# 2. Опис методів розв’язання задачі

**2.1. Існуючі методи розв’язання.**

Виходячи з класу задачі можна використовувати наближені алгоритми розв’язку такі як генетичні алгоритми, алгоритм мурашиних колоній, бджолиний і тп.

**2.2. Розробка генетичного алгоритму**.

Генетичний алгоритм це в першу чергу еволюційний алгоритм, що заснований на схрещуванні(комбінуванні). Шляхом перебору та відбору отримаємо правильну комбінацію. Алгоритм поділяється на три етапи:

* Схрещування
* Селекція(відбір)
* Формування нового покоління

Якщо результат нас не влаштовує, кроки повторюються допоки результат нас не влаштує або якщо кількість поколінь(повторів) досягне свого максимуму.

Опис алгоритму:

1. Випадковим чином генерується кінцевий набір готових рішень:

, (перше покоління, n – розмір популяції)

1. Оцінка пристосовуваності поточного покоління

1. Вихід, якщо виконується критерій зупинки.
2. Генерація нового покоління S, схрещуючи C та мутацій

та перехід до пункту 2 в іншому випадку.

В процесі селекції відбираються лише декілька кращих пробних рішень, інші далі не використовуються. Схрещування за місце пари рішень створює іншу, елементи якої перемішані випадковим чином. Мутація випадковим чином змінює певну компоненту рішення на нуль.

Опис алгоритму:

Вхід: А – матриця сумісності тварин

m – розмір початкової популяції

F – умова завершення, впродовж 20 ітерацій алгоритм не призводив до покращення розв’язку

Крок 1. Генерація початкової популяції

На першому кроці генерується популяція з m членів, в якій тварини, що входять до неї обираються ввипадковим чином. Алгорим перевіряє популяцію та визначає членів, які не задовільняють умові.

while не виконується F do

Крок 2. Обрати батків.

**2.3. Опис методу розв’язання**

Алгоритм 1 будується на випадковому відборі популяції, схрещуванні представників популяції з метою отримати краще потомство.

Алгоритм 1 завжди призведе до допустимого розв’язку так як на одному з кроків відбувається перевірка, чи задовольняє нащадок умові задачі. Отже, алгоритм буде відсіювати недопустимі розв’язки. Метод є наближеним, тому точної гарантії на отримання оптимального розв’язку немає. Алгоритм завершує роботу, після того, як впродовж 10 ітерацій не було покращення ЦФ.

Генетичний алгоритм працює за лінійний час та має лінійну складність.

**2.4. Опис алгоритму мурашиних колоній.**

Алгоритм мурашиних колоній представляє собою жадібний еврістичний алгоритм, що будує нове рішення на основі попереднього. Основною ідеєю є моделювання поведінки мурах на шляху до їжі. Мурахи позначають кращі шляхи феромонами, по котрих ідуть наступні мурахи. В умовах даної задачі позначатимемо «феромоном» тварин, які можуть бути перевезені з найбільшою кількістю інших тварин. На заключних кроках алгоритм будуватиме популяції переважно з цих тварин.

Алгоритм завжди приводитиме до допустимого розв’язку так як буде перевіряти розв’язки на допустимість.

Алгоритм є наближеним, тому не гарантує оптимальність розв’язку.

Псевдокод алгоритму:

feromones = [];

maxPopulation = 0;

bestPopulation = [];

done = false;

while done is false :

i = setFeromon(); //обираємо рядок матриці з найбільшою кількістю одиниць.

feromones.add(i);//помічаємо цей рядок(цю тварину)

population = makePopulationWith(feromones); //побудова та перевірка популяції з використанням феромону

//позначаємо новій розв'язок

if maxPopulation < population.size :

maxPopulation=population.size;

bestPopulation = population;

Складність алгоритму є квадратичною, так як, для кожного члена популяції вираховується сумісність тварин всередині популяції.

**2.5. Приклади застосування розроблених алгоритмів.**

Нехай задача задана матрицею

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |

* *Приклад застосування генетичного алгоритму.*

На першому кроці генеруємо популяцію з 8 членів. Позначимо ЦФ як .

1. (100101) , НЕ ЗАДОВІЛЬНЯЄ
2. (011001) , НЕ ЗАДОВІЛЬНЯЄ
3. (011000)
4. (010101) НЕ ЗАДОВІЛЬНЯЄ
5. (010010)
6. (001011)
7. (110001) НЕ ЗАДОВІЛЬНЯЄ
8. (111000)

Виключимо з популяції членів, що не задовільняють умові.

Для вибору батьків назначимо турнір між 3 та 8 і 5 та 7.

7 та 8 є кандидатами в батьки.

За допомогою одноточкового кросинговеру маємо двох нащадків

1. (110000)
2. (111001)

З певною ймовірністю кожен нащадок мутує, його випадковий ген інвертується. Перший нащадок мутації не піддається, у другого нащадка інвертується випадковий ген. Мутації піддається 5 ген.

Маємо

(111011) , що не задовільняє умові. Проведемо локальне покращення, замінимо 1 в 6 операторі на 0.

(111010)

Далі алгоритм проводить 10 ітерацій та завершує свою роботу, так як не відбуватиметься покращення та рішення буде прийнято, як найкраще.

* *Приклад застосування мурашиного алгоритму.*

Побудуємо граф.



Розмістимо мурах в початкові точки стратегією ковдра, всі вершини є початковими.

На першому кроці мурахи визначають та позначають феромонами вершину, яка зв’язана з найбільшою кількістю інших вершин(рядок з найбільшою кількістю одиниць). У даному випадку це 5 вершина. Позначимо її.



Тепер при підборі популяції, будемо враховувати. Що 5 вершина має бути обов’язково присутня.

(пропустимо кроки, що схожі на генетичний алгоритм)



Проведемо підбір популяції.

…

(111010) ЦФ = 4, що є найкращим поточним результатом та оптимальним для даної задачі.

…

Припиняємо роботу алгоритму, так як подальші кроки не покращать ЦФ.