Одеський національний університет імені І. І. Мечникова
Інститут математики, економіки і механіки
Кафедра оптимального керування та економічної кібернетики

Д и п л о м н а р о б о т а

бакалавра

на тему: **«**Узагальнення задачі про оптимальну зупинку**»**
«Generalization of optimal stopping problem»

Виконала студентка денної форми навчання
напряму підготовки 6.040301 Прикладна математика
Слободянюк Ольга Сергіївна
Керівник к. ф.-м. н, доцент Арсірій А.В. _____
Рецензент к. ф.-м. н, доцент Єфимова Г.О.

Рекомендовано до захисту:

Протокол засідання кафедри

№_____ від _____ р.

Завідувач кафедри

_____ _____
(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК №_____

протокол №____ від _____ р.

Оцінка_____/___ ___/_____
(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ _____
(підпис) (прізвище, ініціали)

Одеса – 2017

ЗМІСТ

ВСТУП

Із задачею про оптимальну зупинку ми зіштовхуємося в усіх галузях життя, у господарстві, економіці, на виробництві. Для нас важливо отримати найкращий результат, який тільки можливо на даний момент. Наша задача полягає у тому, щоб, перебуваючи «між молотом і ковадлом», мінімізувати свої втрати, аби не довелось шкодувати про свій вибір чи мріяти про ідеал, який так і не був досягненим через поспішно прийняте або незважене рішення. З самого початку ми знаходимося у тяжкому положенні, бо не знаємо, яким чином дізнатися, що варіант найкращий, без якогось стандарту оцінки та яким чином можна було б цей стандарт визначити. Неможливо визначити його без того, щоб втратити якусь кількість варіантів. Чим більше ми отримуємо інформації, тим більше стає наш шанс, що ми розпізнаємо наш ідеальний варіант, але при цьому ж збільшується ймовірність того, що ідеальний варіант залишився у минулому.

Що ж робити? Як прийняти зважене рішення, враховуючи те, що сам процес збору інформації ставить під питання можливість позитивного розв’язання ситуації? Це ставить нас у непросте положення і стоїть на границі з парадоксом. Головна дилема не у виборі певного варіанту, а у тому, яку кількість цих варіантів повинно бути у нашому розпорядженні. Зіткнувшись із такою ситуацією більшість людей інтуїтивно кажуть, що необхідно знайти певний баланс між пошуком і прийняттям рішення, що нам необхідно побачити достатню кількість варіантів, щоб зрозуміти, чого саме ми бажаємо, а потім обрати той варіант, який буде відповідати встановленому нами стандарту. По суті справи це абсолютно вірно, але досить складно внести ясність у те, який повинен бути цей баланс. Саме це я намагатимуся визначити у своїй роботі.

ВИСНОВКИ

Розглядаючи проблему цієї роботи аналітичним та емпіричним шляхом, ми отримали, що якщо ми бажаємо збільшити свої шанси на отримання найкращого результату, нам необхідно приблизно 37% $\left(\frac{n}{e}\right)$ усіх варіантів виділити на ознайомлення з ними та встановлення стандарту, який же з них буде кращим. Після того, як ці 37% закінчились, ми повинні бути готові прийняти швидке рішення: обрати перший зустрічний варіант, який стане гідним конкурентом на фоні побачених раніше. Це не є компромісом із самим собою відносно процесу вибору і прийняття рішення. Це є оптимальною стратегією. Емпіричним шляхом знайшли цьому підтвердження.

У реальному житті ми часто зустрічаємо більш складні проблеми, коли ми не можемо однозначно сказати, який з варіантів буде кращим, тому що кожен варіант характеризується декількома властивостями.

Тому для цього було вирішено ускладнити цю задачу до багатокритеріальної. Були розглянуті різні варіанти, яким чином можливо звести багатокритеріальність до скалярного порівняння. Ці методи мають гнучке пристосування, тобто у залежності від умов задачі можна змінювати методи згортки та отримувати кращий результат. Для цього були зроблена програмна реалізація, яка емпірично шукає кращих кандидатів і виводить різними кольорами переможні й програшні перестановки – порядок виходу кандидатів до нареченої.

Результати роботи написаних мною програм демонструють, що незалежно від методу згортки багатокритеріальної задачі, коефіцієнтів та способу вибору кращого кандидату, рішення не змінюється, тобто кількість виграшних перестановок залишається однаковим для кожної вибірки k кандидатів, яких пропускаємо, із загальної кількості n кандидатів. Це означає, що стратегія пропускати саме $\frac{n}{e}$ кандидатів і обирати першого кращого є оптимальною – тією, що з найбільшою ймовірністю призведе до перемоги.

СПИСОК ЛІТЕРАТУРИ

1. Гусейн-Заде С.М., Разборчивая невеста, – М.: МЦНМО, 2003. — 24 с.
2. Лотов А.В., Поспелова И.И. Многокритериальные задачи принятия решений: Учебное пособие. – М.: МАКС Пресс, 2008. – 197 с.
3. Ногин В.Д. Принятие решений в многокритериальной среде: количественный подход. – М.: ФИЗМАТЛИТ, 2002. – 144 с.
4. Фергюсон Т.С., «Who solved the secretary problem?», Statistical Science 4 (3), 1989. – 282-289 с.
5. Фріман П., «The secretary problem and its extensions», International Statistical Review 51 (2), 1983. – 189-206 с.

ДОДАТОК А

Код програми

```
import java.math.BigInteger;
import java.util.Comparator;
import com.bride.Comparator.*;

public class BrideProblem
{
   private static Comparator gComparator;
   private static Groom[] groom;
   private static final int n = 4;

   public static BigInteger factorial(BigInteger n) {
      if(n.equals(BigInteger.ONE)) {
         return BigInteger.ONE;
      }
      return factorial(n.subtract(BigInteger.ONE)).multiply(n);
   }

   public static void swap(Groom[] array, int i, int j) {
      Groom temp = array[i];
      array[i] = array[j];
      array[j] = temp;
   }

   public static boolean nextSet(Groom[] array) {
      int j = array.length - 2;
      while (j != -1 && array[j].getId() >= array[j + 1].getId())
         j--;
      if (j == -1)
         return false;
      int k = array.length - 1;
      while (array[j].getId() >= array[k].getId())
         k--;
      swap(array, j, k);
      int l = j + 1, r = array.length - 1;
      while (l<r)
         swap(array, l++, r--);
      return true;
   }

   public static void print(int[] array) {
```

```
    for (int element : array) {
        System.out.print(element + " ");
    }
}

public static void print(Groom[] array, boolean full) {
    for (Groom element : array) {
        if(!full) {
            System.out.print(element.getId() + " ");
        } else {
            System.out.println(element.toString());
        }
    }
}

public static boolean optimalStrategy(Groom[] array, int sampleSize) {
    Groom best = new Groom(0), max = best;
    for(int i = 0; i < array.length; i++) {
        if(gComparator.compare(array[i], best) == 1) {
            best = array[i];
        }
    }

    for(int i = 0; i < sampleSize; i++) {
        if(gComparator.compare(array[i], max) == 1) {
            max = array[i];
        }
    }

    for(int i = sampleSize; i < array.length; i++) {
        if(gComparator.compare(array[i], max) == 1) {
            if(gComparator.compare(array[i], best) == 0) {
                return true;
            } else {
                return false;
            }
        }
    }
    return false;
}

public static void bridgeProblem() {
    int[] sampleSize = new int[n];
```

```
        BigInteger factorialOfN = factorial(BigInteger.valueOf(n));
        while((factorialOfN =
factorialOfN.subtract(BigInteger.ONE)).compareTo(BigInteger.ZERO) != -1) {
            for (int i = 0; i < sampleSize.length; i++) {
                print(groom, false);
                if (optimalStrategy(groom, i)) {
                    System.out.print(" " + 1);
                    sampleSize[i]++;
                } else {
                    System.out.print(" " + 0);
                }
                System.out.println("");
            }
            nextSet(groom);
        }
        print(sampleSize);
    }

    private static void initializeGroom() {
        groom = new Groom[n];
        for (int i = 0; i < n; i++) {
            groom[i] = new Groom(i + 1, new int[] {(i + 5)*(i + 5), (n + 1 - i) * 3, (i +
2) * (i + 2) * 4, (n + 3 - i)});
        }
    }

    /**
     * Main function. Uncomment line with necessary comparator
     */
    public static void main(String[] args) {
        //gComparator = new GroomIdComparator();
        //gComparator = new GroomIdealPointComparator(groom);
        //gComparator = new GroomLinearComparator(new double[] {0.1, 0.3, 0.1,
0.5});
        //gComparator = new GroomMainCriterionComparator(3, new int[] {0, 30});
        gComparator = new GroomMaxMinComparator();
        //gComparator = new GroomMultiplComparator(new double[] {0.2, 0.2, 0.2,
0.4});

        bridgeProblemNonOptimal();
    }
}
```

ДОДАТОК Б

Програма-парсер

```
import java.io.*;

public class BrideProblemParser {
   private int n = 4;

   public static void main(String[] args) {
      try (FileWriter writer = new FileWriter("res\\output.html", false)) {
         try (BufferedReader reader = new BufferedReader(new
FileReader("res\\input.txt"))) {
            String line;
            boolean sampleStart = false;

            writer.write("<html>\n");
            writer.write("<head>\n");
            writer.write("<style type=\"text/css\">\n" +
                  "    tr.win { background: #aaff80; }\n" +
                  "    tr.lose { background: #ff8080; }\n" +
                  "  </style>\n" +
                  " </head>\n" +
                  " <body>\n");

            while ((line = reader.readLine()) != null) {
               if (line.contains("SAMPLE SIZE")) {
                  if(sampleStart) {
                     writer.write("</table>\n");
                  }
                  sampleStart = true;
                  writer.write("<h2> Количество пропущенных \nженихов" +
line.substring(line.indexOf('=') + 1, line.length()) + "</h2>");
                  writer.write("<table table border=\"1\" cellpadding=\"1\"
cellspacing=\"1\">\n");
                  reader.readLine();
                  reader.readLine();
                  line = reader.readLine();
```

```
                }

                if (!line.equals("\n")) {

                    String result = line.substring(0, line.indexOf("-"));
                    String sol = line.substring(line.indexOf("-"), line.length());

                    if (sol.charAt(2) == '1') {
                        writer.write("<tr class=\"win\">\n" +
                                "    <td>");
                        writer.write(result + "</td>\n</tr>\n");
                    } else {
                        writer.write("<tr class=\"lose\">\n" +
                                "    <td>");
                        writer.write(result + "</td>\n</tr>\n");
                    }
                }
            }
            writer.write("</table>\n" +
                    "</body>\n" +
                    "</html>\n");
            writer.flush();
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
  }
}
```