МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. БАУМАНА

Факультет: Информатика и системы управления
Кафедра: Информационная безопасность (ИУ8)

ТЕОРИЯ ИГР И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

Лабораторная работа № 6 на тему:

«Решение антагонистической игры информационного противоборства»

Вариант 4

Преподаватель:
Коннова Н.С.

Студент:
Куликова А.В.

Группа:
ИУ8-21М

Цель работы

Изучить теоретико-игровую модель информационного противоборства в социальных сетях. Промоделировать информационное управление в рамках игры и определить итоговое мнение агентов.

Постановка задачи

1. Для 10 агентов случайным образом сгенерировать стохастическую матрицу доверия.

2. Назначить всем агентам случайное начальное мнение из диапазона на усмотрение преподавателя. Провести моделирование для игры без влияния до получения устойчивого мнения. Привести ответ.

3. Случайным образом выбрать количество и номера (непересекающиеся) агентов влияния из общего числа агентов для первого и второго игроков. Назначить им начальные мнения первого и второго игрока. Остальным агентам (нейтральным) назначить случайные начальные мнения. Промоделировать информационное управление в рамках игры и определить итоговое мнение агентов.

Ход работы

Рассмотрим игру для 10 агентов. Сначала сгенерируем матрицу доверия, убедимся, что она стохастическая:

0.002 0.076 0.022 0.167 0.196 0.110 0.073 0.035 0.314 0.005
0.173 0.180 0.147 0.013 0.117 0.110 0.031 0.071 0.154 0.004
0.099 0.064 0.130 0.059 0.079 0.226 0.124 0.081 0.116 0.022
0.137 0.079 0.144 0.003 0.160 0.059 0.067 0.183 0.153 0.016
0.084 0.069 0.083 0.190 0.155 0.072 0.160 0.109 0.054 0.024
0.147 0.042 0.050 0.079 0.032 0.194 0.134 0.027 0.199 0.096
0.008 0.207 0.126 0.110 0.026 0.069 0.185 0.154 0.069 0.046
0.002 0.139 0.158 0.134 0.007 0.024 0.259 0.175 0.032 0.072
0.169 0.078 0.079 0.136 0.031 0.146 0.002 0.138 0.085 0.134
0.105 0.091 0.013 0.069 0.042 0.250 0.127 0.082 0.061 0.160

Затем сформируем случайным образом вектор начальных мнений агентов (без влияния), например, в диапазоне от 1 до 20, и произведем необходимое количество итераций до схождения матрицы (зададим точность $\varepsilon = 10{-}6$), получим итоговое мнение агентов:

X(0) = (18.409 3.809 15.873 1.444 19.294 8.011 7.680 13.863 15.289 5.963)

Потребовалось итераций: 12

Результирующее мнение агентов (без влияния):

X($t \to \infty$)= (10.989 10.989 10.989 10.989 10.989 10.989 10.989 10.989 10.989 10.989)

Итоговая матрица:

0.094	0.104	0.099	0.096	0.080	0.123	0.114	0.107	0.126	0.057
0.094	0.104	0.099	0.096	0.080	0.123	0.114	0.107	0.126	0.057
0.094	0.104	0.099	0.096	0.080	0.123	0.114	0.107	0.126	0.057
0.094	0.104	0.099	0.096	0.080	0.123	0.114	0.107	0.126	0.057
0.094	0.104	0.099	0.096	0.080	0.123	0.114	0.107	0.126	0.057
0.094	0.104	0.099	0.096	0.080	0.123	0.114	0.107	0.126	0.057
0.094	0.104	0.099	0.096	0.080	0.123	0.114	0.107	0.126	0.057
0.094	0.104	0.099	0.096	0.080	0.123	0.114	0.107	0.126	0.057
0.094	0.104	0.099	0.096	0.080	0.123	0.114	0.107	0.126	0.057
0.094	0.104	0.099	0.096	0.080	0.123	0.114	0.107	0.126	0.057

Сформируем вектор начальных мнений агентов с учетом информационного влияния, нейтральным агентам назначим случайные стартовые мнения. Проведем моделирование:

Агенты первого игрока: (1,2), агенты второго игрока: (3,4)

Сформированное начальное мнение агентов первого игрока: 42.954

Сформированное начальное мнение агентов второго игрока: -23.264

Изначальные мнения с учетом сформированных:

X(0) = (18,409 42,954 42,954 -23,264 -23,264 8,011 7,680 13,863 15,289 5,963)

Потребовалось итераций: 13

Результирующее мнение агентов (без влияния):

X($t \to \infty$)= (11,938 11,938 11,938 11,938 11,938 11,938 11,938 11,938 11,938 11,938)

Первый игрок выигрывает

Результат программы представлен в приложении А.

Выводы

Промоделировали информационное управление в рамках игры и определили итоговое состояние агентов. Был сделан вывод о победе первого игрока.

Контрольные вопросы

1. Дайте определение и опишите основные свойства матрицы доверия.

Матрица доверия (Trust Matrix) - это инструмент, который используется для оценки уровня доверия между участниками взаимодействия в системе или сообществе. Основные свойства матрицы доверия включают:

1. Двустороннее отношение: Матрица доверия обычно представляет собой двумерную таблицу, где участники системы или сообщества расположены по строкам и столбцам. Каждая ячейка матрицы показывает уровень доверия одного участника к другому.

2. Симметричность: Матрица доверия обычно симметрична относительно главной диагонали, что означает, что уровень доверия от участника A к участнику B равен уровню доверия от участника B к участнику A.

3. Относительный характер: Уровень доверия в матрице может быть представлен числовыми значениями или категориями (например, "высокий", "средний", "низкий"). Это позволяет оценить, насколько один участник доверяет другому по сравнению с остальными участниками.

4. Использование в алгоритмах принятия решений: Матрица доверия может быть использована в различных алгоритмах принятия решений, например, при распределении ресурсов или принятии решений о сотрудничестве между участниками.

5. Изменяемость: Уровень доверия между участниками может изменяться в зависимости от различных факторов, таких как прошлый опыт сотрудничества, репутация участников, информация о предыдущих взаимодействиях и т.д.

2. Что такое информационное управление и как оно осуществляется?

Информационное управление (Information Management) - это процесс управления информацией в организации, который включает в себя сбор, хранение, обработку, передачу, анализ и использование информации для поддержки бизнес-процессов и принятия решений.

Основные задачи информационного управления включают:

1. Сбор и накопление информации: организация должна аккумулировать данные из различных источников для последующего анализа и использования.

2. Хранение и обработка информации: информация должна быть структурирована, классифицирована и храниться в безопасном и доступном виде. Также информация должна обрабатываться с использованием соответствующих технологий.

3. Передача и обмен информацией: информация должна передаваться между различными участниками организации или сторонними организациями в безопасной и эффективной форме.

4. Анализ и использование информации: информация должна анализироваться для выявления паттернов, трендов и важных данных, которые

могут быть использованы для принятия решений и оптимизации бизнес-процессов.

3. Опишите принцип формирования итогового мнения агентов при моделировании информационного управления.

При моделировании информационного управления с использованием агентов, итоговое мнение агентов формируется на основе их индивидуальных характеристик, знаний, опыта, целей и предпочтений. Принцип формирования итогового мнения агентов может быть описан следующим образом:

1. Сбор информации: Каждый агент собирает информацию из своего окружения, включая данные о состоянии системы, решениях других агентов, внешних факторах и т.д.

2. Анализ информации: Агенты анализируют собранную информацию с учетом своих целей, задач и предпочтений. Они могут использовать различные методы анализа данных, статистические модели, машинное обучение и другие техники для выявления закономерностей и трендов.

3. Формирование мнения: На основе анализа информации каждый агент формирует свое собственное мнение или прогноз относительно текущей ситуации или будущего развития событий.

4. Обмен мнениями: Агенты могут обмениваться своими мнениями, обсуждать различные точки зрения, аргументировать свои решения и пытаться достичь консенсуса или согласованного решения.

5. Синтез итогового мнения: На основе собранных мнений от всех агентов система может производить синтез итогового мнения путем учета весов различных мнений, уровня доверия к агентам, предшествующих результатов и других факторов.

6. Принятие решений: Итоговое мнение агентов может быть использовано для принятия решений, определения стратегии управления информацией, оптимизации бизнес-процессов и других целей.

Приложение А

```
[Trust Matrix]
 0.002 0.076 0.022 0.167 0.196 0.110 0.073 0.035 0.314 0.005
 0.173 0.180 0.147 0.013 0.117 0.110 0.031 0.071 0.154 0.004
 0.099 0.064 0.130 0.059 0.079 0.226 0.124 0.081 0.116 0.022
 0.137 0.079 0.144 0.003 0.160 0.059 0.067 0.183 0.153 0.016
 0.084 0.069 0.083 0.190 0.155 0.072 0.160 0.109 0.054 0.024
 0.147 0.042 0.050 0.079 0.032 0.194 0.134 0.027 0.199 0.096
 0.008 0.207 0.126 0.110 0.026 0.069 0.185 0.154 0.069 0.046
 0.002 0.139 0.158 0.134 0.007 0.024 0.259 0.175 0.032 0.072
 0.169 0.078 0.079 0.136 0.031 0.146 0.002 0.138 0.085 0.134
 0.105 0.091 0.013 0.069 0.042 0.250 0.127 0.082 0.061 0.160
[Initial opinions (X0)]
  18.409   3.809  15.873   1.444  19.294   8.011   7.680  13.863  15.289   5.963
Number of iterations n:   12
Final opinions after n iterations of X(n):
  10.989  10.989  10.989  10.989  10.989  10.989  10.989  10.989  10.989  10.989
[Final matrix (all rows must be the same)]
 0.094 0.104 0.099 0.096 0.080 0.123 0.114 0.107 0.126 0.057
 0.094 0.104 0.099 0.096 0.080 0.123 0.114 0.107 0.126 0.057
 0.094 0.104 0.099 0.096 0.080 0.123 0.114 0.107 0.126 0.057
 0.094 0.104 0.099 0.096 0.080 0.123 0.114 0.107 0.126 0.057
 0.094 0.104 0.099 0.096 0.080 0.123 0.114 0.107 0.126 0.057
 0.094 0.104 0.099 0.096 0.080 0.123 0.114 0.107 0.126 0.057
 0.094 0.104 0.099 0.096 0.080 0.123 0.114 0.107 0.126 0.057
 0.094 0.104 0.099 0.096 0.080 0.123 0.114 0.107 0.126 0.057
 0.094 0.104 0.099 0.096 0.080 0.123 0.114 0.107 0.126 0.057
 0.094 0.104 0.099 0.096 0.080 0.123 0.114 0.107 0.126 0.057

[Game with information influence]
Management for agents of influence of the first player u: 42.954
Control for agents of influence of the second player v:   -23.264
Indices of agents of influence of the first player:   1 2
Indices of agents of influence of the second player:  3 4
[Initial opinions (X0)]
  18.409  42.954  42.954 -23.264 -23.264   8.011   7.680  13.863  15.289   5.963
Number of iterations n:   13
[Final opinions after n iterations X(n)]
  11.938  11.938  11.938  11.938  11.938  11.938  11.938  11.938  11.938  11.938
The first player wins
```

Приложение Б

```
#include <string>

#include <iostream>
#include <math.h>
#include <array>
#include <vector>
#include <iomanip>
#include <algorithm>

const int N = 10;

namespace gameTheoryAndOperationsResearch
{

// Умножение матрицы на вектор
// Принимает матрицу размерности NxN и вектор размерности N, возвращает
результат умножения - вектор размерности N
std::array<double, N>
matrix_vector_multiplication(std::array<std::array<double, N>, N> &Matr,
std::array<double, N> &Vec)
{
    std::array<double, N> Vecout; // Результирующий вектор
    for (int i = 0; i < N; i++)
    {
        Vecout[i] = 0; // Инициализация элемента результирующего вектора
        for (int j = 0; j < N; j++)
            Vecout[i] += Matr[i][j] * Vec[j]; // Умножение и суммирование
элементов для получения нового значения
    }
    return Vecout; // Возвращение результирующего вектора
}


// Умножение матрицы на матрицу
// Принимает две матрицы размерности NxN и возвращает произведение этих
матриц
std::array<std::array<double, N>, N>
matrix_matrix_multiplication(std::array<std::array<double, N>, N> &Matr1,
std::array<std::array<double, N>, N> &Matr2)
{
    std::array<std::array<double, N>, N> MatrOut; // Результирующая матрица
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            MatrOut[i][j] = 0; // Инициализация элемента результирующей
матрицы
            for (int k = 0; k < N; k++)
                MatrOut[i][j] += Matr1[i][k] * Matr2[k][j]; // Умножение и
суммирование элементов для получения нового значения
        }
    }
    return MatrOut; // Возвращение результирующей матрицы
}


// Печать матрицы
```

```cpp
void print(std::array<std::array<double, N>, N> &Matr)
{
    std::cout << std::fixed;
    std::cout << std::setprecision(3);
    for (auto &pos : Matr)
    {
        for (auto pos2 : pos)
            std::cout << std::setw(6) << pos2;
        std::cout << std::endl;

    }
}

// Печать вектора
void print(std::array<double, N> &Vec)
{
    std::cout << std::fixed;
    std::cout << std::setprecision(3);
    for (auto pos : Vec)
        std::cout << std::setw(8) << pos;
    std::cout << std::endl;
}

// Функция для сравнения двух векторов с заданной точностью
// Принимает два вектора, и значение точности e
bool comparison_of_accuracy_vectors(std::array<double, N>& Vec1,
std::array<double, N>& Vec2, double e)
{
    for (int i = 0; i < N; i++)
        if (fabs(Vec1[i] - Vec2[i]) > e)
            return false; // Если разница между элементами больше e, точность
не достигнута
    return true; // Если все элементы удовлетворяют условию, точность
достигнута
}


double rand_A_B(double a, double b) // Генерация псевдослучайного числа в
интервале [a, b)
{
    return ((double)rand() / RAND_MAX)*(b - a) + a;
}

// Функция для моделирования игры с агентами влияния
// Принимает матрицу взаимодействий, вектор мнений, индексы агентов влияния
первого и второго игроков
std::array<double, N> influencer_games(std::array<std::array<double, N>, N>&
Matr, std::array<double, N>& Vec,
    std::vector<size_t>& index1, std::vector<size_t>& index2)
{
    std::array<double, N> Vecout; // Выходной вектор мнений
    std::vector<size_t> index0; // Индексы пользователей не являющихся
агентами

    // Получение индексов пользователей не являющихся агентами
    for (int i = 0; i < N; i++) {
        if (find(index1.begin(), index1.end(), i) != index1.end()) continue;
        if (find(index2.begin(), index2.end(), i) != index2.end()) continue;
        index0.push_back(i);
    }

    // Инициализация выходного вектора нулями
    for (int i = 0; i < N; i++)
        Vecout[i] = 0;
```

```
    // Вычисление влияния агентов первого игрока на мнения
    for (auto pos : index1)
        for (int i = 0; i < N; i++)
            Vecout[pos] += Matr[pos][i] * Vec[i];

    // Вычисление влияния агентов второго игрока на мнения
    for (auto pos : index2)
        for (int i = 0; i < N; i++)
            Vecout[pos] += Matr[pos][i] * Vec[i];

    // Вычисление влияния пользователей не являющихся агентами на мнения
    for (auto pos : index0)
        for (int i = 0; i < N; i++)
            Vecout[pos] += Matr[pos][i] * Vec[i];

    return Vecout; // Возврат результирующего вектора мнений
}


int _main(int argc, char* argv[])
{
    std::array<std::array<double, N>, N> A; // Матрица A

    srand(43); // Устанавливаем начальное значение для генератора случайных
чисел

    std::cout << "[Trust Matrix]\n"; // Выводим заголовок матрицы

    for (int i = 0; i < N; i++)
    {
        double ost = 1, step = 0.1; // Инициализация остатка и шага для
распределения вероятностей
        for (int j = 0; j < N - 1; j++)
        {
            A[i][j] = rand_A_B(0.001, step * 2 - 0.001); // Заполняем
элементы матрицы случайными значениями
            ost -= A[i][j]; // Вычитаем текущее значение из остатка
            step = ost / (N - 1 - j); // Рассчитываем новый шаг
        }
        A[i][N - 1] = ost; // Заполняем последний элемент строки для
обеспечения суммы равной 1
    }

    print(A);

    std::array<std::array<double, N>, N> ItogMatr = A; // Итоговая матрица,
инициализируется начальной матрицей A
    std::array<double, N> X0; // Начальный вектор

    for (int i = 0; i < N; i++)
        X0[i] = rand_A_B(1, 20); // Заполняем начальный вектор случайными
значениями от 1 до 20

    std::cout << "[Initial opinions (X0)]\n"; // Вывод заголовка для
начального вектора
    print(X0); // Вывод начального вектора

    std::array<double, N> X = X0, XOld; // Вектора для хранения текущего и
предыдущего состояний
    int n = 0; // Счетчик итераций

    do // Цикл перехода из состояния в состояние
    {
```

```
        XOld = X; // Сохраняем текущий вектор
        X = matrix_vector_multiplication(A, X); // Вычисляем новый вектор
        ItogMatr = matrix_matrix_multiplication(ItogMatr, A); // Умножаем
итоговую матрицу на матрицу A
        n++; // Увеличиваем счетчик итераций
    } while (!comparison_of_accuracy_vectors(X, XOld, 1e-6)); // Проверяем
точность совпадения векторов

    std::cout << "Number of iterations n:   " << n << std::endl; // Вывод
количества итераций
    std::cout << "Final opinions after n iterations of X(n):    " <<
std::endl; // Вывод финального вектора после n итераций
    print(X); // Вывод финального вектора X

    std::cout << "[Final matrix (all rows must be the same)]\n"; // Вывод
заголовка для финальной матрицы
    print(ItogMatr); // Вывод финальной матрицы

    std::cout<<std::endl; // Пустая строка для отделения вывода

    // Далее игра с информационным влиянием
    std::cout<<"[Game with information influence]\n"; // Вывод заголовка для
игры с информационным влиянием

    double u = rand_A_B(10, 100); // Управление для агентов 1-го игрока
    double v = -rand_A_B(10, 100); // Управление для агентов 2-го игрока

    std::cout << "Management for agents of influence of the first player u: "
<< u << std::endl; // Вывод управления для агентов 1-го игрока
    std::cout << "Control for agents of influence of the second player v:   "
<< v << std::endl; // Вывод управления для агентов 2-го игрока

    // Назначаем агентов влияния
    std::vector<size_t> index1 = { 1, 2 };  // Индексы агентов влияния 1-го
игрока
    std::vector<size_t> index2 = { 3, 4 }; // Индексы агентов влияния 2-го
игрока


    // Вывод индексов агентов влияния первого игрока
    std::cout << "Indices of agents of influence of the first player:   ";
    for (auto pos : index1)
        std::cout << pos << ' ';
    std::cout << std::endl;

    // Вывод индексов агентов влияния второго игрока
    std::cout << "Indices of agents of influence of the second player:  ";
    for (auto pos : index2)
        std::cout << pos << ' ';
    std::cout << std::endl;

    // Присвоение управлений агентам влияния
    for (auto pos : index1)
        X0[pos] = u;
    for (auto pos : index2)
        X0[pos] = v;

    std::cout << "[Initial opinions (X0)]\n";
    print(X0); // Вывод начальных мнений

    X = X0; // Присвоение начальных мнений
    n = 0; // Счетчик итераций
    do
    {
```

```
        XOld = X; // Сохранение предыдущего состояния мнений
        X = influencer_games(A, X, index1, index2); // Выполнение игры с информационным влиянием
        n++; // Увеличение счетчика итераций
    } while (!comparison_of_accuracy_vectors(X, XOld, 1e-6)); // Проверка на точность сходимости

    std::cout << "Number of iterations n:   " << n << std::endl; // Вывод количества итераций
    std::cout << "[Final opinions after n iterations X(n)]\n"; // Вывод финальных мнений после n итераций
    print(X); // Вывод финальных мнений

    // Определение победителя
    (X[0] >= 0) ?
        std::cout << "The first player wins" :
        std::cout << "The second player wins";

    return 0;
}

}


int main(int argc, char* argv[])
{
    gameTheoryAndOperationsResearch::_main(argc, argv);
    return 0;
}
```