# 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

α – ймовірність мутації

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 батьків.

Для вибору батьків скористаємося турнірним методом, в якому кожному популяція буде поділена на дві частини випадковим чином. Кожному члену буде присвоєно номер команди (1 або 2). З кожної команди буде обрано одного найкращого представника. Так, у результаті ми матимемо двох батьків та можемо переходити до створення нащадків.

Крок 3. Схрещення.

На етапі схрещення отриманих у попередньому кроці батьків ми застосуємо одноточковий кросинговер. Для цього визначимо опорну точку кросинговеру, поділивши порівну гени на дві частини. Далі формуємо двох нащадків. Випадковим чином обирається точка кросинговеру. Перший нащадок отримає гени першого батька по ліву частину точки та другого по праву, другий- праву частину першого та ліву другого.

Крок 4. Мутації.

На цьому етапі, з певною ймовірністю з кожним отриманим нащадком може відбутися мутація. Використаємо одноточковий оператор мутації. Приймемо ймовірність мутації за 0,1. При мутації, випадковий ген нащадка змінює своє значення.

Крок 5. Локальне покращення(Реанімація)

Якщо один з нащадків виявився непридатним, спробуємо зробити реанімацію. Оцінимо, які тварина конфліктують одна з одною. Оцінимо, з якою кількістю конфліктує кожна тварина. Далі, поки не буде досягнуто допустимого розв’язку, на кожній ітерації прибираємо тварину, яка має найбільше конфліктів, після чого перераховуємо конфлікти, і, якщо їх немає, переходимо до включення в популяцію.

Крок 6. Оновлення популяції.

При оновленні популяції слід робити відсів 2-х найгірших розв’язків, щоб не працювати з поганими результатами. Для цього додамо отриманих нащадків в популяцію та оцінимо цільову функцію (кількість тварин в кожному наборі) для кожного члена популяції. Два найгірших результати видаляються з популяції. Якщо один з отриманих результатів є кращим за попередній рекорд, визначаємо новий рекорд. Якщо покращення не було впродовж 20 ітерацій, завершуємо роботу алгоритму.

**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 так як використовуються графи та мурахи можуть переходити лише до суміжних вершин.

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

Опис алгоритму

Крок 0. Ініціалізація початкових параметрів.

Початкове значення феромону – 1. Кількість мурах – 1.

α = 1

β = 1

ρ = 0.4

Доки не досягнуто умов виходу виконуються наступні кроки:

Крок 1. Створення мурах.

На першому кроці кожної ітерації розміщаємо всю колонію у випадкову вершину. Ймовірність розміщення колонії у вершині вираховується за формулою

де – кількість шляхів з вершини і, N – кількість ребер. Тобто ймовірність розміщення колонії у вершині прямо пропорційна кількості суміжних з нею вершин. Поточну вершину включимо у шлях S

Крок 2. Пошук рішень.

Вирахуємо ймовірності переходу мурахи у суміжну вершину. Ймовірність переходу у і-ту суміжну вершину:

(1)

Де – кількість спільних суміжних вершин з вершиною і та вершин зі шляху S, - значення феромону для даного шляху.

Після переходу додамо вершину у шлях та за формулою (1) вирахуємо наступний перехід. Цю операцію повторюватимемо доки не залишиться вершин, що суміжні з усіма вершинами шляху.

Коли знайдено всі вершини шляху, якщо поточний шлях кращий на поточний кращий, оновлюємо його.

Крок 3. Оновлення феромонів.

Перерахуємо феромони. Для кожного ребра, що входить в шлях нове значення феромону

Де L – кількість вершин у знайденому шляху, N – вершин всього.

В іншому випадку:

Якщо існують вершини, що зв’язані лише з однією суміжною, помітимо її як не оптимальну перерахувавши феромон як

Кроки 1-3 повторюються, доки вподовж 20 ітерацій не буде покращення результату.

Алгоритм є еврістичним, буде зроблено m (m ≥ 20) ітерацій, кожна з яких є лінійною.

**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 6.

6 та 8 є кандидатами в батьки.

За допомогою одноточкового кросинговеру маємо двох нащадків. Випадковою точкою э точка 3.

1. (001000)
2. (111011)

З ймовірністю α кожен нащадок мутує, його випадковий ген інвертується. Обидва нащадки не піддаються мутації.

Перший нащадок задовольняє умові. Другий – ні. Проведемо для нього реанімацію. Для цього доки розв’язок не стане допустимим, прибиратимемо найконфліктнішу тварину. Для другого нащадка мажмо, що 1 та 2 тварина мають по одному конфлікту(з 6) 3 та 5 – не має жодного, а 6 – конфліктує з 2. Прибираємо її як найконфліктнішу.

Змінений нащадок має вигляд:

(111010)

Так як цей розв’язок є кращим, за поточний найкращий, запишемо його, як поточний найкращий. Тобто ми отримали покращення розв’язку.

Додамо наших нащадків до популяції та приберемо з популяції 2 найгірші розв’язки.

Нова популяція матиме вигляд:

1. (010010)
2. (001011)
3. (111000)
4. (111010)

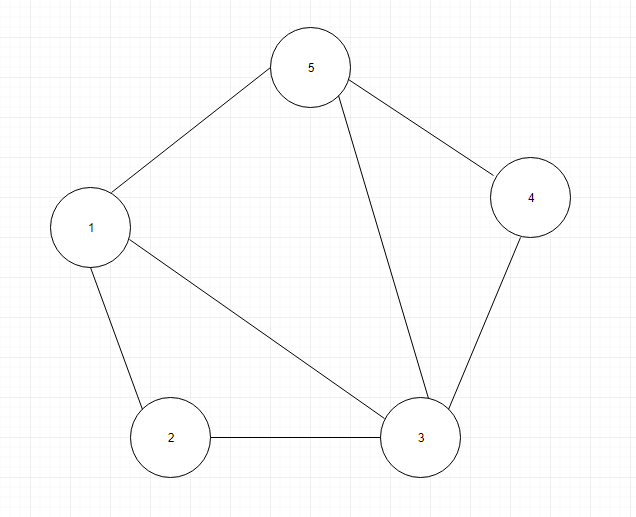
Повторюватимемо алгоритм, доки впродовж 20 ітерацій не буде покращення розв’язку. У цьому випадку, алгоритм повертає поточне найкраще значення як (наближено) оптимальний розв’язок.

* *Приклад застосування мурашиного алгоритму.*

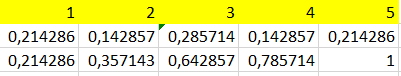
Маємо наступну задачу

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

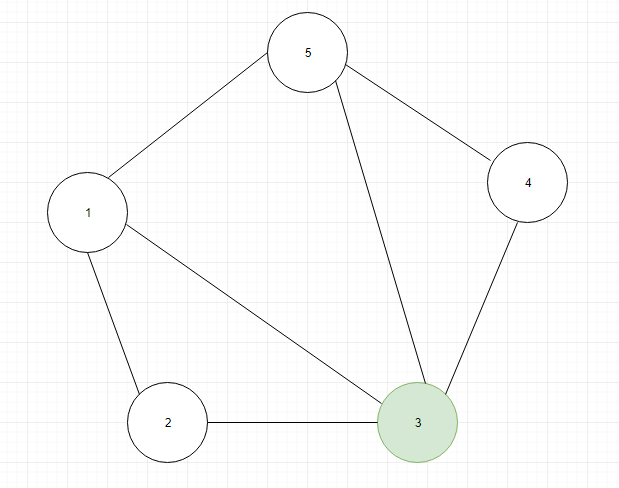
Що може бути представлена наступним графом



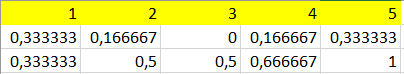
Ймовірності переходів та відповідних діапазонів представлені



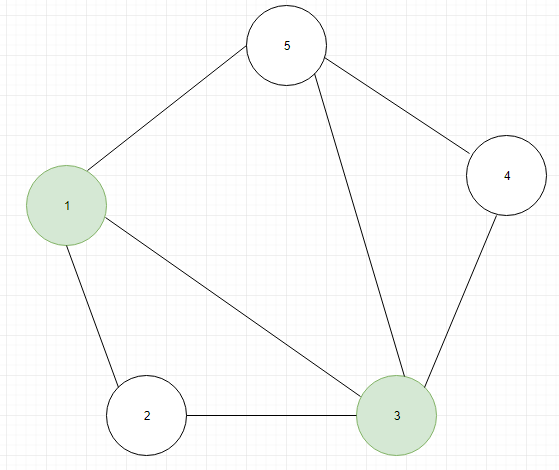
Випадає число 0,57, початкова вершина – 3.



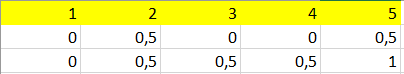
Ймовірності переходів та відповідні діапазони представлені нижче:



Випадає число 0,21. Переходимо у 1 вершину.

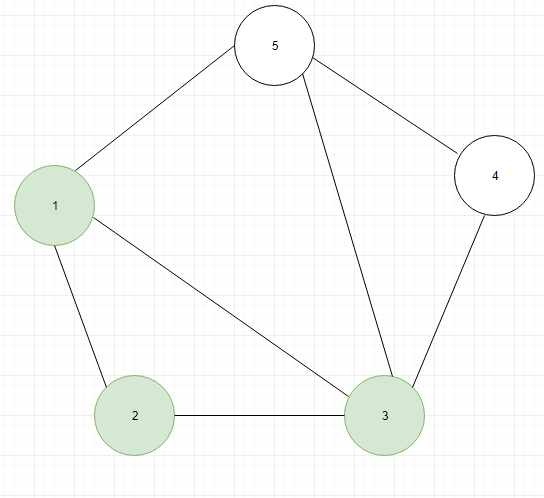


Перераховуємо ймовірності



4 вершина 0 так як . Тобто 4 вершина не суміжна кожному елементу поточного шляху.

Випадає число 0,37. Переходимо у вершину 2.



Перша ітерація завершена так як усі

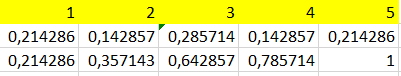
Перерахуємо феромони нова матриця феромонів матиме вигляд

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1,36 | 1,36 | 0 | 0,4 |
| 1,36 | 0 | 1,36 | 0 | 0 |
| 1,36 | 1,36 | 0 | 0,4 | 0,4 |
| 0 | 0 | 0,4 | 0 | 0,4 |
| 1,36 | 0 | 0,4 | 0,4 | 0 |

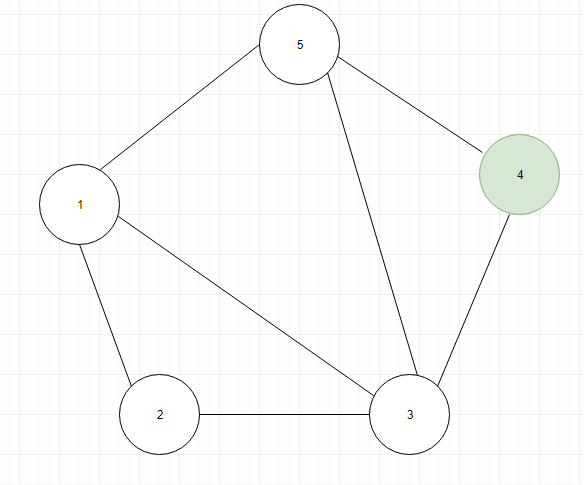
L = 3.

Ітерація 2.

Ймовірності переходів та відповідних діапазонів представлені



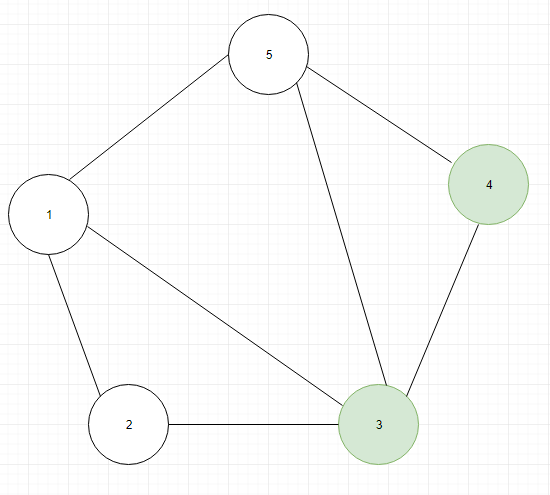
Випадає число 0,84, початкова вершина – 4.



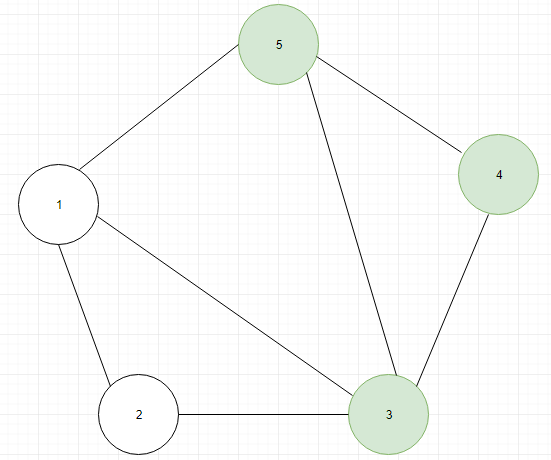
Розрахуємо ймовірності подальшого переходу.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 0 | 0 | 0,5 | 0 | 0,5 |
| 0 | 0 | 0,5 | 0,5 | 1 |

Випадає число 0,40. Переходимо у вершину 3.



Зі ймовірністю 1 переходимо у вершину 5, як єдину, що сумісна з кожним елементом шляху.



L = 3. Розв’язок не покращився, перерахуємо феромони.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0,544 | 0,544 | 0 | 0,16 |
| 0,544 | 0 | 0,544 | 0 | 0 |
| 0,544 | 0,544 | 0 | 0,76 | 0,76 |
| 0 | 0 | 0,76 | 0 | 0,76 |
| 0,544 | 0 | 0,76 | 0,76 | 0 |

Через 18 ітерацій у разі не покращення розв’язку алгоритм припинить свою роботу.