Зміст

[Перелік термінів та умовних позначень 3](#_Toc263852278)

[Вступ 4](#_Toc263852279)

[1. Кластерний аналіз 8](#_Toc263852280)

[1.1. Постановка задачі 8](#_Toc263852281)

[1.2. Застосування кластеризації 9](#_Toc263852282)

[1.3. Відомі методи розв’язку 12](#_Toc263852283)

[2. Еволюційні обчислення 21](#_Toc263852284)

[2.1. Еволюційні алгоритми 21](#_Toc263852285)

[2.2. Генетичні алгоритми 23](#_Toc263852286)

[3. Розробка генетичного алгоритму кластеризації 28](#_Toc263852287)

[3.1. Загальний принцип роботи 28](#_Toc263852288)

[3.2. Кодування геному 32](#_Toc263852289)

[3.3. Оцінка пристосованості 33](#_Toc263852290)

[3.4. Початкова популяція 34](#_Toc263852291)

[3.5. Селекція 36](#_Toc263852292)

[3.6. Рекомбінація 37](#_Toc263852293)

[3.7. Мутація 39](#_Toc263852294)

[3.8. Репродукція 40](#_Toc263852295)

[3.9. Критерій зупинки 40](#_Toc263852296)

[3.10. Критерій кластеризації 41](#_Toc263852297)

[4. Аналіз результатів роботи алгоритму 45](#_Toc263852298)

[4.1. Визначення оптимальних параметрів 45](#_Toc263852299)

[4.2. Порівняння результатів 45](#_Toc263852300)

[5. Висновки 48](#_Toc263852301)

[5.1. Переваги та недоліки алгоритму 48](#_Toc263852302)

[5.2. Перспективи вдосконалення алгоритму 49](#_Toc263852303)

[Список використаної літератури 50](#_Toc263852304)

[Додаток 1. Копії графічних матеріалів 51](#_Toc263852305)

[Додаток 2. Програмний код генетичних операцій 55](#_Toc263852306)

# Перелік термінів та умовних позначень

**ГА** – генетичний алгоритм

**Ген** – найменша функціональна і структурна одиниця спадковості індивідів

**Геном, хромосома** – множина всіх параметрів, що описує розв’язок задачі

**Генотип** – сукупність всіх генів конкретного індивіда

**ЕА** – еволюційний алгоритм

**Індивід, особина, організм** – потенційний кандидат на розв’язок задачі

**Кластер** – множина схожих за метрикою об’єктів або спостережень

**Кластеризація** – розбиття заданої множини об’єктів або спостережень на кластери

**Метрика** – спосіб оцінки міри подібності об’єктів або спостережень при кластеризації

**Мутація** – генетична операція для підтримання різноманітності від одного покоління популяції хромосом до наступного

**Популяція** – сукупність всіх організмів, які співіснують у виділеному просторі і претендують на розв’язок задачі

**Репродукція** – створення наступного нового покоління нащадків старим попереднім поколінням

**Селекція** – етап ГА, на якому виконується відбір індивідів для здійснення подальшого розмноження

**Схрещування, рекомбінація, кроссовер (crossover)** – генетична операція для наслідування хромосом від одного покоління до наступного

**Функція пристосованості, фітнес-функція (fitness)** – цільова функція для оцінки оптимальності розв’язку задачі і порівняння з іншими розв’язками

# Вступ

Останнім часом збільшення об’ємів інформації та даних потребує розробки та введення нових методів щодо вирішення задач аналізу даних. Однією з актуальних проблем є розбиття по категоріям даних з великою кількістю взаємопов’язаних ознак, зокрема в багатовимірній статистиці.

Кластерний аналіз є однією з фундаментальних задач аналізу даних (Data Mining). Вперше даний термін був введений Тріоном (Tryon) в 1939 році. На відміну від задач класифікації, кластерний аналіз не накладає обмеження на представлення об’єктів, що досліджуються, дозволяє аналізувати показники різних типів даних (інтервальні, частотні, бінарні тощо). Кластерний аналіз дає можливість скорочувати розмірність даних, робити їх наглядними. Кластерний аналіз може застосовуватись до сукупностей часових рядів, де мають виділятися періоди схожості деяких показників і виявлятися групи часових рядів зі схожою динамікою.

Кластерний аналіз паралельно розвивався у декількох напрямках, таких як біологія, психологія та інших, тому більшість методів має по дві і більше назв.

Задачі кластерного аналізу можна об’єднати в наступні групи:

1. розробка типології або класифікації;
2. дослідження корисних концептуальних схем групування об’єктів;
3. представлення гіпотез на основі дослідження даних;
4. перевірка гіпотез або досліджень для визначення, чи дійсно типи (групи), виділені тим чи іншим способом, присутні в наявних даних.

Результатом кластерного аналізу можуть бути не лише набори даних, об’єднані в кластери. В залежності від предмета дослідження може бути поставлена, наприклад, задача виявлення новизни, коли виділяються нетипові об’єкти, які не вдається приєднати до жодного з кластерів.

Розв’язок задачі кластеризації, як правило, не є однозначним. Досить часто кластеризація може бути виконана багатьма різними способами в залежності від критерію та методики. У випадку, коли дані є не числовими, а категоріальними, виникає складність оцінки в числовому вигляді якості кластеризації. Ще складнішою є ситуація, коли в просторі ознак зустрічаються як кількісні, так і якісні характеристики. Вирішення цих проблем є завданням аналітика.

Пріоритетною вимогою до виконання кластеризації може бути максимальна швидкість або якість. Але з розвитком розподілених обчислювальних систем та появою надвеликих баз даних останнім часом досить важливою характеристикою алгоритму кластеризації є масштабованість, можливість його простого розпаралелювання.

В залежності від досліджуваних даних та їх характеристик, основною вимогою до кластеризації є критерій, відповідно до якого вона буде здійснюватись. Виходячи зі способу оцінки подібності та розбіжності об’єктів (метрики), формулюється критерій, згідно з яким дані будуть об’єднуватися у кластери. Для однієї і тієї ж метрики може бути сформульовано безліч критеріїв, які за своїм змістом можуть бути навіть протилежними. Так, наприклад, у найпростішому випадку, може відбуватися об’єднання найближчих за відстанню об’єктів у багатовимірному просторі, але іноді виникає потреба об’єднання найвіддаленіших об’єктів. Така постановка задачі може бути сформульована, наприклад, для формування груп осіб з метою проведення соціологічних опитувань, випробовування лікарських засобів тощо. Досить часто необхідно обробляти одні виміри з більшими ваговими коефіцієнтами, ніж інші, що також враховується критерієм кластеризації.

На сьогоднішній день існує декілька десятків алгоритмів кластеризації та їх модифікацій [1]. Як правило, кожен із них при роботі з однією і тією ж множиною даних дає різні результати. Кожен з цих методів має свої переваги та недоліки. При цьому не існує ідеального алгоритму, який би виконував кластеризацію відповідно до заданого критерію з максимальною якістю. Найбільш поширені алгоритми виконують кластеризацію відповідно до одного критерію, на якому, по суті, базується сама ідея роботи алгоритму. Так, наприклад, один з найвідоміших алгоритмів – k-середніх (k-means) та його модифікація з використанням нечіткої логіки (fuzzy c-means, FCM) побудовані на основі критерію кластеризації, що мінімізує відстані до центру тяжіння кожного кластеру. Даний алгоритм має високу швидкодію, але абсолютно не приданий для вирішення задач з іншими критеріями кластеризації. Крім того, для деяких наборів вхідних даних розрахунок центра тяжіння є неможливим. Ще одним із суттєвих недоліків алгоритму k-середніх є ефект «розщеплення» великого кластеру. Також досить часто одна числова характеристика об’єкта може бути на порядок більшою, ніж інша. У такому випадку необхідно проводити нормалізацію таких величин даних, щоб забезпечити їх однакову вагу. Але тоді потрібно спотворити або продублювати з нормалізацією всі вхідні дані. Ієрархічні алгоритми кластеризації непридатні для роботи з великими масивами даних, а також є чуттєвими до викидів.

У даній роботі запропоновано новий підхід до вирішення задачі – за допомогою генетичного алгоритму (ГА). Вибір ґрунтується на тому, що на даний момент використання адаптивних алгоритмів є малодослідженим стосовно задач кластерного аналізу і немає достовірної інформації щодо їх розробки у цій сфері. Генетичні алгоритми є привабливими у даному випадку, оскільки для них характерна збіжність до локальних екстремумів, які і представляють кластери у просторі рішень. Крім того, генетичні алгоритми виконують пошук відповідно до цільової функції (фітнес-функції пристосованості), яка і визначає їх поведінку. Змінюючи реалізацію даної функції, можна задавати критерій кластеризації, залишаючи при цьому реалізацію ГА незмінною. Таким чином можна розробити ГА, який може виконувати кластеризацію відповідно до будь-якого критерію. Задачею аналітика є лише розробка функції, яка дає числову характеристику якості вирішення, а ГА сам виконує максимізацію значення даної функції. Оскільки задача кластеризації є NP-складною і для отримання ідеального розв’язку у відповідності до заданого критерію потребує повного перебору, то можна припустити, що ГА буде ефективно виконувати пошук для оптимізації цільової функції. Крім того, як відомо, ГА дуже ефективно розпаралелюється, що забезпечить високу масштабованість алгоритму. Коректно підібравши параметри ГА, можна досягти найоптимальнішої роботи ГА для будь-якої множини вхідних даних.

У випадку кластеризації двовимірних даних їх можна представити у вигляді точок в координатній системі на площині, що забезпечує високу наглядність та інтуїтивне розуміння очікуваних від правильної кластеризації результатів. Виходячи з візуальної оцінки отриманих результатів, можна відразу дати висновок щодо правильності чи неправильності кластеризації без порівняння числових результатів. Це дозволяє ефективно проводити тестування алгоритму під час розробки та виявляти його особливості.

# Кластерний аналіз

## Постановка задачі

Кластерний аналіз (англ. Data clustering) – задача розбиття заданої вибірки об'єктів (спостережень, даних) на підмножини, що називаються кластерами, так, щоб кожен кластер складався зі схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися. Задача кластеризації відноситься до статистичної обробки, а також до широкого класу задач навчання без учителя.

Кластерний аналіз – це багатовимірна статистична процедура, що виконує збір даних, які містять інформацію про вибірку об'єктів, сортування об'єктів в порівняно однорідні групи (кластери) (Q-кластеризація, або Q-техніка, власне кластерний аналіз) [2].

Кластер – група елементів, які характеризуються загальною властивістю, головна мета кластерного аналізу – знаходження груп схожих об'єктів у вибірці.

Формальна постановка задачі кластеризації наступна. Нехай задана множина спостережень та скінченна вибірка об’єктів

.

Необхідно розбити вибірку на підмножини, що не перетинаються, – кластери таким чином, щоб забезпечити екстремум (мінімум або максимум) деякого критерію (функціонала якості), тобто:

.

Алгоритм кластеризації – це функція , яка будь-якому об’єкту ставить у відповідність кластер .

Типи вхідних даних:

1. Опис об'єктів (спостережень) за ознаками. Кожен об'єкт описується набором своїх характеристик, які називаються ознаками. Ознаки можуть бути числовими або нечисловими. Таким чином, множина вхідних спостережень є набором векторів у m-вимірному просторі ознак.
2. Матриця відстаней між об'єктами. Кожен об'єкт описується відстанями до всіх інших об'єктів навчальної вибірки.

Кластеризація ([навчання без вчителя](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B5%D0%B7_%D1%83%D1%87%D0%B8%D1%82%D0%B5%D0%BB%D1%8F)) відрізняється від класифікації ([навчання з](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81_%D1%83%D1%87%D0%B8%D1%82%D0%B5%D0%BB%D0%B5%D0%BC) вчителем) тим, що мітки вихідних об’єктів  заздалегідь не задані, і навіть може бути невідомою сама множина .

Вирішення задачі кластеризації принципово неоднозначне, і цьому є декілька причин:

* Не існує однозначно найкращого критерію якості кластеризації. Відомий цілий ряд евристичних критеріїв, а також ряд алгоритмів, які не мають чітко вираженого критерію, але здійснюють достатньо розумну кластеризацію «з побудови». Всі вони можуть давати різні результати.
* Число кластерів може бути невідоме заздалегідь і встановлюється відповідно з деяким суб'єктивним критерієм.
* Результат кластеризації істотно залежить від метрики, вибір якої, як правило, також суб'єктивний і визначається експертом.

## Застосування кластеризації

Техніка кластеризації застосовується в найрізноманітніших галузях. Хартіган (Hartigan, 1975) дав всеохоплюючий огляд багатьох опублікованих досліджень, що містять результати, отримані методами кластерного аналізу [3].

Кластерний аналіз виконує такі основні завдання:

* розробка типології або класифікації;
* дослідження корисних концептуальних схем групування об'єктів;
* породження гіпотез на основі дослідження даних;
* перевірка гіпотез або дослідження для визначення, чи дійсно типи (групи), виділені тим або іншим способом, присутні у наявних даних.

Широке застосування має кластеризація в **біології**:

* в області екології рослин та тварин кластеризація застосовується для опису і здійснення просторових та часових порівнянь общин (зборів) організмів в неоднорідних середовищах; вона також застосовується у систематизації рослин для генерації штучних філогеній або кластерів організмів (індивідів) у видах, класах або вищих рівнях, які містять множину атрибутів;
* в області обчислювальної біології та біоінформатиці:
  + в транскриптоміці кластеризація використовується для побудови груп генів з вираженими родинними моделями (також відомі як співвиражені гени). Часто такі групи містять функціонально пов’язані білки, такі як ферменти специфічної структури, або гени, що співрегулюються. Високопродуктивні експерименти, використовуючи метод вираження послідовностей маркерів (expressed sequence tags, ESTs) або мікромасиви ДНК, можуть бути потужним інструментом для визначення геномів, основного аспекту геноміки;
  + в аналізі послідовностей кластеризація використовується для групування гомологічних послідовностей в родини генів. Це дуже важлива концепція в біоінформатиці та еволюційній біології взагалі;
  + у високопродуктивних платформах генотипування алгоритми кластеризації використовується для автоматичного встановлення генотипів;
* в кількісних взаємозв’язках між структурою і активністю (Quantitative Structure Activity Relationships, QSCAR), дослідженнях молекулярного моделювання, а також хемоінформатиці.

В області **медицини** кластеризація захворювань, лікування захворювань або симптомів захворювань приводить до широко використовуваних таксономій. У галузі психіатрії правильна діагностика кластерів симптомів, таких як параноя, шизофренія і т.д., являється вирішальною для успішної терапії.

В **медичній візуалізації**, на зразок магнітно-резонансної томографії, кластерний аналіз може бути використаний для диференціювання між різними типами тканин і крові у тривимірному зображенні. У такій програмі поточне положення не має значення, але вексельна інтенсивність розглядається як вектор з розмірністю для кожного зображення, яке було отримане протягом часу. Ця техніка дозволяє, наприклад, точно виміряти величину радіоактивності шляхом розташування індикатора біля необхідної зони, що усуває необхідність надання окремих проб артеріальної крові.

Кластерний аналіз широко застосовується в **маркетингових дослідженнях** під час роботи с багатовимірними даними за результатами опитувань та випробувальних стендів. Маркетингові дослідники використовують кластерний аналіз для розбиття основного населення споживачів на маркетингові сегменти і для кращого розуміння відношень між різними групами споживачів і потенціальних клієнтів.

Кластеризація в маркетингових дослідженнях також застосовується для:

* сегментації ринку та визначення цільових ринків;
* позиціонування продукту;
* розробки нового продукту;
* вибору ринків для перевірки (експериментальна техніка).

У дослідженнях **соціальних мереж** кластеризація застосовується для визначення общин.

Кластеризація корисна в **еволюції програмного забезпечення**, так як допомагає зменшити залишкові властивості у коді шляхом реформування функціональності, яка стала розсіяною. Це являється однією з форм реструктуризації і, як наслідок, способом безпосередньої профілактики обслуговування.

Кластеризація може використовуватись для **сегментації зображень**, а саме для розділення цифрового зображення на області з метою визначення границь та розпізнавання об’єктів.

В процесі **інтелектуального групування файлів** та веб-сайтів за замістом кластеризація використовується для створення більш доречної множини результатів пошуку в порівнянні зі звичайними пошуковими системами. На даний час у всесвітній мережі працюють інструменти для web-кластеризації, такі як Сlusty, Nigma.

Кластеризація використовується для **групування торгових пунктів**, наявних у мережі web у множини за унікальністю продуктів.

У **аналізі злочинності** кластерний аналіз застосовуються для визначення областей, де трапляється найбільше випадків окремих видів злочинів. Виявивши ці окремі райони або «гарячі точки», де аналогічні злочини траплялися протягом певного періоду, можна використовувати більш ефективно ресурси правоохоронних органів.

## Відомі методи розв’язку

Більшість методів кластерного аналізу, які використовуються, можна розділити на дві всеохоплюючі групи:

* ієрархічні (hierarchical);
* неієрархічні (partitional).

Кожна з груп включає набір підходів і алгоритмів (рис.  1).

Використовуючи різні методи кластерного аналізу, аналітик може отримати різні розв’язки для одних і тих же даних. Це вважається нормальним явищем [4].

Суть **ієрархічної кластеризації** полягає в послідовному об'єднанні менших кластерів у великі, або поділі великих кластерів на менші.

Рис.  1. Класифікація алгоритмів та методів кластеризації

Ієрархічні **агломеративні** методи (Agglomerative Nesting, AGNES) характеризуються послідовним об'єднанням початкових елементів та відповідним зменшенням числа кластерів. На початку роботи алгоритму всі об'єкти є окремими кластерами. На першому кроці найбільш схожі об'єкти об'єднуються в кластер. На наступних кроках об'єднання продовжується до тих пір, поки всі об'єкти не будуть складати один кластер.

Ієрархічні **дивізимні** (ділимі) методи (DIvisive ANAlysis, DIANA) є логічною протилежністю агломеративних методів. На початку роботи алгоритму всі об'єкти належать одному кластеру, який на наступних кроках ділиться на менші кластери, в результаті утворюється послідовність розщеплених груп.

Ієрархічні методи кластеризації розрізняються правилами побудови кластерів. В якості правил виступають критерії, які використовуються при вирішенні питання «схожості» об'єктів при їх об'єднанні в групу (агломеративні методи) або поділу на групи (дивізимні методи).

Ієрархічні методи кластерного аналізу використовуються при невеликих обсягах наборів даних. Перевагою ієрархічних методів кластеризації є їх наочність.

Ієрархічні алгоритми пов'язані з побудовою дендрограм (від грецького dendron — «дерево»), які є результатом ієрархічного кластерного аналізу. Дендрограма описує близькість окремих точок і кластерів один до одного (рис.  2), представляє в графічному вигляді послідовність об'єднання (поділу) кластерів. Дендрограма (dendrogram) — деревоподібна діаграма, що містить *n* рівнів, кожен з яких відповідає одному із кроків процесу послідовного укрупнення кластерів. Дендрограму також називають деревоподібною схемою, деревом об'єднання кластерів, деревом ієрархічної структури. Дендрограма представляє собою вкладене угруповання об'єктів, яке змінюється на різних рівнях ієрархії.

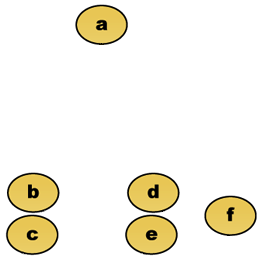
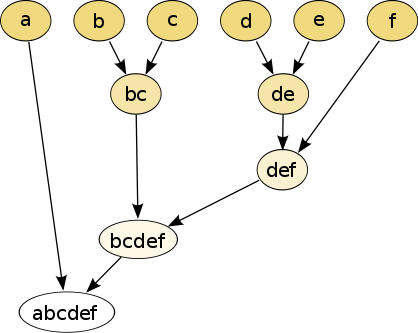
 

Рис.  2. Вихідна множина об'єктів та дендрограма її ієрархічної кластеризації

**Алгоритм k-середніх** (k-means) найбільш поширений серед неієрархічних методів, також званий швидким кластерним аналізом. Повний опис алгоритму можна знайти в роботі Хартігана і Вонга (Hartigan and Wong, 1978). На відміну від ієрархічних методів, які не вимагають попередніх припущень щодо числа кластерів, для можливості використання цього методу необхідно мати гіпотезу про найбільш ймовірну кількість кластерів. Алгоритм k-середніх будує *k* кластерів, розташованих на якомога великих відстанях один від одного. Основний тип завдань, які вирішує алгоритм k-середніх, — наявність припущень (гіпотез) щодо числа кластерів, при цьому вони повинні бути різні настільки, наскільки це можливо. Вибір числа k може базуватися на результатах попередніх досліджень, теоретичних міркуваннях або інтуїції. Загальна ідея алгоритму наступна. Задане фіксоване число *k* кластерів. Спостереження співставляються кластерам так, що середні в кластері (для всіх змінних) максимально можливо відрізняються один від одного.

Опис алгоритму:

1. Первинний розподіл об'єктів по кластерах.

Вибирається число *k*, і на першому кроці ці точки вважаються «центрами» кластерів. Кожному кластеру відповідає один центр.

Вибір початкових центроїдів може здійснюватися таким чином:

* вибір k-спостережень для максимізації початкової відстані;
* випадковий вибір k-спостережень;
* вибір перших k-спостережень.

У результаті кожен об'єкт призначений певному кластеру.

1. Ітеративний процес.

Обчислюються центри кластерів, якими потім і далі вважаються координатні середні кластерів. Об'єкти знову перерозподіляються.

Процес обчислення центрів і перерозподілу об'єктів продовжується до тих пір, поки не буде виконана одна з умов:

* кластерні центри стабілізувалися, тобто всі спостереження належать кластеру, якому належали до поточної ітерації;
* кількість ітерацій досягла максимального числа ітерацій.

Після отримання результатів кластерного аналізу методом k-середніх слід перевірити правильність кластеризації (тобто оцінити, наскільки кластери відрізняються один від одного). Для цього розраховуються середні значення для кожного кластеру. При якісній кластеризації мають бути отримані дуже розбіжні середні для всіх вимірів або хоча б переважної їх частини.

Переваги алгоритму k-середніх:

* простота використання;
* висока швидкодія;
* зрозумілість та прозорість алгоритму.

Недоліки алгоритму k-середніх:

* алгоритм надто чутливий до викидів, які можуть спотворювати середні. Можливим вирішенням цієї проблеми являється використання модифікації алгоритму – алгоритму k-медіани;
* алгоритм може повільно працювати на великих базах даних. Можливим вирішенням цієї проблеми являється використання вибірки даних.

**Алгоритм PAM** (Partitioning Around Medoids) є модифікацією алгоритму k-середніх, алгоритмом k-медіани (k-medoids). Алгоритм менш чутливий до шумів і викидів даних, ніж алгоритм k-means, оскільки медіана менш схильна до впливів викидів. PAM ефективний для невеликих баз даних, але його не слід використовувати для надвеликих наборів даних.

**Алгоритм BIRCH** (Balanced Iterative Reducing and Clustering using Hierarchies) запропонований Тьян Зангом і його колегами. Завдяки узагальненим уявленням кластерів швидкість кластеризації збільшується, алгоритм при цьому володіє високим масштабуванням. У цьому алгоритмі реалізований двоетапний процес кластеризації. У ході першого етапу формується попередній набір кластерів. На другому етапі до виявлених кластерам застосовуються інші алгоритми кластеризації – придатні для роботи в оперативній пам'яті. Якщо кожен елемент даних уявити собі як намистину, що лежить на поверхні столу, то кластери намистин можна «замінити» тенісними кульками і перейти до більш детального вивчення кластерів тенісних кульок. Число намистин може виявитися досить великим, однак діаметр тенісних кульок можна підібрати таким чином, щоб на другому етапі можна було, застосувавши традиційні алгоритми кластеризації, визначити дійсну складну форму кластерів.

**Алгоритм WaveCluster** представляє собою алгоритм кластеризації на основі хвильових перетворень. На початку роботи алгоритму дані узагальнюються шляхом накладення на простір даних багатовимірної решітки. На подальших кроках алгоритму аналізуються не окремі точки, а узагальнені характеристики точок, що потрапили в одну клітинку решітки. У результаті такого узагальнення необхідна інформація вміщується в оперативній пам'яті. На наступних кроках для визначення кластерів алгоритм застосовує хвильове перетворення до узагальнених даних.

Головні особливості WaveCluster:

* складність реалізації;
* алгоритм може виявляти кластери довільних форм;
* алгоритм не чутливий до шумів;
* алгоритм застосовується тільки до даних низької розмірності.

**Алгоритм CLARA** (Clustering LARge Applications) був розроблений Kaufmann і Rousseeuw в 1990 році для кластеризації даних у великих базах даних. Даний алгоритм використовується в статистичних аналітичних пакетах, наприклад, таких як S+. Алгоритм CLARA витягує множину зразків з бази даних. Кластеризація застосовується до кожного із зразків, на виході алгоритму пропонується краща кластеризація. Для великих баз даних цей алгоритм ефективніший, ніж алгоритм PAM. Ефективність алгоритму залежить від набору даних, обраного в якості зразка. Якісна кластеризація на вибраному наборі може не дати достатньо якісну кластеризацію на всій множині даних.

**Алгоритм Clarans** (Clustering Large Applications based upon RANdomized Search) формулює завдання кластеризації як випадковий пошук у графі. У результаті роботи цього алгоритму сукупність вузлів графа являє собою розбиття множини даних на число кластерів, задане користувачем. «Якість» отриманих кластерів визначається за допомогою критеріальної функції. Алгоритм Clarans сортує всі можливі розбиття множини даних у пошуках прийнятного розв’язку. Пошук розв’язку зупиняється в тому вузлі, де досягається мінімум серед зумовленого числа локальних мінімумів. Серед нових масштабованих алгоритмів також можна відзначити алгоритм **CURE** – алгоритм ієрархічної кластеризації, і алгоритм **DBScan**, де поняття кластеру формулюється з використанням концепції щільності (density). Основним недоліком алгоритмів BIRCH, Clarans, CURE, DBScan є та обставина, що вони вимагають задання деяких порогів щільності точок, а це не завжди прийнятно. Ці обмеження зумовлені тим, що описані алгоритми орієнтовані на надвеликі бази даних і не можуть використовувати значні обчислювальні ресурси.

Над масштабованими методами активно працюють багато дослідників, основне завдання яких – подолати недоліки алгоритмів, існуючих на сьогоднішній день.

Для обчислення відстані між об'єктами використовуються різні міри схожості (міри подібності), звані також метриками або функціями відстаней.

Для надання великих ваг більш віддаленим один від одного об'єктам можна скористатися квадратом **Евклідової відстані** шляхом зведення в квадрат стандартної Евклідової відстані.

**Манхеттенська відстань** (відстань міських кварталів), також звана «сіті-блок» відстанню, розраховується як середнє різниць за координатами. У більшості випадків ця міра відстані призводить до результатів, подібних за розрахунками до відстані Евкліда. Однак, для цієї міри вплив окремих викидів менший, ніж при використанні квадратичної Евклідової відстані, оскільки тут координати не зводяться в квадрат.

**Відстань Чебишева** варто використовувати, коли необхідно визначити два об'єкти як «різні», якщо вони відрізняються по якомусь одному вимірюванню. Відсоток незгоди обчислюється, якщо дані є категоріальними.

Коли кожен об'єкт представляє собою окремий кластер, відстані між цими об'єктами визначаються обраною метрикою. Виникає наступна задача – визначення відстані між кластерами. Існують різні правила, звані методами об'єднання або зв'язку для двох кластерів.

**Метод ближнього сусіда** або одиночний зв'язок – відстань між двома кластерами визначається відстанню між двома найбільш близькими об'єктами (найближчими сусідами) в різних кластерах. Цей метод дозволяє виділяти кластери як завгодно складної форми за умови, що різні частини таких кластерів з'єднані ланцюжками близьких один до одного елементів. У результаті роботи цього методу кластери представляються довгими «ланцюжками» або «волокнистими» кластерами, «зчепленими разом» тільки окремими елементами, які випадково опинилися ближче інших один до одного.

**Метод найбільш віддалених сусідів** або повний зв'язок – відстані між кластерами визначаються найбільшою відстанню між будь-якими двома об'єктами в різних кластерах (тобто «найбільш віддаленими сусідами»). Метод добре використовувати, коли об'єкти дійсно походять з різних «гаїв». Якщо ж кластери мають у деякому роді видовжену форму або їх природний тип є «ланцюжковим», то цей метод не слід використовувати.

**Метод Варда** (Ward's method) – в якості відстані між кластерами береться приріст суми квадратів відстаней об'єктів до центрів кластерів, що отримується в результаті їх об'єднання (Ward, 1963). На відміну від інших методів кластерного аналізу, для оцінки відстаней між кластерами тут використовуються методи дисперсійного аналізу. На кожному кроці алгоритму об'єднуються такі два кластери, які призводять до мінімального збільшення цільової функції, тобто внутрішньогрупової суми квадратів. Цей метод направлений на об'єднання близько розташованих кластерів і «прагне» створювати кластери малого розміру.

**Метод незваженого попарного середнього** (метод невиваженого попарного арифметичного середнього – Unweighted Pair-Group Method using Arithmetic averages, UPGMA (Sneath, Sokal, 1973)) – в якості відстані між двома кластерами береться середня відстань між усіма парами об'єктів в них. Цей метод слід використовувати, якщо об'єкти дійсно походять з різних «гаїв», у випадках присутності кластерів «ланцюгового» типу, при припущенні нерівних розмірів кластерів.

**Метод зваженого попарного середнього** (метод зваженого попарного арифметичного середнього – Weighted Pair-Group Method using Arithmetic averages, WPGMA (Sneath, Sokal, 1973)) – метод, схожий на метод незваженого попарного середнього, різниця полягає лише в тому, що тут у якості вагового коефіцієнта використовується розмір кластера (число об'єктів, що містяться в кластері). Цей метод рекомендується використовувати саме за наявності припущення про кластери різних розмірів.

**Незважений центроїдний** метод (метод незваженого попарного центроїдного усереднення – Unweighted Pair-Group Method using the Centroid average, UPGMC (Sneath and Sokal, 1973)) – в якості відстані між двома кластерами в цьому методі береться відстань між їх центрами тяжіння.

**Зважений центроїдний** метод (метод зваженого попарного центроїдного усереднення – Weighted Pair-Group Method using the Centroid average, WPGMC (Sneath, Sokal 1973)) – схожий на попередній, відмінність полягає в тому, що для врахування різниці між розмірами кластерів (числа об'єктів в них), використовуються ваги. Цей метод переважно застосовується у випадках, якщо є припущення щодо істотних відмінностей у розмірах кластерів.

# Еволюційні обчислення

В комп’ютерних науках еволюційні обчислення є галуззю штучного інтелекту (зокрема обчислювального інтелекту), яка включає задачі комбінаторної оптимізації. Еволюційні обчислення використовують ітеративний прогрес, такий як зростання розвитку в популяції. Ця популяція надалі відбирається шляхом керованого випадкового пошуку, використовуючи паралельну обробку для досягнення бажаної мети. Такі процеси часто запозичені з біологічних механізмів еволюції.

## Еволюційні алгоритми

В області штучного інтелекту еволюційні алгоритми (ЕА) є підмножиною еволюційних обчислень [5]. Це загальні метаевристичні алгоритми оптимізації, що базуються на популяціях. ЕА використовують механізми, запозичені з біологічної еволюції: репродукція, мутація, рекомбінація і селекція. Розв’язки-кандидати для задачі оптимізації грають роль індивідів у популяції. А функція пристосованості визначає середовище, в якому індивіди «живуть». Еволюція популяції відбувається після повторення алгоритмом зазначених вище операцій. Штучна еволюція (ШЕ) описує процес за участю окремих еволюційних алгоритмів. ЕА – це окремі компоненти, які приймають участь у ШЕ.

Еволюційні алгоритми часто показують високонаближені розв’язки до всіх видів задач, оскільки вони в ідеалі не роблять жодних припущень щодо закладеного цільового середовища. Це в основному успішно показано в таких різних напрямках як інженерія, мистецтво, біологія, економіка, маркетинг, генетика, дослідження операцій, робототехніка, соціальні науки, фізика, політика та хімія.

Крім використання в якості математичної оптимізації еволюційні обчислення і алгоритми також були використані як експериментальне середовище для затвердження теорій про біологічну еволюцію та природну селекцію, частково у роботах в області штучного життя.

Реалізація біологічного процесу здійснюється наступним чином. Зазвичай перше покоління охоплює початкову популяцію випадковим чином згенерованих кандидатів на розв’язок. Функція пристосованості застосовується до розв’язків-кандидатів і наступних поколінь.

Внаслідок селекції батьки для нового покоління вибираються з ухилом у бік більш високої функції пристосованості. Батьки породжують одного або двох нащадків (нових кандидатів) шляхом копіювання своїх генів з двома можливими змінами: кроссовер (схрещування, рекомбінація) рекомбінує батьківські гени, мутація призводить до зміни генотипу індивіда випадковим чином. Ці нові кандидати конкурують зі старими кандидатами на місця в наступному поколінні (виживання найбільш пристосованих).

Описаний процес може бути повторений до тих пір, поки кандидат з достатньою якістю (розв’язку) не буде знайдений або заздалегідь визначений ліміт не буде досягнутий.

Подібні методи відрізняються за деталями в реалізації в залежності від характеру конкретної прикладної задачі:

* генетичний алгоритм – це найбільш популярній тип ЕА. Пошук вирішення задачі відбувається у формі рядків чисел (зазвичай двійкових, хоча найкраще кодування зазвичай те, яке відображає щось із змісту розв’язуваної задачі). Цей тип ЕА зазвичай використовується в задачах оптимізації;
* генетичне програмування – розв’язки представляються у формі комп’ютерних програм, а їх пристосованість визначається їхньою здатністю вирішувати обчислювальні програми;
* еволюційне програмування – подібне генетичному програмуванню, але структура програми фіксована, а її числові параметри схильні до еволюції;
* еволюційна стратегія – працює з векторами або з дійсними числами як поданнями розв’язку і частково використовує самоадаптаційну мутацію.

## Генетичні алгоритми

Генетичний алгоритм – це техніка пошуку, яка використовується в обчисленнях для пошуку точного або наближеного вирішення задач оптимізації або пошуку. Генетичні алгоритми класифікуються як евристичні методи глобального пошуку. Генетичні алгоритми є різновидом еволюційних алгоритмів, які використовують техніку, запозичену з еволюційної біології, таку як спадковість, мінливість та схрещування.

Генетичні алгоритми реалізуються в комп’ютерному моделюванні, де популяція абстрактного подання (хромосоми або генотипи геномів) розв’язків-кандидатів (індивіди, створіння або фенотипи) для задачі оптимізації еволюціонує у напрямку кращого вирішення. Зазвичай розв’язки подаються у вигляді двійкових рядків нулів та одиниць, але можливе також інше кодування. Еволюція зазвичай стартує з популяції випадково згенерованих індивідів і відбувається з поколіннями. В кожному поколінні обчислюється функція пристосованості кожного індивіда, стохастично обирається множина індивідів з поточної популяції (на основі їх пристосованості) та модифікується (рекомбінується і, можливо, мутує) для формування нової популяції. Якщо алгоритм завершився у зв’язку з максимальним числом поколінь, прийнятний розв’язок може бути знайдено або не знайдено.

Генетичні алгоритми знаходять застосування в біоінформатиці, філогенетиці, обчислювальних методах, інженерії, економіці, хімії, виробництві, математиці, фізиці та інших галузях.

Типовий генетичний алгоритм потребує:

1. Генетичного подання розв’язку задачі.
2. Функцію пристосованості для обчислення прийнятності розв’язку.

Стандартне подання розв’язку – масив бітів. Масиви або інші типи структур можуть по суті бути використані таким же шляхом. Основна властивість, яка робить такі генетичні подання придатними, це можливість їх частин легко бути відрегульованими до фіксованих розмірів, що сприяє простим операціям схрещування. Змінне подання довжини також може бути використане, але в такому випадку реалізація схрещування є складнішою. Каскадні подання, розроблені в генетичному програмуванні і поданні графів, також розроблені в еволюційному програмуванні.

Функція пристосованості, визначена для генетичного подання, вимірює якість поданого розв’язку. Функція пристосованості завжди залежить від задачі.

Після того, як були визначені генетичне подання та функція пристосованості, ГА переходить до ініціалізації популяції розв’язків випадковим чином, далі вдосконалює її шляхом повторного застосування операцій мутації, схрещування, інверсії та селекції.

**Ініціалізація.** Спочатку багато окремих розв’язків випадково генеруються для формування початкової популяції. Розмір популяції залежить від характеру задачі, але зазвичай включає декілька сотень або тисяч можливих розв’язків. Традиційно популяція генерується випадковим чином, покриваючи весь спектр можливих розв’язків (пошуковий простір). У деяких випадках розв’язки можуть бути «розсіяні» в областях, де, ймовірно, будуть знайдені оптимальні розв’язки.

**Селекція**. З кожним наступним поколінням частина наявної популяції обирається для породження нового покоління. Окремі розв’язки обираються через процес, що базується на пристосованості, в якому більш пристосовані індивіди (за значенням функції пристосованості), як правило, будуть обрані з більшою ймовірністю. Деякі методи оцінюють пристосованість кожного розв’язку і переважно обирають найкращі з них. Інші методи оцінюють лише випадкові зразки популяції, оскільки цей процес може бути довготривалим. Це дає можливість підтримувати широку різноманітність популяції, попереджуючи передчасну збіжність до «слабких» розв’язків. Популярні добре вивчені методи селекції включають селекцію за допомогою колеса рулетки та турнірну селекцію.

**Репродукція**. Наступний крок для генерації іншої популяції розв’язків з обраних внаслідок селекції – застосування генетичних операції: рекомбінації та мутації. Для створення кожного нового розв’язку з відібраного пулу береться пара «батьків» для породження нащадка. Внаслідок породження «розв’язку-нащадка», використовуючи зазначені вище методи схрещування і мутації, створюється новий розв’язок, який зазвичай представляє багато властивостей його «батьків». Нові батьки обираються для кожного нового нащадка і процес продовжується до тих пір, поки нова популяція розв’язків відповідного розміру не буде згенерована. Хоча методи репродукції, які базуються на використанні двох батьків, є більш біологічно запозичені, деякі дослідження пропонують більше, ніж двох «батьків» для репродукції з метою отримання хромосом вищої якості.

Ці процеси, в кінцевому рахунку, призводить до нового покоління популяції хромосом, яке відрізняється від початкового покоління. В основному середня пристосованість має зрости від цієї процедури над популяцією, оскільки тільки найкращі організми обираються з першого покоління для розмноження, а також обирається невелика частка менш пристосованих розв’язків із зазначених вище причин.

**Критерій зупинки**. Процеси над поколіннями повторюються до тих пір, поки не буде виконуватись досягнута умова зупинки. Основними критеріями зупинки є наступні:

* знайдений розв’язок задовольняє критерій достатнього мінімуму;
* задане число поколінь досягнуте;
* виділений бюджет (час обчислень або якість) досягнутий;
* найвищий рівень пристосованості серед індивідів досягає або досяг такого плато, що послідовні ітерації більше не дають кращих результатів;
* ручна ревізія;
* комбінація перерахованих вище.

Generate a population of chromosomes of size N:

s1, s2, …, sN

Calculate the fitness of each chromosome:

f(s1), f(s2), …, f(sN)

Is the termination criterion satisfied?

Select a pair of chromosomes for mating

With the crossover probability pc, exchange parts of the two selected chromosomes and create two offspring

With the mutation probability pm, randomly change the gene values in the two offspring chromosomes

Place the resulting chromosomes in the new population

Is the size of the new population equal to N?

Replace the current chromosome population with the new population

Stop

Yes

Yes

No

No

Start

Рис.  3. Блок-схема канонічного генетичного алгоритму

Спрощений псевдокод генетичного алгоритму (рис.  3):

1. Створити початкову популяцію індивідів.
2. Обчислити пристосованість кожного індивіда в популяції.
3. Повторювати для поточного покоління до зупинки (ліміт фітнесу, прийнятна пристосованість отримана і т.д.):
   1. Обрати найпристосованіших індивідів для репродукції.
   2. Створити нові індивіди шляхом операцій схрещування і мутації, щоб дати народження новому потомству.
   3. Обчислити окремі значення пристосованості для нових індивідів.
   4. Замінити найменш пристосовані індивіди на нові.

# Розробка генетичного алгоритму кластеризації

## Загальний принцип роботи

Для виконання кластеризації було запропоновано підхід, коли ГА працює поетапно (рис.  4). На кожному етапі роботи ГА групує і повертає один кластер. На початку роботи, тобто на першому етапі, на вхід ГА подається вся множина об’єктів (спостережень). Алгоритм із всієї вхідної вибірки об’єктів здійснює пошук одного першого кластеру. Коли із вхідної множини об’єктів згруповано підмножину, тобто коли кластер знайдено, він заноситься до списку кластерів, а також знаходиться різниця вихідної множини та множини об’єктів, що представляє знайдений кластер. На другому етапі в якості вхідної множини об’єктів використовується отримана різниця множин, тобто кластеризація продовжується для множини об’єктів, що залишилася некластеризованою після першого етапу. Даний процес виконується до тих пір, поки вся множина об’єктів не буде вичерпаною. Алгоритм завершується, якщо вся множина є кластеризованою, тобто якщо кожному об’єкту поставлено у відповідність кластер, або якщо кількість об’єктів, що залишилася, є меншою, ніж мінімальний розмір кластеру. У другому випадку число об’єктів залишається некластеризованим, тобто об’єкту не поставлено у відповідність жодного кластеру. Така ситуація означає, що дані об’єкти є нетиповими, тобто їх не можна приєднати до жодного з кластерів, або можна приєднати до декількох кластерів одночасно.

Апріорними вхідними даними алгоритму є розміри кластерів. Ці дані можуть бути вказані одним із наступних способів:

1. Розмір усіх кластерів є однаковим і задається конкретно. Якщо потужність вхідної множини не ділиться націло на розмір кластеру, то залишиться число некластеризованих об’єктів, яке дорівнює залишку від ділення потужності вхідної множини на розмір кластеру. Як зазначалося раніше, ці об’єкти можуть розглядатися як нетипові.
2. Задається кількість кластерів. У такому випадку розраховується розмір всіх кластерів шляхом ділення потужності вхідної множини на задану кількість. Даний спосіб є аналогією попереднього, оскільки всі кластери також матимуть однаковий розмір, проте максимально можливе число некластирозованих об’єктів, що може залишитися, буде меншим, ніж задана кількість кластерів.
3. Задаються окремі розміри для кожного з кластерів. При такому способі задані розміри кластерів спочатку сортуються в порядку незростання. Далі виконується пошук і групування кластеру, який має найбільший розмір, тобто перший розмір у відсортованому списку. Пошук кожного наступного кластера здійснюється до тих пір, поки список не буде вичерпаний або поки не буде кластеризованою вся множина. Число об’єктів, що можуть залишитися некластеризованими, не перевищить мінімального з заданих розмірів кластерів. Сума всіх заданих розмірів кластерів має не перевищувати потужність початкової вхідної множини об’єктів. В залежності від реалізації алгоритму, кластеризація може продовжуватись за рахунок подальшого пошуку кластерів найменшого розміру після вичерпання всіх розмірів.
4. Задається діапазон у вигляді верхньої та нижньої границі розміру усіх кластерів або кожного кластеру окремо. В такому випадку ГА намагається автоматично підібрати кластер найбільш оптимального розміру в заданих межах. Такий підхід є необхідним, якщо розмір кластеру невідомий точно, а лише приблизно, проте вимагає розробки складної функції пристосованості і може дати менш якісні результати зі збільшенням різниці між верхньою та нижньою границями розміру.

Input( X, |X| = N )

Input( {|Si|}, i=1, 2,…, k )

j := 1

Sort( {|Si|} : |Si| ≥ |Si+1| )

Sj := FindCluster( X, |Sj| )

X := X / Sj

j := j + 1

j > k **OR** |X| ≤ |Sj|

Output( {Si}, i=1, 2,…, k }

Output( X )

Stop

No

Yes

Start

Рис.  4. Блок-схема узагальненого генетичного алгоритму кластеризації

Практично усі з відомих алгоритмів кластеризації потребують введення деяких вхідних даних апріорі. Такими даними можуть бути як специфічні параметри для конкретного алгоритму (кількість епох, степеневі коефіцієнти), так і дані, які описують структуру результату [6]. Наприклад, відомий алгоритм k-середніх потребує введення кількості кластерів апріорі, проте розміри кластерів визначаються алгоритмом автоматично. Деякі види ієрархічних алгоритмів потребують введення порогових щільностей об’єктів у кластерах. При різних постановках задач такі вхідні параметри можуть ускладнювати ситуацію, оскільки експерту необхідно визначити ці параметри до початку роботи алгоритму. Проте існує ряд задач, постановка яких вимагає задання цих параметрів вручну. Розроблений генетичний алгоритм кластеризації дозволяє ефективно вирішувати задачу кластеризації для випадку, коли необхідно вказати розміри кластерів, на що не здатен жоден з відомих алгоритмів кластеризації.

У загальному випадку, при заданій вхідній множині потужністю N та розміром S кожного з кластерів, складність комбінаторної задачі, що вирішується у роботі, становить

Якщо задано розміри кластерів , при чому для будь-яких , , то складність задачі становить

Далі будуть описані генетичні операції, які працюють на кожному етапі кластеризації, тобто в процесі пошуку одного поточного кластеру.

## Кодування геному

Кожна хромосома представляє собою один кластер. Оскільки порядок об’єктів у межах одного кластеру не має значення, то порядок елементів у хромосомі також є відсутнім. Інакше кажучи, хромосома представляє собою множину. Таким чином, популяція представляє також множину можливих кластерів, які є розв’язками-кандидатами для поточного етапу роботи ГА. Якщо на поточному етапі здійснюється пошук кластера, розмір якого заданий в діапазоні верхньої та нижньої границі, то в межах популяції можуть співіснувати хромосоми-кластери різного розміру.

Такий підхід до кодування геному суттєво відрізняється від класичного, коли хромосома є рядком бітів або інших змінних. Даний вибір зумовлений специфікою задачі, що вирішується за допомогою ГА. Оскільки вхідні дані, вихідні результати та основні терміни у постановці задачі базуються на множинах, то доцільно використовувати множини при кодуванні геному.

Даний підхід також накладає додаткову умову на формування нових хромосом та зміну сформованих, а саме виникає необхідність уникання конфліктних ситуацій. Конфліктна ситуація виникає у випадку, якщо посилання на один і той самий об’єкт зустрічається в хромосомі більше, ніж один раз.

При реалізації даного кодування геному хромосома може бути описана як масив посилань на об’єкти (спостереження) кластеру, проте порядок елементів у масиві значення не має. Суттєва лише наявність однократного посилання на об’єкт в масиві. Якщо вхідна множина всіх об’єктів також зберігається у масиві, який не змінює порядку розташування своїх елементів протягом поточного етапу, то хромосома також може бути реалізована у формі невпорядкованого масиву цілих невід’ємних чисел – індексів об’єктів у основному вхідному масиві всіх об’єктів, що кластеризуються.

## Оцінка пристосованості

Оскільки визначення пристосованості передбачає обчислення міри компактності кластеру відповідно до метрики (способу оцінки подібності) [7], то метрика задається у контексті критерію кластеризації.

Під час розробки алгоритму та для його відлагодження й тестування у якості метрики була використана класична просторова Евклідова відстань, яка обчислюється за формулою

де – пара об’єктів у *m*-вимірному просторі; – їх відповідні проекції. Для реалізації ГА був введений найпоширеніший критерій кластеризації, відповідно до якого максимізується обернене середнє значення відстані об’єктів кластеру до його центру тяжіння:

де – розмір кластеру *S*, – евклідова відстань j-го об’єкта кластеру до центру тяжіння кластеру, радіус-вектор якого обчислюється як фізичний центр маси:

Обчислювальна складність такої фітнес-функції становить

У випадку, коли в популяції передбачаються кластери різних розмірів, необхідна розробка складнішої функції пристосованості, яка б не допускала передчасної збіжності популяції до кластерів мінімального розміру. Така фітнес-функція матиме вищу обчислювальну складність і повинна базуватися на щільності елементів в кластері, щоб «заохочувати» ГА обирати для селекції кластери великих розмірів.

Аналогічним за змістом до даного критерію є мінімізація середньої відстані між будь-якою парою об’єктів кластеру. У загальному випадку, якщо , значення такої функції пристосованості буде обчислюватись як обернене середнє суми відстаней:

Очевидно, що обчислювальна складність такої функції пристосованості буде складати , що сприяє нижчій швидкодії, ніж в попередньому варіанті. Однак існує ряд задач, постановка яких не може дати формулювання центру мас, або для яких центр мас не може бути обчислений принципово. В такому разі заздалегідь, до початку роботи алгоритму кластеризації, будується матриця відстаней (подібностей) між всіма парами об’єктів. При розмірах кластерів до кількох сотень швидкодія такої функції пристосованості не суттєво відрізняється від випадку з центром мас, оскільки на кожній ітерації не потрібно обчислювати відстань від центру маси до об’єкту, а лише звертатися до матриці відстаней.

## Початкова популяція

Початкова популяція генерується випадковим чином, але так, щоб покрити всю вхідну множину об’єктів, що кластеризуються. Початкова популяція містить всі об’єкти заданої вхідної множини хоча б в одному екземплярі в будь-якому з кластерів. Це забезпечує розмаїття можливих варіантів пошуку. Якщо початкова популяція не буде містити деякі об’єкти, а вони мають входити в остаточний результат поточного етапу роботи ГА, то вони можуть бути отримані лише внаслідок мутації, що значно сповільнить роботу алгоритму та його збіжність, а також спричинить низьку якість кластеризації.

Input( X, |X| = N )

Input( |S| )

P := Population Size

i := 1

χ := { Ø }

i > P

|χ| < |S|

Si := { χ[k] : k = 1, 2, …, |S| }

χ := { χ[k] : k = |Si|+1, |Si|+2, …, |χ| }

i := i + 1

Stop

Output( { Si : i = 1, 2, …, P } )

χ := Randomize( X )

Yes

No

No

Yes

Start

Рис.  5. Блок-схема алгоритму генерації початкової популяції

Якщо розміри кластерів задані у межах діапазону від верхньої до нижньої границі, то розмір кожної наступної хромосоми під час створення початкової популяції генерується випадковим чином. Такий підхід забезпечує можливість потенційно отримати результат будь-якого розміру. Кількість хромосом в початковій популяції дорівнює розміру популяції, що задається як параметр роботи алгоритму і не змінюється протягом роботи ГА.

Алгоритм генерації початкової популяції поточного етапу передбачає наступні дії (рис.  5):

1. Створити копію масиву вхідної вибірки об’єктів (спостережень).
2. Відсортувати випадковим чином копію масиву вхідної вибірки.
3. Видалити з відсортованої копії вибірки перші елементів і створити з них окремий кластер , де – розмір кластеру, що необхідно отримати. Розмір генерується випадковим чином в межах заданого діапазону.
4. Додати кластер до популяції.
5. Повторити пункти 3÷4 необхідну кількість разів, яка дорівнює розміру популяції. Якщо розмір наявної копії масиву вибірки менший, ніж заданий розмір кластеру , повторити кроки 1÷2.

## Селекція

Для виконання генетичних операцій рекомбінації та мутації селекція виконується за допомогою «колеса рулетки». Селекція обирає для схрещування більш пристосованих індивідів з більшою ймовірністю. Ймовірність вибору індивіда з поточної популяції прямопропорційна значенню функції пристосованості індивіда. Один і той самий індивід може бути обраний багаторазово для виконання генетичних операцій. Такий підхід забезпечує виживання більш пристосованих особин, але в той же час не виключає виживання найменш пристосованих, що забезпечує різноманіття об’єктів в межах популяції і запобігає передчасній збіжності ГА.

Алгоритм селекції передбачає наступні кроки:

1. Обчислення суми значень функції пристосованості всіх індивідів в межах поточної популяції.
2. Нормалізація значення функції пристосованості для кожного індивіда шляхом її ділення на суму значень пристосованості. Сума нормалізованих значень пристосованості для всіх індивідів в межах популяції має дорівнювати одиниці.
3. Сортування популяції за незростанням значення функції пристосованості.
4. Акумуляція нормалізованих значень пристосованості, тобто акумульоване значення пристосованості кожного наступного індивіда у відсортованому масиві обчислюється як сума власного нормалізованого значення пристосованості та нормалізованого значення пристосованості попереднього індивіда. Акумульоване значення пристосованості останнього індивіда у масиві дорівнює одиниці, а першого – залишається незмінним з попереднього кроку.
5. Генерація випадкового дійсного числа від нуля до одиниці за допомогою генератора псевдовипадкових чисел.
6. Вибір з акумульованого масиву індивідів першого від початку індивіда, акумульоване значення пристосованості якого більше за згенероване випадкове число.

## Рекомбінація

Оскільки геном подається у вигляді множини об’єктів, порядок розташування яких не має значення, то доцільним є кроссовер на основі операцій над множинами (рис.  6). Схрещування є статевим. Пара особин породжує пару нащадків. За допомогою селекції з популяції обирається пара «індивідів-батьків» . Далі виконуються наступні операції:

1. знайти перетин двох множин-батьків ;
2. знайти різниці множин , ;
3. розбити випадковим чином множини і на пари однакових за потужністю підмножин , та , відповідно;
4. створити двох нащадків та , виконавши об’єднання множин

та .

R1A

R2A

I

R1B

R2B

A

B

Рис.  6. Виконання операції кроссоверу над парою множин

Особливість такої реалізації операції рекомбінації полягає в тому, що при схрещенні двох тотожних індивідів буде отримана пара тих самих індивідів, що не зіпсує результат роботи ГА на попередніх епохах. Якщо ж популяція буде містити множини квазіоптимальних розв’язків, індивіди в межах якої відрізняються незначним числом об’єктів, то в результаті їх схрещування також буде отриманий квазіоптимальний розв’язок, ядром якого є перетин множин-батьків. В загальному випадку така реалізація кроссоверу також дозволяє схрещувати батьків різних розмірів, тобто не залежить від розмірів кластерів, при чому отримані розміри кластерів-нащадків будуть знаходить в межах меншого та більшого розмірів батьків.

Розробка описаної операції рекомбінації була одночасно спрямована на вирішення проблеми, пов’язаної з виключенням можливості конфліктів.

## Мутація

В даному алгоритмі мутація використовується для відновлення об’єктів в популяції, які можуть бути втрачені внаслідок заміни старого покоління наступним новим. Мутація виконується з деякою заданою ймовірністю одразу над кожною парою нащадків, що отримується внаслідок схрещування пари батьків. Мутація полягає в заміні одного об’єкта в межах кластеру, вибраного випадковим чином, на інший випадковий об’єкт. Слід зазначити, що операція мутації не повинна призводити до виникнення конфліктів, тобто до заміни новий об’єкт не повинен міститися в поточному кластері. У зв’язку з цим алгоритм виконання мутації наступний:

1. Знайти різницю заданої для кластеризації множини (вибірки) всіх об’єктів та множини об’єктів поточного кластеру.
2. Із поточного кластеру видалити один, вибраний випадковим чином, об’єкт.
3. До поточного кластеру додати один об’єкт, вибраний випадковим чином із різниці множин, отриманої в першому пункті.

У випадку, якщо в популяції співіснують кластери різних розмірів, мутація може змінювати один наявний об’єкт в межах кластеру, додавати один випадковий об’єкт до кластеру або видаляти один об’єкт, обраний випадковим чином. При цьому розмір кластеру не повинен внаслідок мутації виходити за межі заданого діапазону. Видалення випадкового елементу з кластеру може виконуватись лише за умови, що розмір поточного кластеру більший, ніж нижня границя розміру. Додавання випадкового елементу можливе лише за умови, що розмір кластеру менший за верхню границю. У всіх інших випадках виконується лише заміна випадково вибраного об’єкту на інший.

Описаний алгоритм мутації виконує заміну, додавання або видалення лише одного об’єкта кластеру. Даний підхід зумовлений тим, що введення двох і більше нових об’єктів до кластеру може значно знизити значення його функції пристосованості. Внаслідок цього мутований індивід буде з малою ймовірністю обраний для селекції при наступній репродукції, тому мутація в такому випадку не принесе нових об’єктів до популяції.

## Репродукція

Репродукція полягає в заміні старих індивідів поточної популяції на нові, отримані після селекції, схрещування та мутації. Після виконання перерахованих генетичних операцій необхідну кількість разів з поточної популяції видаляються всі індивіди, за винятком невеликого фіксованого числа так званих «елітних» осіб з найвищим значенням функції пристосованості. Число таких осіб визначається коефіцієнтом «елітизму», який задає відсоток «елітних» осіб від загального розміру популяції. На основі коефіцієнту елітизму обчислюється число індивідів, які необхідно замінити на нові. Дане число ділиться на два, внаслідок чого отримується кількість пар, які необхідно обрати для схрещування за допомогою селекції.

Коефіцієнт елітизму введений з метою збереження найкращого розв’язку в популяції після репродукції. Даний підхід гарантує, що якість розв’язку ГА не буде знижуватись у наступних поколіннях [8].

Коефіцієнт елітизму має бути невисоким, оскільки при надто високих коефіцієнтах елітизму ГА може передчасно збігатись до локальних оптимумів.

## Критерій зупинки

Зупинка кожного етапу роботи алгоритму здійснюється, якщо не було зміни значення пристосованості найбільш пристосованого індивіда довше, ніж протягом деякого заданого числа ітерацій (репродукцій) ГА. В програмну модель ГА введено лічильник, який рахує кількість ітерацій ГА, протягом яких якість розв’язку не змінилася. Якщо після чергової ітерації ГА відбулася зміна розв’язку, тобто змінилось значення фітнес-функції найпристосованішого з індивідів, то даний лічильник скидається в нуль. При спрацьовуванні критерію зупинки поточний етап алгоритму завершується і запускається наступний. Чим вище максимальне значення вказаного лічильника, тим вища якість кластеризації, але також і більше часу витрачається на кластеризацію. Дане максимальне значення визначається емпірично з міркувань досягнення такого стану популяції, коли кожна наступна репродукція не дає суттєвого покращення результату, тобто коли схрещування залишається нераціональним, а покращення результату досягається лише за рахунок мутації.

На відміну від класичного критерію зупинки, який спрацьовує після низького відносного підвищення якості результату при репродукції, введений підхід дає ГА можливість продовжувати пошук надалі деякий час після отримання одного й того самого результату. Перевага такого критерію зупинки спостерігається у зв’язку з тим, що одні й ті самі «елітні» індивіди можуть зберігати своє «лідерство» декілька поколінь підряд, після чого внаслідок вдалої селекції пара квазіоптимальних індивідів буде схрещена в оптимальніший розв’язок за всі наявні в популяції.

## Критерій кластеризації

Критерій кластеризації алгоритму задається у формі цільової функції ГА. Шляхом введення в програмну реалізацію функціонального типу або динамічних бібліотек можна задавати необхідний критерій кластеризації у вигляді додаткового параметру. Вся інша реалізація алгоритму при цьому змін не потребує. Для подання критерію кластеризації у формі, придатній для роботи розробленого алгоритму, необхідно задати функцію, яка б повертала вищу числову оцінку для більш прийнятних кластерів. У відповідності до заданого критерію алгоритм може працювати з даними будь-якої розмірності, типу та формату.

Критерій кластеризації, що забезпечує мінімізацію суми відстаней всіх об’єктів в межах кластеру до його центру мас (рис.  7, рис.  8), може бути використаний, наприклад, при організації та прокладанні локальних комп’ютерних мереж в будівлях.

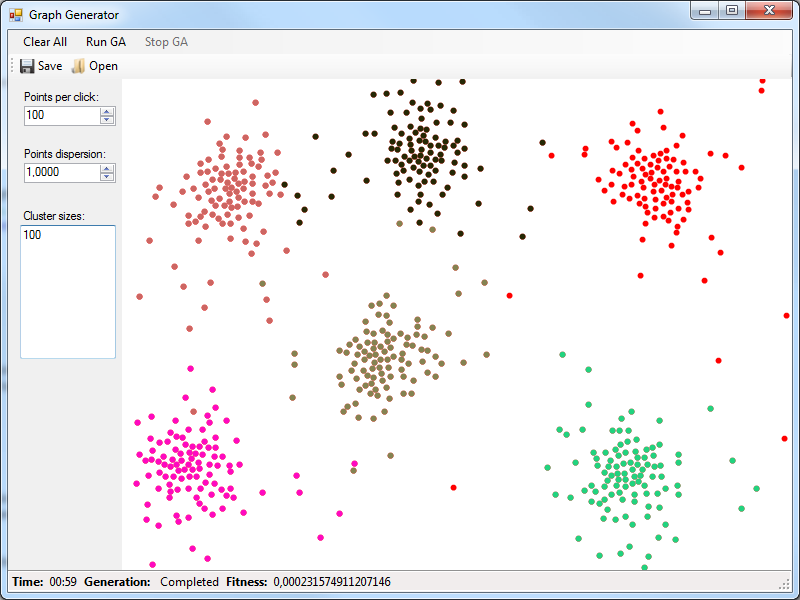


Рис.  7. Кластеризація відповідно до критерію мінімізації відстаней  
об’єктів до центру мас кластеру. Розв’язок є інтуїтивно очевидним

Суть оптимізації ставить за мету використання найменшої сумарної довжини мережевого кабелю. При цьому точки розташування всіх хостів розглядаються як вхідна множина об’єктів, а центри мас представляють місця розташування комутаторів. При цьому необхідним вхідним параметром здійснення такої кластеризації є розміри кластерів, які дорівнюють кількості портів кожного комутатора.

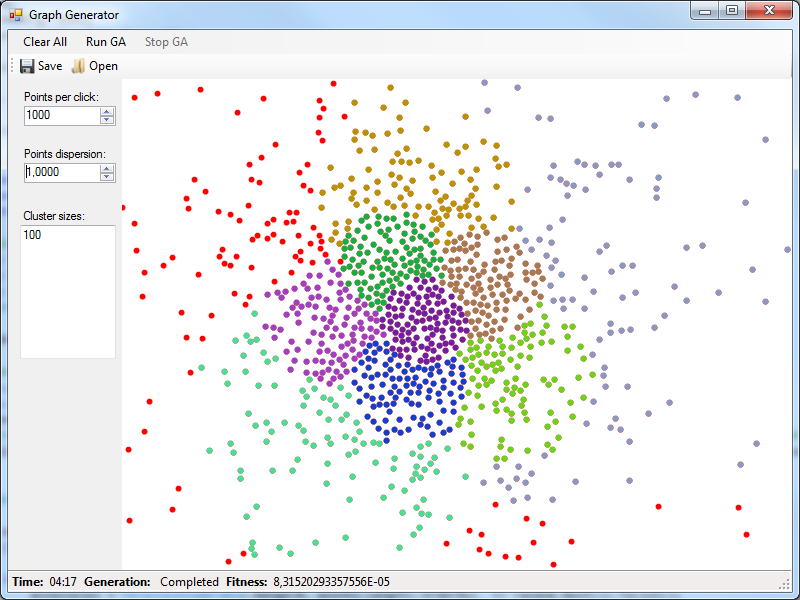


Рис.  8. Кластеризація відповідно до критерію мінімізації відстаней  
об’єктів до центру мас кластеру. Розв’язок не є очевидним

Найпростіший приклад альтернативного критерію кластеризації – обернений до того, який використовувався при розробці та тестуванні ГА кластеризації, а саме максимізація відстаней об’єктів кластеру до його центру мас (рис.  9). Така постановка критерію може, наприклад, бути необхідною при застосуванні кластеризації для розбиття людей на групи з метою статистичних досліджень чи порівняння ефективності дії різних медичних препаратів та плацебо. При цьому необхідно з множини добровольців сформувати однакові за кількістю групи людей абсолютно різних категорій за багатьма параметрами: стать, вік, достаток, середньодобова тривалість сну тощо.

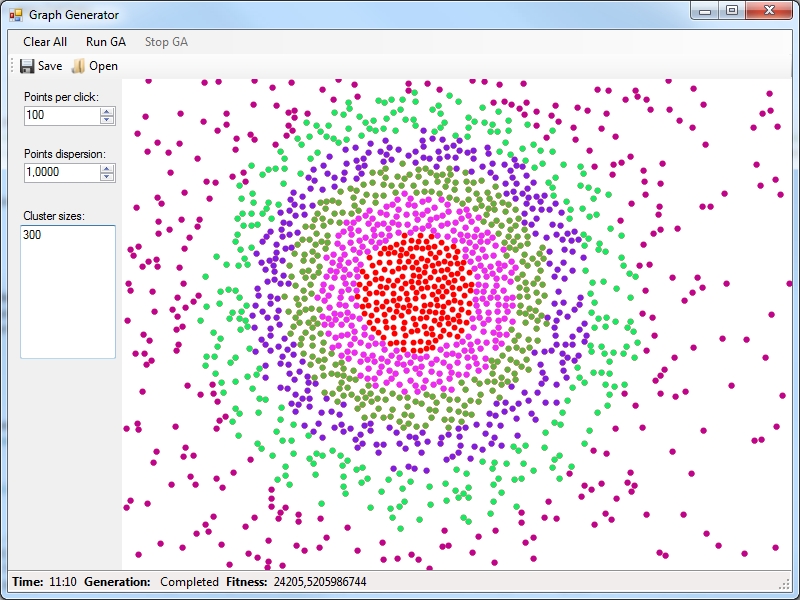


Рис.  9. Кластеризація відповідно до критерію максимізації відстані до центру мас кластера

Іншим прикладом задачі, для вирішення якої можна застосувати розроблений алгоритм, є одна з задач декомпозиції графів, а саме розбиття графа на фіксоване число підграфів заданих порядків (частковий випадок – бірозбиття графа) [9]. Критерієм якості розбиття є мінімізація зовнішніх ребер, інцидентних вершинам різних підграфів. У такому випадку кожен підграф розглядається як кластер. Відстань між двома вершинами підграфа дорівнює кількості ребер, що з’єднують їх. Функція пристосованості у такому випадку повертатиме суму кількості ребер між усіма парами вершин поточного кластеру.

Ще одним прикладом задачі з нетиповим критерієм є задача інтелектуальної кластеризації документів за тематикою. У такому випадку функція пристосованості може повертати суму кількості спільних специфічних слів для всіх пар документів в межах поточного кластеру.

# Аналіз результатів роботи алгоритму

## Визначення оптимальних параметрів

Розроблений генетичний алгоритм був реалізований програмно за допомогою середовища розробки Visual Studio 2010 на платформі .NET Framework 4.0 мовою програмування C# v4.0. Засобами Parallel FX, зокрема PLINQ, було здійснено розпаралелювання роботи з множинами ГА, а саме паралельне виконання кроссоверів над парами нащадків та мутацій при репродукції. Розроблено також графічний інтерфейс, в якому об’єкти подаються у вигляді точок та розфарбовуються в процесі роботи генетичним алгоритмом у відповідні кольори за ознакою належності до одного з кластерів. Програма працює з об’єктами, вектори ознак яких мають довжину 2, тобто координати по осі абсцис та ординат.

При багатократних випробуваннях роботи розробленого алгоритму над однією і тією ж вибіркою об’єктів були визначені наступні оптимальні числові параметри ГА:

* Розмір популяції – 512÷1024
* Ймовірність мутації – 0,5
* Коефіцієнт елітизму – 0,05
* Критерій зупинки – 10 поколінь без зміни результату

## Порівняння результатів

Для порівняння ефективності роботи розробленого алгоритму та відомих алгоритмів було використано засоби кластеризації пакету MATLAB 7.10.0 (R2010a), зокрема один з найпоширеніших методів кластеризації k-means та його модифікація fuzzy c-means [10].

Як показали дослідження, на вибірках об’єктів, де розміри кластерів відрізняються незначно і розв’язок є очевидним, внаслідок роботи розробленого генетичного алгоритму та алгоритмів k-means і fuzzy c-means були отримані близькі за якістю результати [11].

Для більш ретельного порівняння було створено вибірку із 700 об’єктів, яка внаслідок коректної кластеризації має бути розбита на один великий та два менші за розміром кластери. У якості вхідних параметрів розробленого генетичного алгоритму були вказані розміри кластерів – 500, 100, 100. Для алгоритмів k-means і fuzzy c-means, реалізованих в пакеті MATLAB, була вказана кількість кластерів – 3.

Як видно з рис.  10, генетичний алгоритм правильно згрупував один великий кластер розміром в 500 об’єктів та два кластери розмірами по 100 об’єктів. В деяких випадках іноді трапляється помилка ідентифікації лише поодиноких об’єктів вхідної множини.

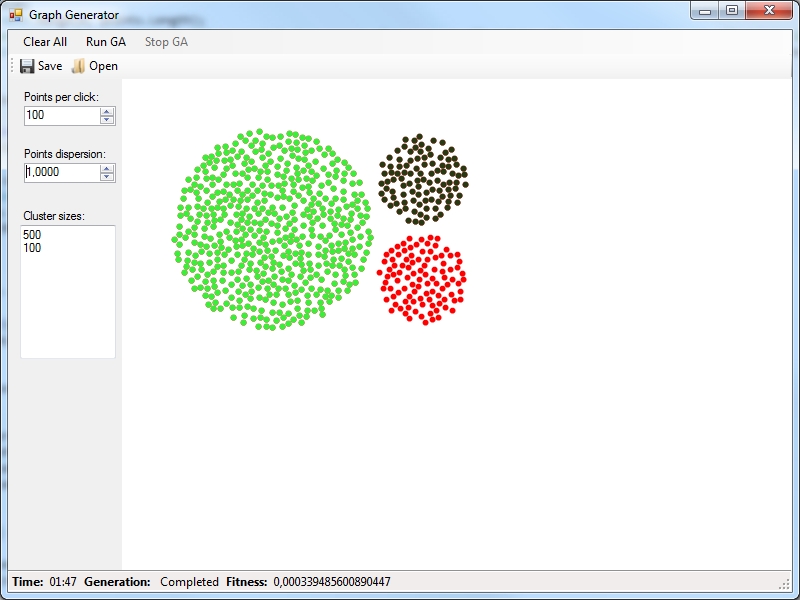


Рис.  10. Результати роботи генетичного алгоритму кластеризації 700 об’єктів

Результат роботи алгоритму k-means зображено на рис.  11. Результат роботи алгоритму fuzzy c-means аналогічний. В даному випадку помітний ефект «розщеплення» великого кластеру на два, а також помилкове злиття двох менших кластерів в один. Таким чином, алгоритм k-means виконав кластеризацію невірно для заданої вибірки об’єктів та кількості кластерів.

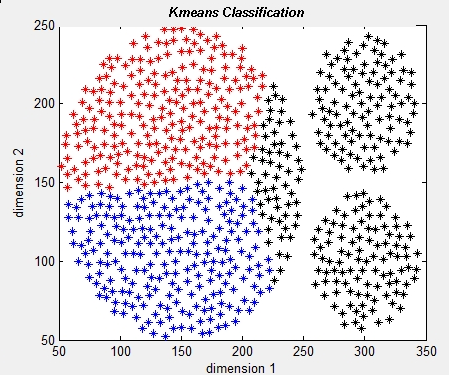


Рис.  11. Результати кластеризації 700 об’єктів методом k-середніх.  
Ефект «розщеплення» великого кластеру

Дослідження показують перевагу розробленого алгоритму для випадків, коли кластери мають значно відрізнятися один від одного. Це є черговим підтвердженням того, що явне задання розмірів кластерів має значну перевагу для окремих задач та вхідних даних. Крім того, розглянуті алгоритми k-means і fuzzy c-means не мають можливості регулювання розмірів кластерів у випадках, коли цього вимагає постановка задачі.

Час виконання генетичного алгоритму кластеризації поступається алгоритмам k-means і fuzzy c-means.

# Висновки

## Переваги та недоліки алгоритму

Розроблений генетичний алгоритм дозволяє ефективно розв’язувати задачу кластеризації при специфічних параметрах пошуку, таких як розміри кластерів та їх кількість одночасно. Необхідність задання розмірів кластерів часто є додатковою проблемою, оскільки ці розміри не є відомими заздалегідь. Проте існує ряд задач, постановка яких вимагає задання фіксованих розмірів кластерів для пошуку. Відомі алгоритми кластеризації дозволяють виконувати пошук кластерів лише відповідно до заданої кількості кластерів та інших специфічних параметрів для кожного окремого алгоритму, що впливають лише на якість кластеризації.

Запропонований генетичний алгоритм дозволяє проводити кластеризацію об’єктів у відповідності до заданого критерію, який задається у формі цільової функції ГА. Більшість відомих ефективних алгоритмів кластеризації дозволяють проводити кластеризацію лише відповідно до фіксованого критерію, який найчастіше лежить в основі принципу роботи самого алгоритму. Так, наприклад, основною ідеєю роботи алгоритму k-середніх є знаходження центру мас кластеру. Але існує ряд задач, коли знаходження центру мас є проблематичним, неефективним або взагалі неможливим. Так для задач кластеризації, критерій яких не передбачає знаходження центру мас, такі алгоритми як k-середніх та його модифікації є непридатними.

Розроблений генетичний алгоритм кластеризації, як і більшість еволюційних алгоритмів, дуже ефективно розпаралелюється за рахунок паралельного виконання схрещування та мутації, що забезпечує високу масштабованість алгоритму.

Запропонований алгоритм базується на основі операцій над множинами, що дає можливість реалізувати основні та найбільш ресурсоємкі частини програми за допомогою структурованих мов запитів до баз даних на сучасних СУБД.

Недоліком розробленого алгоритму кластеризації є її поетапне здійснення, тобто те, що пошук кластерів виконується окремо і незалежно один від одного. Це призводить до захоплення окремих об’єктів кластерами, що були сформовані раніше, хоча вони мають належати іншим кластерам. Такий недолік здебільшого проявляється при некластерній структурі множини вхідних даних. Наявність даного недоліку була прийнята до уваги при розробці алгоритму, але ціною його усунення є значно складніше кодування геному та виконання операції кроссоверу.

## Перспективи вдосконалення алгоритму

Надалі є доцільним вдосконалення кодування геному і розробка більш складної операції кроссоверу з метою виконання пошуку всіх кластерів одночасно. Такий підхід дає можливість оцінки якості кластеризації не одного окремого кластеру, а всіх кластерів в цілому. У такому випадку кожна хромосома представлятиме собою усі об’єкти вхідної множини, згруповані певним чином в кластери. Таким чином можна задавати більш глобальні критерії, що, наприклад, максимізують відстані між центрами мас всіх кластерів одночасно.

Перспективною також є розробка ієрархічного генетичного алгоритму кластеризації. Оскільки пошук кластерів малих розмірів виконується значно швидше ніж великих, то є можливим розбиття всієї вхідної множини об’єктів спочатку на кластери малих розмірів. На наступних етапах кожен невеликий кластер може розглядатися генетичним алгоритмом як новий об’єкт і кластеризація виконуватиметься повторно вже над новими об’єктами. Процес об’єднання малих кластерів в більші можна продовжувати до тих пір, поки не будуть отримані кластери необхідних розмірів.

# Список використаної літератури

1. *Fung G.* A Comprehensive Overview of Basic Clustering Algorithms. – IEEE, June, 2001 – Citeseer. – P. 37.
2. *Jain A.K., Murty M.N., Flynn P.J.* Data Clustering: A Review // ACM Computing Surveys, Vol. 31, No. 3, September 1999. – P. 265 – 267.
3. *Hartigan J. A.* Clustering Algorithms. New York, John Wiley and Sons, Inc., 1975. – P. 351.
4. *Al-Sultan K. S., Khan* *M. M*. Computational experience on four algorithms for the hard clustering problem // Pattern Recogn., Vol. 17, No. 3, 1996. – P. 295–308.
5. *Bäck T.* Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford, 1995. – P. 328.
6. *Dubes R. C.* How many clusters are best? An experiment // Pattern Recogn., Vol. 20, No. 6, 1996. – P. 645 – 663.
7. *Hathaway R. J., Bezdek J. C.* Optimization of Clustering Criteria by Reformulation // IEEE Transactions on Fuzzy Systems, Vol. 3, No. 2, May 1995. – P. 242 – 243.
8. *Baricelli N. A.* Numerical testing of evolution theories // ACTA Biotheoretica, Vol. 16, No. 1-2, 1962. – P. 69–126.
9. *Kernighan B. W.* Some Graph Partitioning Problems Related to Program Segmentation // Ph. D. Thesis. Princeton University, January 1969, – P. 74-126.
10. *Babu G. P., Murty M. N.* A nearoptimal initial seed value selection in K-means algorithm using a genetic algorithm // Pattern Recogn., Vol. 14, No. 10, Oct. 1993. – P. 763–769.
11. *Зорін Ю. М., Подольський С. В.* Генетичний алгоритм кластеризації множини відповідно до заданого критерію // ІІ наук.-техн. конф. «Прикладна математика та комп’ютинг». Тези доповідей. – К.: НТУУ «КПІ», 14–16 квітня 2010. – С. 335–338.

# Додаток 1. Копії графічних матеріалів

Блок-схема канонічного генетичного алгоритму



Блок-схема узагальненого генетичного алгоритму кластеризації



Блок-схема алгоритму генерації початкової популяції



Блок-схема алгоритму операції мутації



# Додаток 2. Програмний код генетичних операцій

using System.Collections.Concurrent;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Threading.Tasks;

using GeneticClusteringAlgorithm.Properties;

namespace GeneticClusteringAlgorithm

{

public class GA

{

public static readonly ThreadSafeRandom random = new ThreadSafeRandom();

// GA presets

const double elitismRate = 0.05;

const double MutationRate = 0.5;

const int populationSize = 512;

// Locals

readonly Point[] points;

readonly IEnumerable<int> indexArray;

readonly int clusterSize;

ConcurrentDictionary<IEnumerable<int>, double> population;

public GA(Point[] points, int clusterSize)

{

// Assign locals

this.points = points;

this.clusterSize = clusterSize;

indexArray = ParallelEnumerable.Range(0, points.Length);

// Create initial population with random chromosomes. All generated chromosomes always cover all points. It means that there are all points in initial population.

population = new ConcurrentDictionary<IEnumerable<int>, double>(InitialPopulation().ToDictionary(chromosome => chromosome, chromosome => Fitness(chromosome)));

}

IEnumerable<IEnumerable<int>> InitialPopulation()

{

var allAlleles = new List<int>();

for (var i = 0; i < populationSize; i++)

{

if (allAlleles.Count < clusterSize)

allAlleles = ParallelEnumerable.Range(0, points.Length).OrderBy(\_ => random.Next()).ToList();

yield return allAlleles.Take((int)clusterSize).ToArray();

allAlleles.RemoveRange(0, (int)clusterSize);

}

}

double Fitness(IEnumerable<int> chromosome)

{

var center = new Point((int)chromosome.Average(allele => points[allele].X), (int)chromosome.Average(allele => points[allele].Y));

return 1.0 / chromosome.Sum(allele => FormMain.Distance(points[allele], center));

}

static IEnumerable<int[]> Crossover(IEnumerable<int> father, IEnumerable<int> mother)

{

IEnumerable<int>

intersect = father.Intersect(mother), motherRest = mother.Except(intersect), fatherRest = father.Except(intersect);

int motherRestHalfSize = motherRest.Count() / 2, fatherRestRestHalfSize = fatherRest.Count() / 2;

yield return intersect.Concat(motherRest.Take(motherRestHalfSize)).Concat(fatherRest.Skip(fatherRestRestHalfSize)).ToArray();

yield return intersect.Concat(motherRest.Skip(motherRestHalfSize)).Concat(fatherRest.Take(fatherRestRestHalfSize)).ToArray();

}

IEnumerable<int> Mutation(int[] chromosome)

{

if (points.Length > chromosome.Length && random.NextDouble() <= MutationRate)

chromosome[random.Next(chromosome.Length)] = indexArray.Except(chromosome).ElementAt(random.Next(points.Length - chromosome.Length));

return chromosome;

}

public KeyValuePair<IEnumerable<int>, double> Reproduction()

{

// Normalize, accumulate and order fitness by descending

var fitnessSum = population.Sum(pair => pair.Value);

var accumulator = 0.0;

var roulette = (from pair in population orderby pair.Value descending select new { Chromosome = pair.Key, Fitness = pair.Value, AccumulatudFitness = (accumulator += pair.Value / fitnessSum) }).ToArray();

// Count of children pairs to be produced

var childrenPairs = (int)(populationSize \* (1 - elitismRate) / 2);

// Take count of elite chromosomes

population = new ConcurrentDictionary<IEnumerable<int>, double>(roulette.Take(populationSize - childrenPairs \* 2).ToDictionary(pair => pair.Chromosome, pair => pair.Fitness));

// Produce children and fill population with them

Parallel.For(0, childrenPairs, \_ =>

{

foreach (var child in from chromosome in Crossover(roulette.First(pair => pair.AccumulatudFitness >= random.NextDouble()).Chromosome, roulette.First(pair => pair.AccumulatudFitness >= random.NextDouble()).Chromosome) select Mutation(chromosome))

population[child] = Fitness(child);

});

// Return fittest cluster

return population.Aggregate((max, next) => next.Value > max.Value ? next : max);

}

}

}