-аналітичної системи планування патрулювання

Таблиця 2.1

Критерії ефективності роботи системи

Критерій	**Опис**	**Одиниця виміру**	**Очікуване значення**
Точність визначення зон ризику	Відповідність зон, визначених кластеризацією або тепловим аналізом, фактичній концентрації подій	Відсоток відповідності	Не менше 85–90%
Час побудови маршруту	Тривалість виконання алгоритму маршрутизації на дорожньому графі	Секунди	1–3 сек для району, до 5 сек для міста
Час виконання просторового аналізу	Час, необхідний для обробки просторових даних та генерації результатів	Секунди	1–5 сек залежно від обсягу даних

Джерело: складено автором за матеріалами [2]

Нарешті, технічне завдання має визначати очікуваний результат роботи системи. У даному випадку результатом роботи системи є програмний модуль (плагін) для середовища QGIS, який реалізує алгоритми просторового аналізу подій та автоматизованої побудови маршрутів патрулювання., який дозволяє здійснювати просторовий аналіз криміногенної ситуації та формувати маршрути патрулювання, взаємодіючи з базою PostGIS. Крім того, результатом має бути документація: інструкція користувача, опис архітектури, специфікація інтерфейсу та структура бази даних[8].

Висновок до другого розділу

У другому розділі дипломної роботи було виконано системний аналіз та проєктування інформаційно-аналітичної системи для планування патрулювання територій на основі просторових даних у середовищі QGIS.

У процесі проєктування було розроблено архітектуру інформаційно-аналітичної системи та визначено структуру даних, що використовуються у просторовому аналізі. Встановлено основні сутності предметної області, серед яких події (інциденти), зони ризику та маршрути патрулювання, а також визначено взаємозв’язки між ними.

Крім того, у межах розділу було сформовано технічне завдання для розроблення програмного модуля та визначено функціональні й нефункціональні вимоги до системи. Також визначено основні критерії ефективності її роботи, зокрема точність визначення зон ризику.

Таким чином, результати другого розділу сформували теоретичну та методичну основу для подальшої програмної реалізації системи планування патрулювання у середовищі QGIS. Отримані результати використовуються у наступному розділі дипломної роботи, де розглянуто практичну реалізацію програмного модуля та результати його застосування для аналізу просторових даних.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ПЛАНУВАННЯ ПАТУЛЮВАННЯ У QGIS

Реалізація програмного модуля

Практична частина роботи передбачає розробку програмного модуля для просторово-часового аналізу подій та підтримки планування патрулювання. Реалізація виконана у середовищі геоінформаційної системи QGIS з використанням мови програмування Python та бібліотеки PyQGIS[8].

QGIS є відкритою геоінформаційною системою, що забезпечує широкі можливості для роботи з просторовими даними, виконання геообробки, аналізу та візуалізації результатів. Система підтримує розширення функціональності за допомогою програмних модулів, написаних мовою та інтегрувати їх у стандартний інтерфейс QGIS.

Для реалізації розробленого алгоритму було використано програмний інтерфейс PyQGIS, який надає доступ до основних компонентів системи: векторних шарів, геометричних операцій, атрибутивних таблиць, а також інструментів просторової обробки[4]. Використання PyQGIS дозволяє автоматизувати виконання послідовності операцій аналізу, які у стандартному режимі роботи виконуються користувачем вручну.

Програмний модуль реалізовано у вигляді Processing Algorithm — спеціального типу алгоритмів QGIS, що інтегруються в панель *Processing* стандартних інструментів геообробки, задаючи вхідні параметри та отримуючи результати у вигляді нових просторових шарів. Загальна структура програмного модуля та послідовність обробки даних показані на рисунку 3.1.

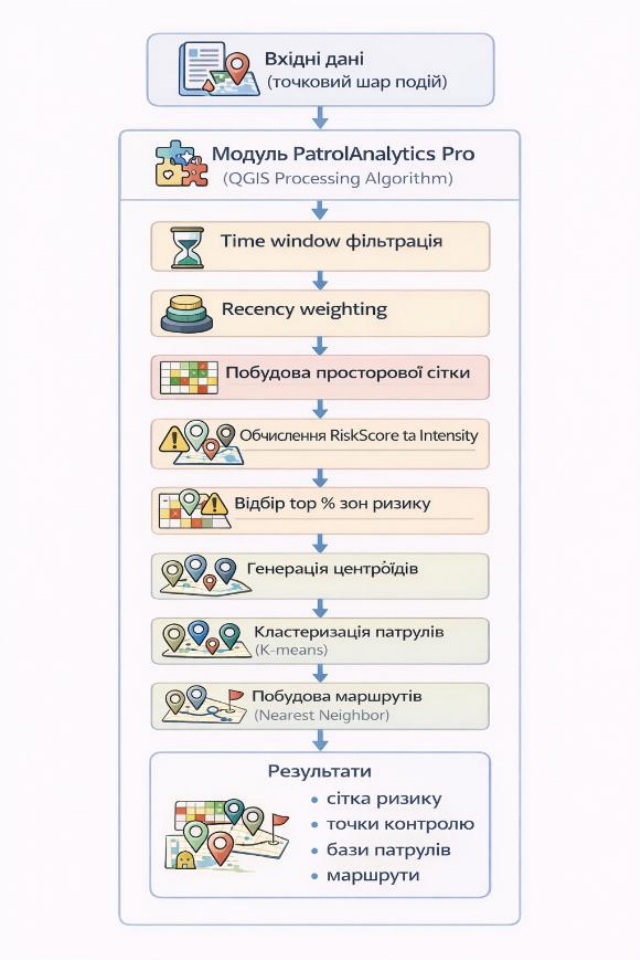

Рис. 3.1. Структурна схема програмного модуля PatrolAnalytics Pro

Вхідними даними для роботи модуля є точковий шар подій (інцидентів), що містить просторові координати та атрибутивну інформацію, зокрема поле дати або часу події. Окрім самого шару подій, користувач задає ряд параметрів, які впливають на результати аналізу. До таких параметрів належать глибина історії подій, параметри ваг за давністю, розмір комірки просторової сітки, частка зон ризику, а також кількість патрулів[9].

Основою програмної реалізації є клас алгоритму, що наслідує базовий клас QgsProcessingAlgorithm. У ньому визначаються основні параметри алгоритму та результуючі шари, які формуються під час виконання аналізу. Фрагмент програмного коду наведено у рисунку 3.2.

```
class PatrolAnalyticsProFull(QgsProcessingAlgorithm):
    # -------- Parameters --------
    INPUT = "INPUT"
    GRID_SIZE = "GRID_SIZE"

    DATE_FIELD = "DATE_FIELD"
    DAYS_BACK = "DAYS_BACK"

    W1_DAYS = "W1_DAYS"
    W1_WEIGHT = "W1_WEIGHT"
    W2_DAYS = "W2_DAYS"
    W2_WEIGHT = "W2_WEIGHT"
    W3_DAYS = "W3_DAYS"
    W3_WEIGHT = "W3_WEIGHT"

    TOP_PERCENT = "TOP_PERCENT"

    K_PATROLS = "K_PATROLS"
    MAX_POINTS_PER_PATROL = "MAX_POINTS_PER_PATROL"
    SCORE_FIELD = "SCORE_FIELD"  # поле пріоритету для патрулювання (звично intensity)

    # -------- Outputs --------
    OUTPUT_RISK_GRID = "OUTPUT_RISK_GRID"
    OUTPUT_RISK_CENTROIDS = "OUTPUT_RISK_CENTROIDS"
    OUTPUT_PATROL_BASES = "OUTPUT_PATROL_BASES"
    OUTPUT_PATROL_POINTS = "OUTPUT_PATROL_POINTS"
    OUTPUT_PATROL_ROUTES = "OUTPUT_PATROL_ROUTES"
```

Рис. 3.2. Фрагмент програмного коду алгоритму PatrolAnalytics Pro a саме

к

У процесі виконання алгоритму формується декілька результуючих просторових шарів. До них належать: а

- сітка ризику, що відображає території з підвищеною концентрацією подій;
- центроїди ризикових осередків, які використовуються як точки контролю;
- рекомендовані базові позиції патрулів;

Таким чином, програмний модуль автоматизує повний цикл обробки даних — від аналізу історичних подій до формування рекомендацій щодо розміщення патрулів та побудови маршрутів патрулювання[5].

Повний програмний код реалізованого модуля наведено у додатку А. Розроблений модуль отримав назву PatrolAnalytics Pro та призначений для виконання просторово-часового аналізу подій у середовищі QGIS.

Логіка роботи алгоритму та основні етапи обробки даних розглядаються у наступному підрозділі.

Алгоритм просторово-часового аналізу

Розроблений програмний модуль PatrolAnalytics Pro реалізує алгоритм просторово-часового аналізу подій, спрямований на виявлення зон підвищеного ризику та формування рекомендацій щодо планування патрулювання. Основною ідеєю алгоритму є використання історичних даних про події для визначення територій з найбільшою концентрацією інцидентів та подальшого розподілу цих зон між патрульними підрозділами.

Алгоритм обробки даних складається з послідовності взаємопов’язаних етапів, які виконуються у середовищі QGIS у межах розробленого Processing Algorithm. На кожному етапі виконується певна операція над просторовими даними, що дозволяє поступово переходити від сирих даних подій до формування маршрутів патрулювання[6].

У програмній реалізації для просторового узагальнення використано регулярну прямокутну сітку, що обумовлено доступністю стандартних інструментів QGIS Processing та простотою інтеграції з подальшими етапами обчислення ризику і ранжування комірок.

Загальна логіка роботи алгоритму представлена на рисунку 3.3.

Рис. 3.3. Загальна схема алгоритму просторово-часового аналізу

Робота запропонованого алгоритму просторово-часового аналізу визначається набором параметрів, які задаються користувачем під час запуску програмного модуля. Ці параметри визначають часовий інтервал аналізу подій, правила обчислення ваг подій залежно від їх давності, просторову деталізацію аналізу, а також параметри формування зон ризику та розподілу патрулів. Основні параметри алгоритму наведено у таблиці 3.1.

Таблиця 3.1.

Основні параметри алгоритму

Параметр	**Опис**
	часовий інтервал аналізу
	розмір комірки сітки
	пороги давності подій
	ваги подій
	частка комірок ризику
	кількість патрулів

Джерело: складено автором за матеріалами [2]

Значення зазначених параметрів можуть змінюватися залежно від характеру досліджуваних даних та поставлених аналітичних завдань, що забезпечує гнучкість використання розробленого програмного модуля.

Етап А. Часова фільтрація подій (Time Window)

На першому етапі алгоритму виконується відбір подій за часовим критерієм. Для кожного запису вхідного шару подій обчислюється різниця між поточною датою та датою виникнення події. Таким чином визначається «вік» події у днях.

До подальшого аналізу включаються лише ті події, для яких дата події задана та знаходиться у межах визначеного часового інтервалу:

$0 \leq age \leq DAYS\_BACK$.

Цей етап дозволяє враховувати лише актуальні дані та виключати застарілі події, які можуть спотворювати результати аналізу.

Фрагмент програмної реалізації обчислення віку події наведено на рисунку 3.4.

```
# --------- (A) Time window filter ---------
# days_expr: кількість днів тому (ціле)
days_expr = f"to_int( age(now(), \"{date_field}\") / make_interval(days:=1) )"
filter_expr = f"\"{date_field}\" IS NOT NULL AND {days_expr} >= 0 AND {days_expr} <= {days_back}"

filtered_pts = processing.run(
    "native:extractbyexpression",
    {"INPUT": parameters[self.INPUT], "EXPRESSION": filter_expr, "OUTPUT": "TEMPORARY_OUTPUT"},
    context=context, feedback=feedback, is_child_algorithm=True
)["OUTPUT"]
filtered_pts = self._as_layer(filtered_pts, context)
```

Рис. 3.4. Фрагмент програмної реалізації часової фільтрації подій

Етап B. Призначення ваг подіям (Recency Weighting)

Після відбору подій виконується оцінювання їх важливості залежно від часу виникнення. Ідея полягає у тому, що більш нові події повинні мати більший вплив на формування зон ризику.

Для цього кожній події призначається вага w, яка залежить від її давності. Якщо подія сталася нещодавно, їй присвоюється більша вага, тоді як події, що відбулися значно раніше, мають менший вплив на результати аналізу.

Використання ваг дозволяє враховувати динаміку зміни криміногенної ситуації та надавати більший пріоритет актуальним подіям. Фрагмент програмної реалізації призначення ваг подіям залежно від давності наведено на рисунку 3.5.

```
# --------- (B) Add recency weight field w ---------
weight_expr = (
    f"CASE "
    f"WHEN {days_expr} <= {w1_days} THEN {w1} "
    f"WHEN {days_expr} <= {w2_days} THEN {w2} "
    f"WHEN {days_expr} <= {w3_days} THEN {w3} "
    f"ELSE 0 "
    f"END"
)

weighted_pts = processing.run(
    "native:fieldcalculator",
    {
        "INPUT": filtered_pts,
        "FIELD_NAME": "w",
        "FIELD_TYPE": 0,  # Float
        "FIELD_LENGTH": 10,
        "FIELD_PRECISION": 3,
        "FORMULA": weight_expr,
        "OUTPUT": "TEMPORARY_OUTPUT",
    },
    context=context, feedback=feedback, is_child_algorithm=True
)["OUTPUT"]
weighted_pts = self._as_layer(weighted_pts, context)
```

Рис. 3.5. Фрагмент програмної реалізації призначення ваг за давністю подій

Етап C. Побудова просторової сітки

На наступному етапі над досліджуваною територією формується регулярна прямокутна сітка з кроком GRID_SIZE. Кожна комірка сітки виступає окремою одиницею просторового аналізу.

Використання регулярної сітки дозволяє перейти від аналізу окремих точок до узагальненого аналізу території. У межах кожної комірки буде обчислюватися сумарний показник ризику.

Розмір комірки визначається у параметрах алгоритму та вимірюється у координатних одиницях обраної системи координат.

Етап D. Обчислення показника ризику

Для кожної комірки сітки виконується підрахунок сумарної ваги подій, що потрапляють у її межі. Отриманий показник називається RiskScore.

$$RiskScore = \Sigma w$$

де

w – вага події.

Крім сумарного ризику обчислюється показник інтенсивності подій, який враховує площу комірки:

Використання показника інтенсивності дозволяє коректно порівнювати комірки різного розміру.

Фрагмент програмної реалізації обчислення показника ризику наведено на рисунку 3.6.

```
# --------- (C) Build grid ---------
grid_layer = processing.run(
    "native:creategrid",
    {
        "TYPE": 2,  # rectangle polygons
        "EXTENT": extent,
        "HSPACING": grid_size,
        "VSPACING": grid_size,
        "HOVERLAY": 0.0,
        "VOVERLAY": 0.0,
        "CRS": crs,
        "OUTPUT": "TEMPORARY_OUTPUT",
    },
    context=context, feedback=feedback, is_child_algorithm=True
)["OUTPUT"]
grid_layer = self._as_layer(grid_layer, context)

# --------- (D) Aggregate weights in polygons (RiskScore) ---------
# Creates field evt_w_sum as sum(w)
joined = processing.run(
 "native:joinbylocationsummary",
    {
        "INPUT": grid_layer,
        "JOIN": weighted_pts,
        "PREDICATE": [0],  # intersects
        "JOIN_FIELDS": ["w"],
        "SUMMARIES": [5],  # sum
        "DISCARD_NONMATCHING": False,
        "PREFIX": "evt_",
        "OUTPUT": "TEMPORARY_OUTPUT",
    },
    context=context, feedback=feedback, is_child_algorithm=True
)["OUTPUT"]
joined = self._as_layer(joined, context)
```

Рис. 3.6. Фрагмент програмної реалізації побудови сітки ризику та обчислення показників RiskScore й Intensity

Етап E. Визначення зон підвищеного ризику

Після обчислення показників ризику всі комірки сітки сортуються за значенням показника Intensity. З отриманого набору відбирається певна частка комірок з найбільшими значеннями показника.

Ця частка задається параметром TOP_PERCENT і визначає відсоток території, що буде класифіковано як зона підвищеного ризику.

Такий підхід дозволяє використовувати відносний критерій відбору замість фіксованого порогу, що забезпечує стабільність результатів при зміні кількості подій.

Запропонований алгоритм є параметрично керованим, оскільки результати його роботи визначаються заданими користувачем значеннями часового вікна, ваг подій, розміру просторової сітки, частки зон ризику та кількості патрулів. За незмінних вхідних даних і однакових параметрів алгоритм забезпечує повторюваність результатів, що є важливою вимогою до інформаційно-аналітичних систем підтримки прийняття рішень.

Етап F. Генерація точок контролю

Для кожної комірки, що віднесена до зони підвищеного ризику, обчислюється її центроїд. Отримані точки використовуються як представники відповідних ризикових осередків.

Центроїди виконують роль потенційних точок контролю, які мають бути включені до маршруту патрулювання. Фрагмент програмної реалізації відбору зон підвищеного ризику та генерації їх центроїдів наведено на рисунку 3.7.

```
# --------- (E) Select top X% cells by intensity ---------
sorted_layer = processing.run(
    "native:orderbyexpression",
    {"INPUT": with_intensity, "EXPRESSION": "\"intensity\"", "ASCENDING": False, "NULLS_FIRST": False, "OUTPUT": "TEMPORARY_OUTPUT"},
    context=context, feedback=feedback, is_child_algorithm=True
)["OUTPUT"]
sorted_layer = self._as_layer(sorted_layer, context)

ranked_layer = processing.run(
    "native:addautoincrementalfield",
    {
        "INPUT": sorted_layer,
        "FIELD_NAME": "rank",
        "START": 1,
        "GROUP_FIELDS": [],
        "SORT_EXPRESSION": "",
        "SORT_ASCENDING": True,
        "SORT_NULLS_FIRST": False,
        "OUTPUT": "TEMPORARY_OUTPUT",
    },
    context=context, feedback=feedback, is_child_algorithm=True
)["OUTPUT"]
ranked_layer = self._as_layer(ranked_layer, context)

n_cells = ranked_layer.featureCount()
if n_cells == 0:
    raise QgsProcessingException("Сітка не містить об'єктів. Перевірте екстент/дані.")

k_top = int((n_cells * top_percent + 99) // 100)  # ceil
if k_top < 1:
    k_top = 1

feedback.pushInfo(f"Cells: {n_cells}. Top {top_percent}% => {k_top} cells.")

top_grid = processing.run(
    "native:extractbyexpression",
    {"INPUT": ranked_layer, "EXPRESSION": f"\"rank\" <= {k_top}", "OUTPUT": "TEMPORARY_OUTPUT"},
    context=context, feedback=feedback, is_child_algorithm=True
)["OUTPUT"]
top_grid = self._as_layer(top_grid, context)

# --------- (F) Centroids of risk cells ---------
risk_centroids = processing.run(
    "native:centroids",
    {"INPUT": top_grid, "ALL_PARTS": False, "OUTPUT": "TEMPORARY_OUTPUT"},
    context=context, feedback=feedback, is_child_algorithm=True
)["OUTPUT"]
risk_centroids = self._as_layer(risk_centroids, context)
```

Рис. 3.7. Фрагмент програмної реалізації відбору top зон ризику та генерації центроїдів

Етап G. Розподіл зон між патрулями

Для розподілу точок контролю між патрулями використовується метод кластеризації K-means. Алгоритм групує точки таким чином, щоб кожен кластер містив географічно близькі точки.

Кількість кластерів визначається параметром K_PATROLS і відповідає кількості патрульних нарядів.

У результаті виконання кластеризації кожній точці призначається ідентифікатор patrol_id, який визначає, до якого патруля вона належить. Фрагмент програмної реалізації розподілу точок контролю між патрулями методом K-means наведено на рисунку 3.8.

```
# --------- (G) Deployment: K-means on centroids ---------
# Якщо точок менше, ніж K, зменшити K
n_pts = risk_centroids.featureCount()
if n_pts == 0:
    # Повертаємо тільки ризикову сітку/центроїди (порожні виходи для патрулів)
    feedback.pushInfo("Ризикових центроїдів немає. Патрулювання не сформовано.")
    # Створимо порожні sinks для патрульних шарів, щоб алгоритм не падав
    self._create_empty_patrol_outputs(parameters, context, crs)
    return {
        self.OUTPUT_RISK_GRID: grid_sink_id,
        self.OUTPUT_RISK_CENTROIDS: cent_sink_id,
        self.OUTPUT_PATROL_BASES: self._empty_base_id,
        self.OUTPUT_PATROL_POINTS: self._empty_points_id,
        self.OUTPUT_PATROL_ROUTES: self._empty_routes_id,
    }

if k_patrols > n_pts:
    feedback.pushInfo(f"K={k_patrols} > N_points={n_pts}. Зменшую K до {n_pts}.")
    k_patrols = n_pts

clustered = processing.run(
    "native:kmeansclustering",
    {
        "INPUT": risk_centroids,
        "CLUSTERS": k_patrols,
        "FIELD_NAME": "patrol_id",
        "SIZE_FIELD_NAME": "cluster_sz",
        "OUTPUT": "TEMPORARY_OUTPUT",
    },
    context=context, feedback=feedback, is_child_algorithm=True
)["OUTPUT"]
clustered = self._as_layer(clustered, context)

clustered_fields = clustered.fields()
```

Рис. 3.8. Фрагмент програмної реалізації кластеризації точок контролю між патрулями

Етап H. Формування баз патрулів

Для кожного отриманого кластера обчислюється середнє значення координат точок, що входять до нього. Отримана точка використовується як рекомендована позиція бази патруля.

Крім координат бази обчислюються додаткові показники:

- кількість точок контролю у кластері;
- сумарний показник ризику;
- середній показник ризику.

Ці показники дозволяють оцінити навантаження на кожен патруль.

Етап I. Побудова маршрутів патрулювання

На завершальному етапі формується порядок відвідування точок контролю в межах кожного патруля. Для цього використовується евристичний алгоритм найближчого сусіда (Nearest Neighbor).

Алгоритм починає побудову маршруту з базової точки патруля та на кожному кроці обирає найближчу ще не відвідану точку. Процес повторюється доти, доки всі точки не будуть включені до маршруту.

Сформований маршрут представляється у вигляді лінійного шару, що містить послідовні сегменти між точками патрулювання.

Слід зазначити, що використаний алгоритм є евристичним і не гарантує глобально оптимального маршруту, проте забезпечує швидке формування практично придатного плану обходу. Фрагмент програмної реалізації формування баз патрулів та побудови маршруту за евристикою найближчого сусіда наведено на рисунку 3.9.

```
# --------- (I) Patrol points output: original centroid fields + seq ---------
out_point_fields = QgsFields()
for fld in clustered_fields:
    out_point_fields.append(fld)
out_point_fields.append(QgsField("seq", QVariant.Int))

(pt_sink, pt_id) = self.parameterAsSink(
    parameters, self.OUTPUT_PATROL_POINTS, context,
    out_point_fields, QgsWkbTypes.Point, clustered.sourceCrs()
)
if pt_sink is None:
    raise QgsProcessingException("Не вдалося створити OUTPUT_PATROL_POINTS.")
```

Рис. 3.9. Фрагмент програмної реалізації формування баз патрулів і побудови маршруту

Таким чином, алгоритм просторово-часового аналізу дозволяє автоматично визначати зони підвищеного ризику на основі історичних подій, формувати точки контролю та генерувати рекомендовані маршрути патрулювання.

Вихідні результати алгоритму

У результаті виконання алгоритму формується набір просторових шарів, що використовуються для підтримки прийняття рішень щодо планування патрулювання:

- сітка ризику території;
- точки ризикових осередків;
- рекомендовані бази патрулів;
- точки патрулювання;
- маршрути патрулювання.

Отримані результати можуть бути використані для подальшої візуалізації та аналізу у середовищі QGIS.

Реалізація алгоритму просторово-часового аналізу та формування маршрутів патрулювання

Для перевірки працездатності розробленого алгоритму було виконано його практичне застосування у середовищі геоінформаційної системи QGIS. Метою експерименту є демонстрація можливостей програмного модуля щодо аналізу просторово-часових даних та формування рекомендацій для планування патрулювання території.

У якості вхідних даних використовувався точковий шар подій, що містить інформацію про координати інцидентів та дату їх виникнення. Кожен об’єкт шару має атрибут, що відповідає часовій характеристиці події, яка використовується для подальшого аналізу[3].

На першому етапі було підготовлено вхідні дані та завантажено їх у середовище QGIS. Дані відображаються на карті у вигляді точок, що відповідають місцям виникнення подій.

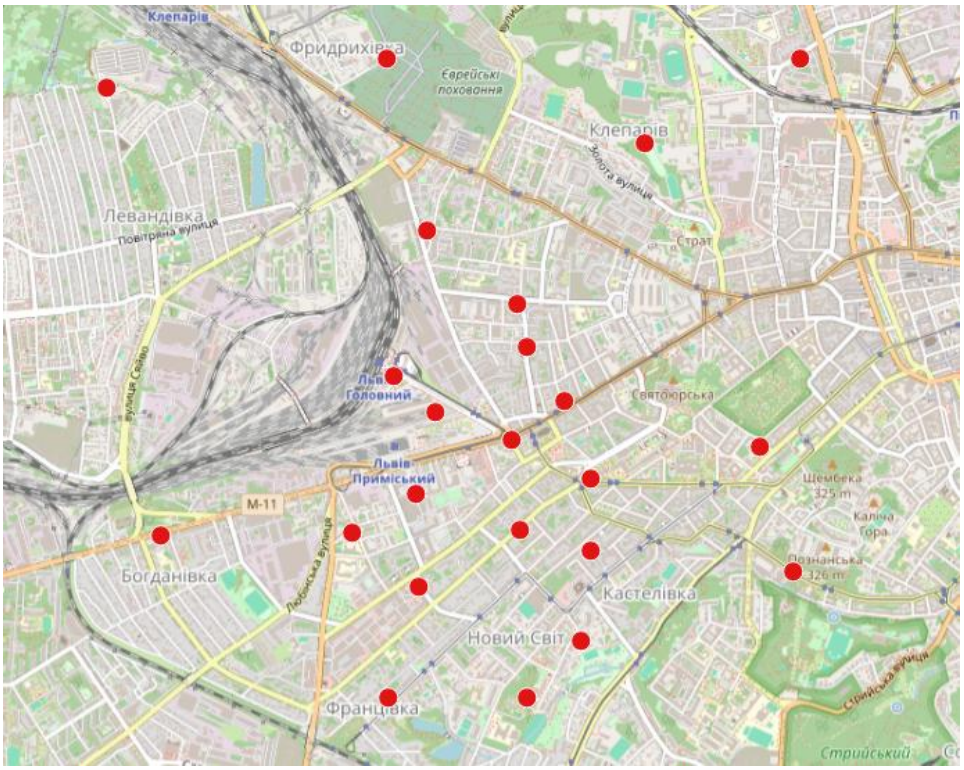

Рис. 3.10. Відображення вхідних даних у середовищі QGIS. (Червоні точки – наші правопорушення)

Після підготовки даних було запущено розроблений алгоритм через інструмент Processing Toolbox. Користувач має можливість задати основні параметри роботи алгоритму, зокрема:

- розмір просторової сітки;
- часовий інтервал аналізу;
- поле, що містить дату події;
- кількість патрулів.

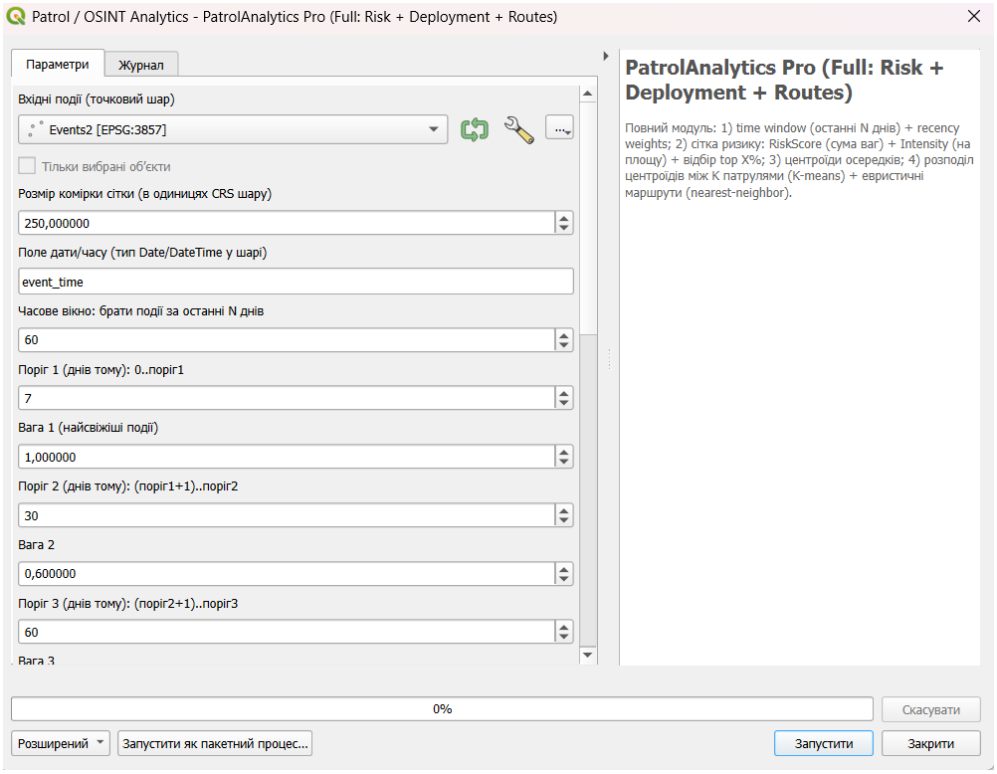

Рис. 3.11. Вікно параметрів алгоритму PatrolAnalytics Pro

Після запуску алгоритму виконується обробка вхідних даних та формування просторової сітки, що використовується для оцінки рівня ризику на різних ділянках території. Для кожної комірки сітки обчислюється інтегральний показник ризику на основі кількості подій та їх часової актуальності.

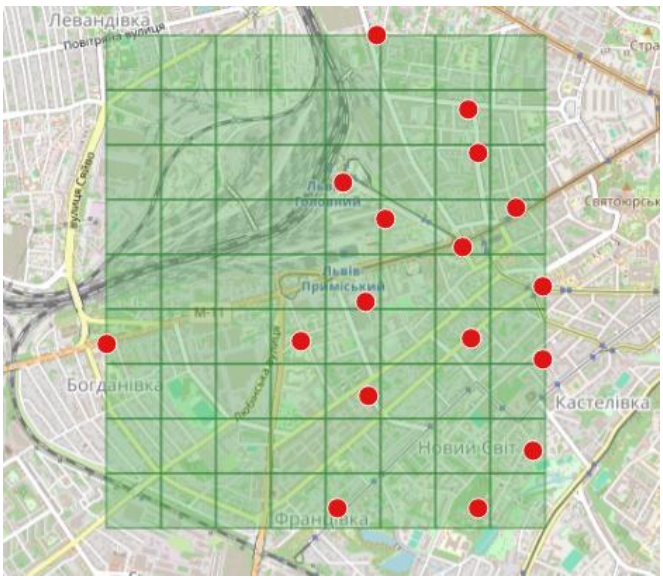

Рис 3.12. Результат формування сітки ризику

На основі отриманих значень ризику визначаються найбільш проблемні ділянки території. Для цих ділянок алгоритм обчислює центри концентрації подій, які можуть використовуватись як точки базування патрульних екіпажів.

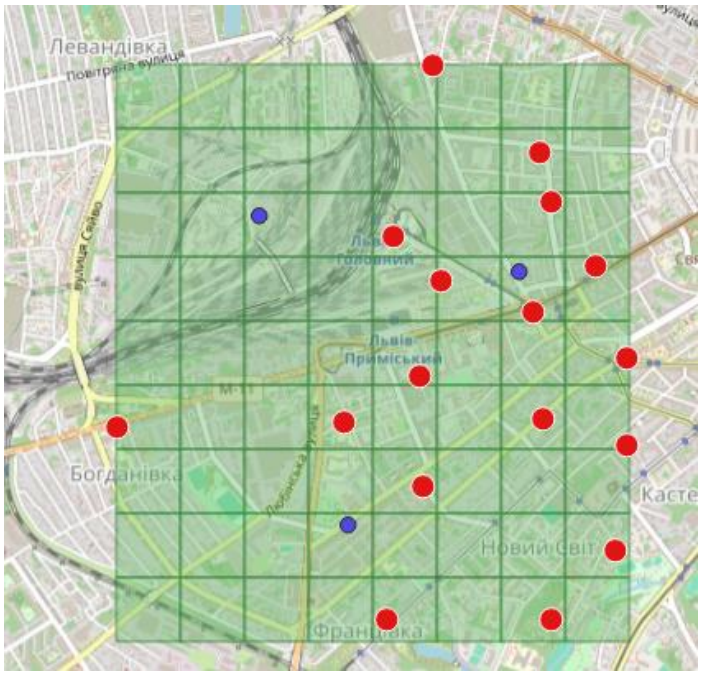

Рис 3.13. Визначені точки базування патрулів(Сині точки)

Наступним етапом роботи алгоритму є побудова маршрутів патрулювання. Маршрути формуються на основі просторового розташування точок ризику та оптимізуються таким чином, щоб забезпечити ефективне охоплення території патрулями.

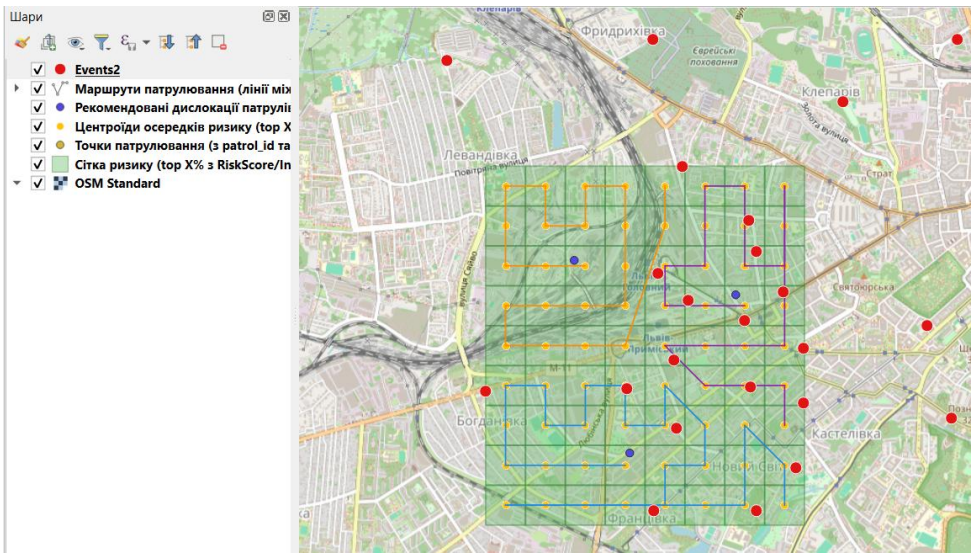

Рис. 3.14. Побудовані маршрути патрулювання

Отримані результати демонструють можливість використання розробленого програмного модуля для підтримки прийняття рішень щодо планування патрулювання. Використання просторово-часового аналізу дозволяє визначити найбільш проблемні зони та оптимізувати розподіл патрульних ресурсів.

Висновки до третього розділу

У третьому розділі було реалізовано програмний модуль просторово-часового аналізу для виявлення зон підвищеного ризику та формування маршрутів патрулювання. Реалізація алгоритму виконана у середовищі геоінформаційної системи QGIS з використанням мови програмування Python та інструментів Processing Framework.

Розроблений алгоритм здійснює комплексну обробку просторово-часових даних та складається з декількох послідовних етапів: часової фільтрації подій, призначення ваг залежно від давності подій, побудови регулярної просторової сітки, обчислення показників ризику, визначення зон підвищеної інтенсивності подій, генерації точок контролю, кластеризації зон ризику між патрулями та формування маршрутів патрулювання.

Для розподілу зон ризику між патрулями використано метод кластеризації K-means, що дозволяє ефективно групувати просторово близькі точки. Формування маршрутів патрулювання здійснюється за допомогою

евристичного алгоритму найближчого сусіда, який забезпечує швидке побудування практично придатних маршрутів обходу.

Крім того, важливим аспектом реалізації програмного модуля є його гнучкість та адаптивність до різних типів вхідних даних. Алгоритм дозволяє враховувати як історичні, так і оперативні дані, що забезпечує актуальність результатів аналізу. Завдяки використанню параметризованих налаштувань користувач може змінювати часові інтервали, вагові коефіцієнти та розмір просторової сітки відповідно до специфіки досліджуваної території або задачі.

Окрему увагу приділено оптимізації обчислювальних процесів, що дозволяє застосовувати розроблений модуль навіть для великих обсягів геопросторових даних. Використання інструментів Processing Framework у середовищі QGIS забезпечує ефективну обробку даних та інтеграцію з іншими геоінформаційними інструментами. Це сприяє підвищенню продуктивності та зменшенню часу виконання аналізу, що є критично важливим у задачах оперативного реагування.

Перспективи подальшого розвитку запропонованого підходу полягають у впровадженні більш складних методів аналізу, зокрема використанні машинного навчання для прогнозування зон ризику. Також можливим є розширення функціональності модуля шляхом інтеграції з системами моніторингу в реальному часі та мобільними додатками для патрульних підрозділів. Це дозволить не лише підвищити точність прогнозів, але й забезпечити більш ефективне управління ресурсами в динамічному середовищі.

Розроблений програмний модуль дозволяє автоматизувати процес аналізу просторово-часових даних та підтримувати прийняття рішень щодо оптимального розподілу патрульних ресурсів. Отримані результати демонструють можливість використання запропонованого підходу для виявлення зон підвищеного ризику та підвищення ефективності планування патрулювання території.

РОЗДІЛ 4. ОЦІНКА ЕФЕКТИВНОСТІ РОБОТИ ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ

Аналіз результатів просторового аналізу кримінальних подій

У даному підрозділі здійснюється аналіз результатів роботи розробленої інформаційно-аналітичної системи на етапі просторового опрацювання вхідних даних. Основною метою цього етапу є виявлення ділянок території, на яких спостерігається підвищена концентрація зареєстрованих подій, та формування основи для подальшого планування маршрутів патрулювання[2].

Робота системи починається з обробки вхідного точкового шару подій у середовищі QGIS. Після часової фільтрації та врахування ваг подій за їх давністю система формує регулярну просторову сітку, у межах якої для кожної комірки обчислюється інтегральний показник ризику. Такий підхід дозволяє перейти від аналізу окремих інцидентів до узагальненої оцінки території та визначити ділянки, які потребують підвищеної уваги з боку патрульних підрозділів.

Результати просторового аналізу показали, що розподіл подій по території є нерівномірним. Частина комірок містить незначну кількість подій або не містить їх зовсім, тоді як окремі ділянки характеризуються підвищеним значенням RiskScore та Intensity. Саме такі комірки визначаються системою як зони підвищеного ризику. Їх виділення дає змогу зосередити увагу не на всій території однаковою мірою, а на найбільш проблемних ділянках, де ймовірність повторного виникнення подій є вищою[11].

Практичне значення цього етапу полягає в тому, що він забезпечує аналітичне обґрунтування для подальших дій системи. На основі побудованої сітки ризику можна не лише візуально оцінити криміногенну ситуацію на території, але й сформувати набір пріоритетних зон для патрулювання. У результаті система переходить від звичайного відображення точок на карті до

структурованого просторового аналізу, який є більш придатним для прийняття управлінських рішень.

Разом з тим отримані результати свідчать, що запропонований підхід має не лише переваги, але й певні обмеження. До позитивних сторін слід віднести наочність представлення результатів, можливість локалізації проблемних зон, автоматизацію аналізу та зменшення впливу суб'єктивного фактора під час оцінювання ситуації. Крім того, використання сіткової моделі забезпечує повторюваність результатів за однакових параметрів запуску алгоритму, що є важливим для інформаційно-аналітичних систем.

Водночас система не може вважатися ідеальною. Результати аналізу залежать від обраних параметрів, зокрема від розміру комірки сітки, глибини історії подій та вагових коефіцієнтів. Якщо розмір комірки буде надто великим, окремі локальні осередки активності можуть бути згладжені. Якщо ж комірка занадто мала, карта ризику може стати надмірно фрагментованою і менш зручною для подальшого використання. Також система враховує лише наявні історичні дані, тому точність просторового аналізу безпосередньо залежить від повноти та актуальності вхідної інформації[4].

Ще одним обмеженням є те, що визначені зони ризику відображають насамперед просторову концентрацію вже зареєстрованих подій, але не враховують усі можливі зовнішні чинники, які можуть впливати на оперативну обстановку, наприклад зміну транспортних потоків, масові заходи, сезонність або оперативну інформацію, що не включена до набору даних. Тому результати роботи системи доцільно розглядати як аналітичну основу для підтримки прийняття рішень, а не як абсолютно точний або вичерпний прогноз[8].

Для узагальнення результатів аналізу доцільно виділити основні переваги та обмеження запропонованої системи. Вони відображають як позитивні сторони застосування просторового аналізу для планування патрулювання, так і фактори, що можуть впливати на точність отриманих результатів.

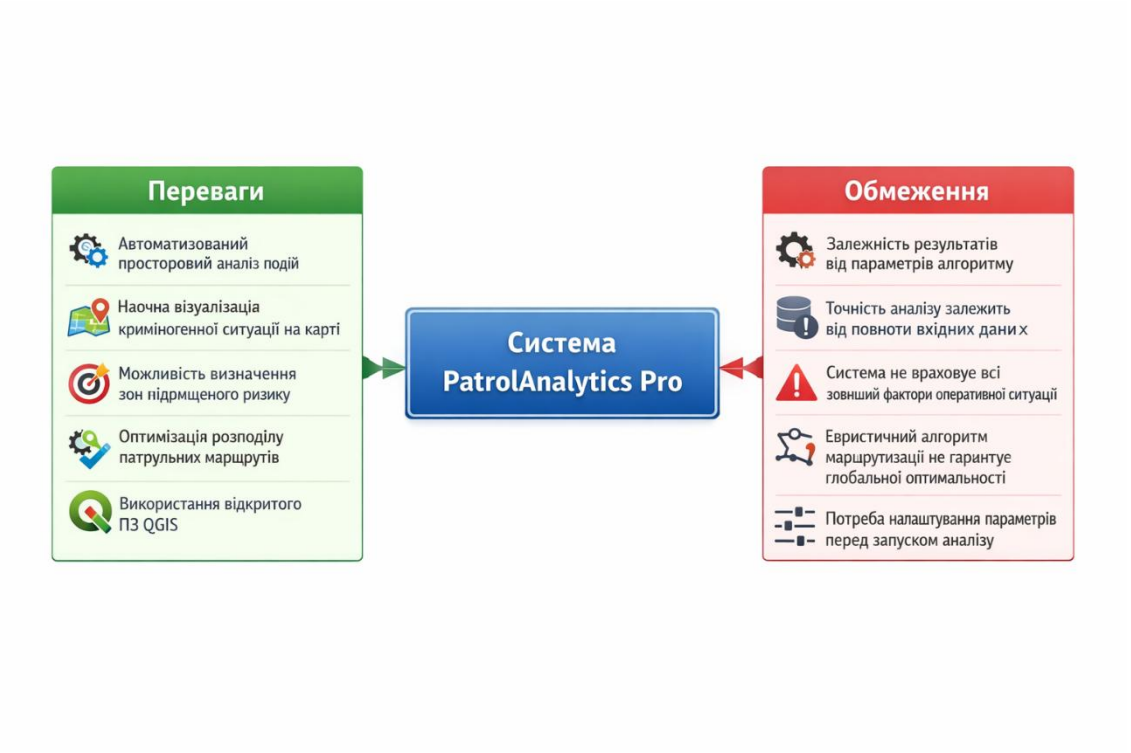

Рис. 4.1. Переваги та обмеження розробленої інформаційно-аналітичної системи

Як видно з рисунка 4.1, запропонована система має ряд важливих переваг, зокрема автоматизацію аналізу просторових даних, можливість виявлення зон підвищеного ризику та наочну візуалізацію результатів. Водночас існують певні обмеження, пов’язані з залежністю результатів від параметрів алгоритму та якості вхідних даних. Тому результати роботи системи доцільно використовувати як інструмент підтримки прийняття рішень, а не як єдину основу для планування патрулювання.

Отже, результати просторового аналізу підтверджують, що розроблена система дозволяє виявляти проблемні ділянки території, структуровано представляти дані про події та формувати основу для подальшого планування маршрутів патрулювання. Водночас отримані результати демонструють необхідність обережного трактування вихідних даних і врахування того, що ефективність системи значною мірою залежить від якості вхідної інформації та налаштування параметрів алгоритму[3].

4.2 Аналіз точності визначення зон підвищеного ризику

Одним із важливих критеріїв ефективності розробленої інформаційно-аналітичної системи є точність визначення зон підвищеного ризику. Даний показник характеризує здатність алгоритму правильно виявляти ділянки території, де спостерігається підвищена концентрація кримінальних подій.

Для оцінювання точності роботи системи було проведено аналіз розподілу зареєстрованих подій відносно визначених алгоритмом зон ризику. У процесі дослідження порівнювалася кількість подій, що потрапляють у комірки сітки з високим значенням інтегрального показника ризику, з кількістю подій у комірках із низьким або середнім рівнем ризику. Розподіл кримінальних подій між зонами різного рівня ризику представлено на рисунку

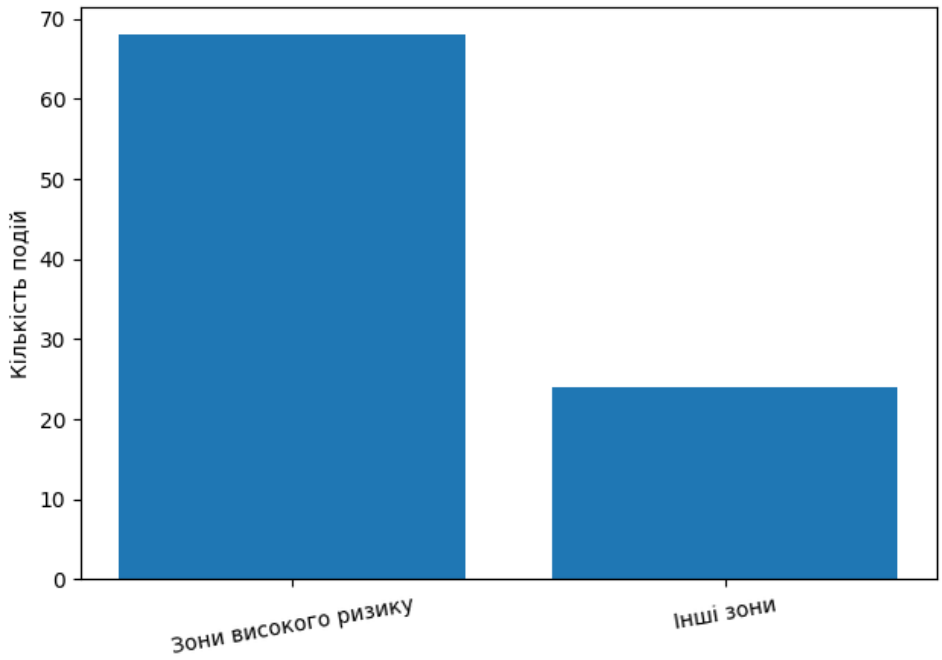

Рис. 4.2**.** Порівняння кількості подій у зонах різного рівня ризику

Як видно з рисунка 4.2, більша частина подій зосереджена саме у зонах підвищеного ризику, що підтверджує ефективність використаного алгоритму просторового аналізу для виявлення проблемних ділянок території.

Разом з тим отримані результати демонструють, що не всі події розташовані виключно в межах визначених зон ризику. Частина інцидентів може виникати і в інших ділянках території, що пояснюється випадковим

характером деяких подій, а також впливом факторів, які не враховуються в межах даної моделі. До таких факторів можуть належати тимчасові зміни оперативної обстановки, сезонні коливання активності або поява нових місць концентрації подій.

Таким чином, результати дослідження свідчать, що розроблена система дозволяє досить точно визначати зони підвищеного ризику та може використовуватися як ефективний інструмент підтримки прийняття рішень під час планування патрулювання. Водночас отримані результати необхідно розглядати як аналітичну рекомендацію, оскільки повна точність визначення ризикових зон залежить від якості вхідних даних та параметрів налаштування алгоритму.

4.3 Аналіз часу виконання алгоритму

Ще одним важливим критерієм оцінювання ефективності розробленої інформаційно-аналітичної системи є швидкодія алгоритму, тобто час, необхідний для виконання основних етапів просторового аналізу та формування результатів. Даний показник є важливим з практичної точки зору, оскільки система повинна забезпечувати оперативну обробку даних та формування рекомендацій для планування патрулювання.

Для оцінювання швидкодії системи було проаналізовано час виконання основних етапів роботи алгоритму. До таких етапів належать підготовка та обробка вхідних даних, виконання просторового аналізу з використанням регулярної сітки, кластеризація точок подій для визначення патрульних баз, а також формування маршрутів патрулювання.

Формування маршрутів патрулювання також виконується за відносно короткий час. Це пояснюється використанням евристичного підходу до побудови маршрутів, який дозволяє отримати прийнятний результат без виконання складних оптимізаційних обчислень.

Результати вимірювання часу виконання основних етапів алгоритму наведено в таблиці 4.1.

Таблиця 4.1

Час виконання основних етапів алгоритму

Етап алгоритму	**Час виконання**
Просторовий аналіз	2–3 с
Кластеризація	1–2 с
Побудова маршрутів	1–2 с

Джерело: складено автором за результатами роботи

Як видно з таблиці 4.1, найбільше часу займає етап просторового аналізу подій, що пояснюється необхідністю обробки великої кількості точкових даних та обчислення інтегральних показників ризику для кожної комірки просторової сітки. Етап кластеризації та визначення патрульних баз виконується дещо швидше, оскільки алгоритм працює лише з відібраними точками подій.

Загалом результати аналізу свідчать, що розроблена система забезпечує достатньо швидке виконання основних етапів обробки даних. Це дозволяє використовувати її для оперативного аналізу кримінальних подій та формування рекомендацій щодо планування патрулювання.

Водночас слід зазначити, що час виконання алгоритму може змінюватися залежно від обсягу вхідних даних, розміру території дослідження та параметрів налаштування алгоритму. У разі значного збільшення кількості подій або зменшення розміру комірки просторової сітки обчислювальні витрати можуть зростати.

4.4 Загальна оцінка ефективності розробленої системи

У процесі дослідження було проведено оцінювання ефективності розробленої інформаційно-аналітичної системи, яка призначена для аналізу кримінальних подій та формування маршрутів патрулювання. Аналіз

результатів роботи системи дозволив оцінити її можливості та визначити ступінь відповідності встановленим критеріям ефективності.

Основними критеріями оцінювання виступали точність визначення зон підвищеного ризику, швидкість виконання алгоритму, а також здатність системи формувати маршрути патрулювання, орієнтовані на найбільш проблемні ділянки території. Отримані результати показали, що запропонований підхід дозволяє достатньо ефективно виконувати просторовий аналіз подій та підтримувати процес планування патрулювання.

Зокрема, аналіз розподілу подій показав, що більшість інцидентів концентрується у межах визначених системою зон підвищеного ризику. Це свідчить про те, що використаний алгоритм просторового аналізу дозволяє коректно виявляти ділянки території з підвищеною ймовірністю виникнення подій.

Крім того, проведений аналіз часу виконання алгоритму показав, що система забезпечує достатньо швидку обробку даних, що є важливим для використання в умовах оперативної діяльності.

Узагальнені результати оцінювання ефективності системи наведено в таблиці 4.2.

Таблиця 4.2

Узагальнена оцінка ефективності розробленої системи

Критерій оцінювання	**Характеристика результату**
Визначення зон підвищеного ризику	система дозволяє виявляти ділянки з підвищеною концентрацією подій
Швидкість виконання алгоритму	основні етапи обробки даних виконуються за декілька секунд
Візуалізація результатів	результати аналізу наочно відображаються на карті
Формування маршрутів патрулювання	маршрути орієнтовані на зони підвищеного ризику

Джерело: складено автором за результатами роботи

Отримані результати свідчать, що розроблена інформаційно-аналітична система може бути ефективно використана для підтримки процесу аналізу кримінальних подій та планування патрулювання. Вона дозволяє автоматизувати частину аналітичної роботи, зменшити вплив суб'єктивних факторів та забезпечити наочне представлення результатів аналізу.

Водночас система не є універсальним рішенням і повинна використовуватися як допоміжний інструмент для прийняття управлінських рішень. Подальший розвиток системи може бути пов'язаний із розширенням набору вхідних даних, використанням додаткових факторів аналізу та вдосконаленням алгоритмів оптимізації маршрутів.

Висновки до четвертого розділу

У даному розділі було проведено аналіз результатів роботи розробленої інформаційно-аналітичної системи для просторового аналізу кримінальних подій та формування маршрутів патрулювання. Оцінювання ефективності системи здійснювалося на основі визначених критеріїв, зокрема точності визначення зон підвищеного ризику, швидкості виконання алгоритму та можливості використання результатів аналізу для планування патрулювання.

Проведений аналіз показав, що запропонований алгоритм дозволяє ефективно виявляти ділянки території з підвищеною концентрацією кримінальних подій. Використання просторової сітки та інтегрального показника ризику забезпечує узагальнення інформації про події та дозволяє визначати найбільш проблемні зони території.

Також було встановлено, що система забезпечує достатньо швидке виконання основних етапів обробки даних. Час виконання алгоритму є прийнятним для практичного використання, що дозволяє застосовувати систему для оперативного аналізу інформації та підтримки процесу прийняття рішень.

ВИСНОВКИ

У процесі виконання дипломної роботи було досліджено можливості застосування геоінформаційних технологій для аналізу просторового розподілу кримінальних подій та підтримки планування патрулювання. Актуальність дослідження зумовлена необхідністю підвищення ефективності аналізу оперативної інформації та оптимізації використання ресурсів підрозділів, що здійснюють патрулювання території.

У першому розділі роботи було проведено аналіз предметної області та розглянуто основні підходи до просторового аналізу подій із використанням геоінформаційних систем. Проаналізовано сучасні методи виявлення зон підвищеного ризику та підходи до оптимізації маршрутів патрулювання. На основі проведеного аналізу було сформульовано основні вимоги до інформаційно-аналітичної системи та визначено задачі дослідження.

У другому розділі було розроблено структуру інформаційно-аналітичної системи для аналізу кримінальних подій. Визначено основні етапи обробки даних, включаючи часову фільтрацію подій, побудову просторової сітки, обчислення інтегрального показника ризику та кластеризацію точок подій для визначення патрульних баз. Також було визначено критерії оцінювання ефективності системи, зокрема точність визначення зон підвищеного ризику, швидкість виконання алгоритму та можливість використання результатів аналізу для підтримки процесу планування патрулювання.

У третьому розділі було реалізовано розроблений алгоритм у середовищі QGIS. Було створено програмний модуль, який виконує просторовий аналіз подій, формує карту ризику та визначає маршрути патрулювання на основі аналізу розподілу подій. Результати роботи алгоритму було візуалізовано у вигляді картографічних матеріалів, що дозволяє наочно оцінювати криміногенну ситуацію на досліджуваній території.

У четвертому розділі було проведено аналіз результатів роботи розробленої системи. Отримані результати показали, що запропонований

підхід дозволяє ефективно виявляти зони підвищеного ризику та забезпечує достатньо швидке виконання основних етапів обробки даних. Аналіз також показав, що система може бути використана як інструмент підтримки прийняття рішень під час планування патрулювання.

Разом з тим встановлено, що ефективність роботи системи значною мірою залежить від якості вхідних даних та параметрів налаштування алгоритму. Крім того, система не враховує всі можливі фактори, що можуть впливати на оперативну обстановку, тому результати її роботи доцільно використовувати як допоміжний аналітичний інструмент.

У результаті виконання дипломної роботи було досягнуто поставленої мети та розроблено інформаційно-аналітичну систему, що дозволяє автоматизувати аналіз просторового розподілу кримінальних подій та підтримувати процес планування патрулювання. Отримані результати можуть бути використані для подальшого вдосконалення систем аналізу оперативної інформації та розвитку інструментів просторової аналітики.

Подальші дослідження можуть бути спрямовані на розширення набору вхідних даних, врахування додаткових факторів, що впливають на криміногенну ситуацію, а також удосконалення алгоритмів оптимізації маршрутів патрулювання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Зацерковний В. І., Бурачек В. Г., Железняк О. О., Терещенко А. О. Геоінформаційні системи і бази даних : навч. посіб. Ніжин : НДУ ім. М. Гоголя, 2014. 492 с.

Костюк В. Л. Міжнародна практика використання поліцією геоінформаційних с

Павленко Л. А. Геоінформаційні системи : навч. посіб. Київ : КНЕУ, 2017. 190 є 196 с.

Часовський О., Андрейчук Ю., Ямелинець Т. Застосування ГІС у природоохоронній справі на прикладі відкритої програми QGIS. Львів : ЛНУ ім

рмакі Х. П., Пилипів Р. М., Веселов М. Ю. Адміністративно-правове регулювання діяльності патрульної поліції щодо забезпечення безпеки дорожнього руху в Україні. URL:

(дата звернення: 15.03.2026).

еоінформаційні системи : навч. посіб. URL:

(дата звернення: 15.03.2026).

вітличний О. О., Плотницький С. В. Основи геоінформатики : навч. посіб. Київ

Національна поліція України

(дата звернення: 15.03.2026).

Додаток А

Програмний модуль просторового аналізу для автоматизованого виявлення зон ризику та побудови оптимальних маршрутів патрулювання

```
"""
Model name: PatrolAnalytics Pro (Full)
Description: Повний модуль: Time window + recency weights + grid risk + top% + centroids + deployment (K-means) + heuristic routes.
Author: Student
"""

from qgis.PyQt.QtCore import QCoreApplication, QVariant
from qgis.core import (
    QgsProcessing,
    QgsProcessingAlgorithm,
    QgsProcessingParameterFeatureSource,
    QgsProcessingParameterNumber,
    QgsProcessingParameterString,
    QgsProcessingParameterFeatureSink,
    QgsWkbTypes,
    QgsFeatureSink,
    QgsProcessingException,
    QgsFields,
    QgsField,
    QgsFeature,
    QgsGeometry,
    QgsPointXY,
    QgsProcessingUtils,
)
```

```
import processing
import math



class PatrolAnalyticsProFull(QgsProcessingAlgorithm):
    # -------- Parameters --------
    INPUT = "INPUT"
    GRID_SIZE = "GRID_SIZE"

    DATE_FIELD = "DATE_FIELD"
    DAYS_BACK = "DAYS_BACK"

    W1_DAYS = "W1_DAYS"
    W1_WEIGHT = "W1_WEIGHT"
    W2_DAYS = "W2_DAYS"
    W2_WEIGHT = "W2_WEIGHT"
    W3_DAYS = "W3_DAYS"
    W3_WEIGHT = "W3_WEIGHT"

    TOP_PERCENT = "TOP_PERCENT"

    K_PATROLS = "K_PATROLS"
    MAX_POINTS_PER_PATROL = "MAX_POINTS_PER_PATROL"
    SCORE_FIELD = "SCORE_FIELD"  # поле пріоритету для патрулювання (звично intensity)

    # -------- Outputs --------
    OUTPUT_RISK_GRID = "OUTPUT_RISK_GRID"
    OUTPUT_RISK_CENTROIDS = "OUTPUT_RISK_CENTROIDS"
    OUTPUT_PATROL_BASES = "OUTPUT_PATROL_BASES"
```

```
    OUTPUT_PATROL_POINTS = "OUTPUT_PATROL_POINTS"

    OUTPUT_PATROL_ROUTES = "OUTPUT_PATROL_ROUTES"


    # ------------------------------

    # Service methods

    # ------------------------------

    def tr(self, string):

        return QCoreApplication.translate("PatrolAnalyticsProFull", string)


    def createInstance(self):

        return PatrolAnalyticsProFull()


    def name(self):

        return "patrol_analytics_pro_full"


    def displayName(self):

        return self.tr("PatrolAnalytics Pro (Full: Risk + Deployment +
Routes)")


    def group(self):

        return self.tr("Patrol / OSINT Analytics")


    def groupId(self):

        return "patrol_osint_analytics"


    def shortHelpString(self):

        return self.tr(

            "Повний модуль: "

            "1) time window (останні N днів) + recency weights; "
```

```
            "2) сітка ризику: RiskScore (сума ваг) + Intensity (на площу) + відбір top X%; "
            "3) центроїди осередків; "
            "4) розподіл центроїдів між K патрулями (K-means) + евристичні маршрути (nearest-neighbor)."
        )

    # ------------------------------
    # Parameters
    # ------------------------------
    def initAlgorithm(self, config=None):
        self.addParameter(
            QgsProcessingParameterFeatureSource(
                self.INPUT,
                self.tr("Вхідні події (точковий шар)"),
                [QgsProcessing.TypeVectorPoint],
            )
        )

        self.addParameter(
            QgsProcessingParameterNumber(
                self.GRID_SIZE,
                self.tr("Розмір комірки сітки (в одиницях CRS шару)"),
                type=QgsProcessingParameterNumber.Double,
                defaultValue=250.0,
                minValue=0.000001,
            )
        )

        self.addParameter(
```

```
            QgsProcessingParameterString(
                self.DATE_FIELD,
                self.tr("Поле дати/часу (тип Date/DateTime у шарі)"),
                defaultValue="event_time",
            )
        )

        self.addParameter(
            QgsProcessingParameterNumber(
                self.DAYS_BACK,
                self.tr("Часове вікно: брати події за останні N днів"),
                type=QgsProcessingParameterNumber.Integer,
                defaultValue=60,
                minValue=1,
            )
        )

        # Ступінчасті ваги за давністю
        self.addParameter(
            QgsProcessingParameterNumber(
                self.W1_DAYS, self.tr("Поріг 1 (днів тому): 0..поріг1"),
                type=QgsProcessingParameterNumber.Integer,     defaultValue=7,
minValue=0
            )
        )
        self.addParameter(
            QgsProcessingParameterNumber(
                self.W1_WEIGHT, self.tr("Вага 1 (найсвіжіші події)"),
                type=QgsProcessingParameterNumber.Double,    defaultValue=1.0,
minValue=0.0
```

```
            )
        )

        self.addParameter(
            QgsProcessingParameterNumber(
                self.W2_DAYS,      self.tr("Поріг      2      (днів      тому):
(поріг1+1)..поріг2"),
                type=QgsProcessingParameterNumber.Integer,    defaultValue=30,
minValue=0
            )
        )
        self.addParameter(
            QgsProcessingParameterNumber(
                self.W2_WEIGHT, self.tr("Вага 2"),
                type=QgsProcessingParameterNumber.Double,    defaultValue=0.6,
minValue=0.0
            )
        )

        self.addParameter(
            QgsProcessingParameterNumber(
                self.W3_DAYS,      self.tr("Поріг      3      (днів      тому):
(поріг2+1)..поріг3"),
                type=QgsProcessingParameterNumber.Integer,    defaultValue=60,
minValue=0
            )
        )
        self.addParameter(
            QgsProcessingParameterNumber(
                self.W3_WEIGHT, self.tr("Вага 3"),
```

```
                type=QgsProcessingParameterNumber.Double,    defaultValue=0.3,
minValue=0.0
            )
        )

        self.addParameter(
            QgsProcessingParameterNumber(
                self.TOP_PERCENT,
                self.tr("Top X% комірок за Intensity (1..100)"),
                type=QgsProcessingParameterNumber.Integer,
                defaultValue=20,
                minValue=1,
                maxValue=100,
            )
        )

        # Параметри патрулювання
        self.addParameter(
            QgsProcessingParameterNumber(
                self.K_PATROLS,
                self.tr("Кількість патрулів (K)"),
                type=QgsProcessingParameterNumber.Integer,
                defaultValue=3,
                minValue=1,
            )
        )
        self.addParameter(
            QgsProcessingParameterNumber(
                self.MAX_POINTS_PER_PATROL,
```

```
type=QgsProcessingParameterNumber.Integer,
                defaultValue=0,
                minValue=0,
            )
        )
        self.addParameter(
            QgsProcessingParameterString(
                self.SCORE_FIELD,

defaultValue="intensity",
            )
        )

        # Outputs
        self.addParameter(
            QgsProcessingParameterFeatureSink(
                self.OUTPUT_RISK_GRID,
                self.tr("Сітка ризику (top X% з RiskScore/Intensity)"),
                QgsProcessing.TypeVectorPolygon,
s           )
e
        )
l
f       self.addParameter(
tr("Поле пріоритету для маршруту (звично: intensity)"),
            QgsProcessingParameterFeatureSink(
                self.OUTPUT_RISK_CENTROIDS,
                self.tr("Центроїди осередків ризику (top X%)"),
                QgsProcessing.TypeVectorPoint,
            )
        )
        self.addParameter(
```

```
            QgsProcessingParameterFeatureSink(
                self.OUTPUT_PATROL_BASES,
                self.tr("Рекомендовані дислокації патрулів (центри кластерів)"),
                QgsProcessing.TypeVectorPoint,
            )
        )
        self.addParameter(
            QgsProcessingParameterFeatureSink(
                self.OUTPUT_PATROL_POINTS,
                self.tr("Точки патрулювання (з patrol_id та seq)"),
                QgsProcessing.TypeVectorPoint,
            )
        )
        self.addParameter(
            QgsProcessingParameterFeatureSink(
                self.OUTPUT_PATROL_ROUTES,
                self.tr("Маршрути патрулювання (лінії між точками, евристика)"),
                QgsProcessing.TypeVectorLine,
            )
        )

    # ------------------------------
    # Helpers
    # ------------------------------
    @staticmethod
    def _dist2(a: QgsPointXY, b: QgsPointXY) -> float:
        dx = a.x() - b.x()
        dy = a.y() - b.y()
```

```
        return dx * dx + dy * dy

    @staticmethod
    def _geom_pointxy(g: QgsGeometry):
        if g is None or g.isEmpty():
            return None
        p = g.asPoint()
        return QgsPointXY(p.x(), p.y())

    @staticmethod
    def _safe_float(v):
        try:
            return float(v) if v is not None else 0.0
        except Exception:
            return 0.0

    @staticmethod
    def _as_layer(output, context):
        layer = QgsProcessingUtils.mapLayerFromString(output, context)
        if layer is None:
            raise QgsProcessingException(f"Не вдалося отримати шар із результату: {output}")
        return layer

    # ------------------------------
    # Core
    # ------------------------------
    def processAlgorithm(self, parameters, context, feedback):
        src = self.parameterAsSource(parameters, self.INPUT, context)
        if src is None:
```

```
            raise QgsProcessingException("Не вдалося отримати вхідний точковий
шар.")


        grid_size = float(self.parameterAsDouble(parameters, self.GRID_SIZE,
context))
        if grid_size <= 0:
            raise QgsProcessingException("GRID_SIZE має бути > 0.")


        date_field = self.parameterAsString(parameters, self.DATE_FIELD,
context).strip()
        if not date_field:
            raise QgsProcessingException("DATE_FIELD не задано.")


        days_back = int(self.parameterAsInt(parameters, self.DAYS_BACK,
context))


        w1_days = int(self.parameterAsInt(parameters, self.W1_DAYS, context))
        w2_days = int(self.parameterAsInt(parameters, self.W2_DAYS, context))
        w3_days = int(self.parameterAsInt(parameters, self.W3_DAYS, context))
        w1 = float(self.parameterAsDouble(parameters, self.W1_WEIGHT,
context))
        w2 = float(self.parameterAsDouble(parameters, self.W2_WEIGHT,
context))
        w3 = float(self.parameterAsDouble(parameters, self.W3_WEIGHT,
context))


        if not (0 <= w1_days <= w2_days <= w3_days):
            raise QgsProcessingException("Потрібно: 0 <= поріг1 <= поріг2 <=
поріг3.")


        top_percent = int(self.parameterAsInt(parameters, self.TOP_PERCENT,
context))
```

```
        k_patrols = int(self.parameterAsInt(parameters, self.K_PATROLS, context))
        max_points = int(self.parameterAsInt(parameters, self.MAX_POINTS_PER_PATROL, context))
        score_field = self.parameterAsString(parameters, self.SCORE_FIELD, context).strip()

        extent = src.sourceExtent()

        crs = src.sourceCrs()
        feedback.pushInfo(f"CRS: {crs.authid() if crs.isValid() else 'невідомо'}")
        feedback.pushInfo(f"Extent: {extent.toString()}")

        # --------- (A) Time window filter ---------
        # days_expr: кількість днів тому (ціле)
        days_expr = f"to_int( age(now(), \"{date_field}\") / make_interval(days:=1) )"
        filter_expr = f"\"{date_field}\" IS NOT NULL AND {days_expr} >= 0 AND {days_expr} <= {days_back}"

        filtered_pts = processing.run(
            "native:extractbyexpression",
            {"INPUT": parameters[self.INPUT], "EXPRESSION": filter_expr, "OUTPUT": "TEMPORARY_OUTPUT"},
            context=context, feedback=feedback, is_child_algorithm=True
        )["OUTPUT"]
        filtered_pts = self._as_layer(filtered_pts, context)

        # --------- (B) Add recency weight field w ---------
        weight_expr = (
```

```
    f"CASE "
    f"WHEN {days_expr} <= {w1_days} THEN {w1} "
    f"WHEN {days_expr} <= {w2_days} THEN {w2} "
    f"WHEN {days_expr} <= {w3_days} THEN {w3} "
    f"ELSE 0 "
    f"END"
)

weighted_pts = processing.run(
    "native:fieldcalculator",
    {
        "INPUT": filtered_pts,
        "FIELD_NAME": "w",
        "FIELD_TYPE": 0,  # Float
        "FIELD_LENGTH": 10,
        "FIELD_PRECISION": 3,
        "FORMULA": weight_expr,
        "OUTPUT": "TEMPORARY_OUTPUT",
    },
    context=context, feedback=feedback, is_child_algorithm=True
)["OUTPUT"]
weighted_pts = self._as_layer(weighted_pts, context)

# --------- (C) Build grid ---------
grid_layer = processing.run(
    "native:creategrid",
    {
        "TYPE": 2,  # rectangle polygons
        "EXTENT": extent,
```

```
            "HSPACING": grid_size,
            "VSPACING": grid_size,
            "HOVERLAY": 0.0,
            "VOVERLAY": 0.0,
            "CRS": crs,
            "OUTPUT": "TEMPORARY_OUTPUT",
        },
        context=context, feedback=feedback, is_child_algorithm=True
    )["OUTPUT"]
    grid_layer = self._as_layer(grid_layer, context)

    # --------- (D) Aggregate weights in polygons (RiskScore) ---------
    # Creates field evt_w_sum as sum(w)
    joined = processing.run(
     "native:joinbylocationsummary",
        {
            "INPUT": grid_layer,
            "JOIN": weighted_pts,
            "PREDICATE": [0],  # intersects
            "JOIN_FIELDS": ["w"],
            "SUMMARIES": [5],  # sum
            "DISCARD_NONMATCHING": False,
            "PREFIX": "evt_",
            "OUTPUT": "TEMPORARY_OUTPUT",
        },
        context=context, feedback=feedback, is_child_algorithm=True
    )["OUTPUT"]
    joined = self._as_layer(joined, context)
```

```
        # Add area, intensity
        with_area = processing.run(
            "native:fieldcalculator",
            {
                "INPUT": joined,
                "FIELD_NAME": "area",
                "FIELD_TYPE": 0,
                "FIELD_LENGTH": 20,
                "FIELD_PRECISION": 3,
                "FORMULA": "$area",
                "OUTPUT": "TEMPORARY_OUTPUT",
            },
            context=context, feedback=feedback, is_child_algorithm=True
        )["OUTPUT"]
        with_area = self._as_layer(with_area, context)

        with_intensity = processing.run(
            "native:fieldcalculator",
            {
                "INPUT": with_area,
                "FIELD_NAME": "intensity",
                "FIELD_TYPE": 0,
                "FIELD_LENGTH": 20,
                "FIELD_PRECISION": 6,
                "FORMULA": "CASE WHEN \"area\" > 0 THEN coalesce(\"evt_w_sum\",
0) / \"area\" ELSE 0 END",
                "OUTPUT": "TEMPORARY_OUTPUT",
            },
            context=context, feedback=feedback, is_child_algorithm=True
        )["OUTPUT"]
```

```
        with_intensity = self._as_layer(with_intensity, context)

        # --------- (E) Select top X% cells by intensity ---------
        sorted_layer = processing.run(
            "native:orderbyexpression",
            {"INPUT":    with_intensity,    "EXPRESSION":    "\"intensity\"",
"ASCENDING": False, "NULLS_FIRST": False, "OUTPUT": "TEMPORARY_OUTPUT"},
            context=context, feedback=feedback, is_child_algorithm=True
        )["OUTPUT"]
        sorted_layer = self._as_layer(sorted_layer, context)

        ranked_layer = processing.run(
            "native:addautoincrementalfield",
            {
                "INPUT": sorted_layer,
                "FIELD_NAME": "rank",
                "START": 1,
                "GROUP_FIELDS": [],
                "SORT_EXPRESSION": "",
                "SORT_ASCENDING": True,
                "SORT_NULLS_FIRST": False,
                "OUTPUT": "TEMPORARY_OUTPUT",
            },
            context=context, feedback=feedback, is_child_algorithm=True
        )["OUTPUT"]
        ranked_layer = self._as_layer(ranked_layer, context)

        n_cells = ranked_layer.featureCount()
        if n_cells == 0:
```

```
            raise QgsProcessingException("Сітка не містить об’єктів. Перевірте екстент/дані.")

        k_top = int((n_cells * top_percent + 99) // 100)  # ceil
        if k_top < 1:
            k_top = 1

        feedback.pushInfo(f"Cells: {n_cells}. Top {top_percent}% => {k_top} cells.")

        top_grid = processing.run(
            "native:extractbyexpression",
            {"INPUT": ranked_layer, "EXPRESSION": f"\"rank\" <= {k_top}", "OUTPUT": "TEMPORARY_OUTPUT"},
            context=context, feedback=feedback, is_child_algorithm=True
        )["OUTPUT"]
        top_grid = self._as_layer(top_grid, context)

        # --------- (F) Centroids of risk cells ---------
        risk_centroids = processing.run(
            "native:centroids",
            {"INPUT": top_grid, "ALL_PARTS": False, "OUTPUT": "TEMPORARY_OUTPUT"},
            context=context, feedback=feedback, is_child_algorithm=True
        )["OUTPUT"]
        risk_centroids = self._as_layer(risk_centroids, context)

        # --------- Write risk grid + centroids to sinks ---------
        (grid_sink, grid_sink_id) = self.parameterAsSink(
            parameters, self.OUTPUT_RISK_GRID, context,
```

```
            top_grid.fields(), QgsWkbTypes.Polygon, top_grid.sourceCrs()
        )
        if grid_sink is None:
            raise QgsProcessingException("Не вдалося створити OUTPUT_RISK_GRID.")

        for f in top_grid.getFeatures():
            grid_sink.addFeature(f, QgsFeatureSink.FastInsert)

        (cent_sink, cent_sink_id) = self.parameterAsSink(
            parameters, self.OUTPUT_RISK_CENTROIDS, context,
            risk_centroids.fields(), QgsWkbTypes.Point, risk_centroids.sourceCrs()
        )
        if cent_sink is None:
            raise QgsProcessingException("Не вдалося створити OUTPUT_RISK_CENTROIDS.")

        for f in risk_centroids.getFeatures():
            cent_sink.addFeature(f, QgsFeatureSink.FastInsert)

        # --------- (G) Deployment: K-means on centroids ---------
        # Якщо точок менше, ніж K, зменшити K
        n_pts = risk_centroids.featureCount()

            # Повертаємо тільки ризикову сітку/центроїди (порожні виходи для патрулів)
f
e
            # Створимо порожні sinks для патрульних шарів, щоб алгоритм не падав
e
dself._create_empty_patrol_outputs(parameters, context, crs)
b
1
f
d
c
k
p
pushInfo("Ризикових центроїдів немає. Патрулювання не сформовано.")
```

```
            return {
                self.OUTPUT_RISK_GRID: grid_sink_id,
                self.OUTPUT_RISK_CENTROIDS: cent_sink_id,
                self.OUTPUT_PATROL_BASES: self._empty_base_id,
                self.OUTPUT_PATROL_POINTS: self._empty_points_id,
                self.OUTPUT_PATROL_ROUTES: self._empty_routes_id,
            }

        if k_patrols > n_pts:
            feedback.pushInfo(f"K={k_patrols} > N_points={n_pts}. Зменшую K до
{n_pts}.")
            k_patrols = n_pts

        clustered = processing.run(
            "native:kmeansclustering",
            {
                "INPUT": risk_centroids,
                "CLUSTERS": k_patrols,
                "FIELD_NAME": "patrol_id",
                "SIZE_FIELD_NAME": "cluster_sz",
                "OUTPUT": "TEMPORARY_OUTPUT",
            },
            context=context, feedback=feedback, is_child_algorithm=True
        )["OUTPUT"]
        clustered = self._as_layer(clustered, context)

        clustered_fields = clustered.fields()

        # --------- (H) Build patrol bases (cluster centers) ---------
        base_fields = QgsFields()
```

```
        base_fields.append(QgsField("patrol_id", QVariant.Int))
        base_fields.append(QgsField("n_points", QVariant.Int))
        base_fields.append(QgsField("sum_score", QVariant.Double))
        base_fields.append(QgsField("avg_score", QVariant.Double))

        (base_sink, base_id) = self.parameterAsSink(
            parameters, self.OUTPUT_PATROL_BASES, context,
            base_fields, QgsWkbTypes.Point, clustered.sourceCrs()
        )
        if base_sink is None:
            raise QgsProcessingException("Не вдалося створити OUTPUT_PATROL_BASES.")

        # Collect points per patrol
        patrol_dict = {i: [] for i in range(k_patrols)}  # pid -> list of (feature, point, score)
        field_names = clustered_fields.names()

        for f in clustered.getFeatures():
            pid = f["patrol_id"]
            try:
                pid = int(pid)
            except Exception:
                continue
            p = self._geom_pointxy(f.geometry())
            if p is None:
                continue

            # score for patrol planning
            score = 0.0
```

```
            if score_field and score_field in field_names:
                score = self._safe_float(f[score_field])

            patrol_dict.setdefault(pid, []).append((f, p, score))

        # Apply max_points cap (top by score)
        if max_points > 0:
            for pid, lst in patrol_dict.items():
                lst.sort(key=lambda x: x[2], reverse=True)
                patrol_dict[pid] = lst[:max_points]

        # Compute & write bases
        bases = {}  # pid -> point
        for pid, lst in patrol_dict.items():
            if not lst:
                continue
            sx = sum(p.x() for _, p, _ in lst)
            sy = sum(p.y() for _, p, _ in lst)
            base_pt = QgsPointXY(sx / len(lst), sy / len(lst))
            bases[pid] = base_pt

            sum_score = sum(s for _, _, s in lst)
            avg_score = sum_score / len(lst) if lst else 0.0

            bf = QgsFeature(base_fields)
            bf.setGeometry(QgsGeometry.fromPointXY(base_pt))
            bf.setAttributes([pid,         len(lst),         float(sum_score),
float(avg_score)])
            base_sink.addFeature(bf, QgsFeatureSink.FastInsert)
```

```
        # --------- (I) Patrol points output: original centroid fields + seq -
--------
        out_point_fields = QgsFields()
        for fld in clustered_fields:
            out_point_fields.append(fld)
        out_point_fields.append(QgsField("seq", QVariant.Int))

        (pt_sink, pt_id) = self.parameterAsSink(
            parameters, self.OUTPUT_PATROL_POINTS, context,
            out_point_fields, QgsWkbTypes.Point, clustered.sourceCrs()
        )
        if pt_sink is None:
            raise      QgsProcessingException("Не      вдалося      створити
OUTPUT_PATROL_POINTS.")

        # --------- (J) Patrol routes output (segments) ---------
        route_fields = QgsFields()
        route_fields.append(QgsField("patrol_id", QVariant.Int))
        route_fields.append(QgsField("from_seq", QVariant.Int))
        route_fields.append(QgsField("to_seq", QVariant.Int))
        route_fields.append(QgsField("seg_len", QVariant.Double))

        (route_sink, route_id) = self.parameterAsSink(
            parameters, self.OUTPUT_PATROL_ROUTES, context,
            route_fields, QgsWkbTypes.LineString, clustered.sourceCrs()
        )
        if route_sink is None:
            raise      QgsProcessingException("Не      вдалося      створити
OUTPUT_PATROL_ROUTES.")
```

```
# --------- (K) Nearest-neighbor ordering per patrol ---------
for pid, lst in patrol_dict.items():
    if not lst:
        continue

    start = bases.get(pid)
    if start is None:
        start = lst[0][1]

    remaining = lst[:]
    ordered = []
    current = start

    while remaining:
        best_i = 0
        best_d2 = self._dist2(current, remaining[0][1])
        for i in range(1, len(remaining)):
            d2 = self._dist2(current, remaining[i][1])
            if d2 < best_d2:
                best_d2 = d2
                best_i = i
        item = remaining.pop(best_i)
        ordered.append(item)
        current = item[1]

    # write points with seq
    for seq, (f, p, _s) in enumerate(ordered, start=1):
        of = QgsFeature(out_point_fields)
        of.setGeometry(QgsGeometry.fromPointXY(p))
```

```
                attrs = list(f.attributes()) + [seq]
                of.setAttributes(attrs)
                pt_sink.addFeature(of, QgsFeatureSink.FastInsert)

            # write route segments
            for i in range(len(ordered) - 1):
                p1 = ordered[i][1]
                p2 = ordered[i + 1][1]
                geom = QgsGeometry.fromPolylineXY([p1, p2])
                seg_len = math.sqrt(self._dist2(p1, p2))

                rf = QgsFeature(route_fields)
                rf.setGeometry(geom)
                rf.setAttributes([pid, i + 1, i + 2, float(seg_len)])
                route_sink.addFeature(rf, QgsFeatureSink.FastInsert)

        return {
            self.OUTPUT_RISK_GRID: grid_sink_id,
            self.OUTPUT_RISK_CENTROIDS: cent_sink_id,
            self.OUTPUT_PATROL_BASES: base_id,
            self.OUTPUT_PATROL_POINTS: pt_id,
            self.OUTPUT_PATROL_ROUTES: route_id,
        }

    # ----------------------------
    # Create empty patrol outputs if no centroids
    # ----------------------------
    def _create_empty_patrol_outputs(self, parameters, context, crs):
        base_fields = QgsFields()
```

```
        base_fields.append(QgsField("patrol_id", QVariant.Int))
        base_fields.append(QgsField("n_points", QVariant.Int))
        base_fields.append(QgsField("sum_score", QVariant.Double))
        base_fields.append(QgsField("avg_score", QVariant.Double))
        (base_sink, base_id) = self.parameterAsSink(
            parameters, self.OUTPUT_PATROL_BASES, context, base_fields,
QgsWkbTypes.Point, crs
        )

        pt_fields = QgsFields()
        pt_fields.append(QgsField("patrol_id", QVariant.Int))
        pt_fields.append(QgsField("seq", QVariant.Int))
        (pt_sink, pt_id) = self.parameterAsSink(
            parameters, self.OUTPUT_PATROL_POINTS, context, pt_fields,
QgsWkbTypes.Point, crs
        )

        route_fields = QgsFields()
        route_fields.append(QgsField("patrol_id", QVariant.Int))
        route_fields.append(QgsField("from_seq", QVariant.Int))
        route_fields.append(QgsField("to_seq", QVariant.Int))
        route_fields.append(QgsField("seg_len", QVariant.Double))
        (route_sink, route_id) = self.parameterAsSink(
            parameters, self.OUTPUT_PATROL_ROUTES, context, route_fields,
QgsWkbTypes.LineString, crs
        )

        self._empty_base_id = base_id
        self._empty_points_id = pt_id
        self._empty_routes_id = route_id
```