Universidade do Minho

2ºSemestre 2019/20

(MIEI, 3º Ano)

Modelos Estocásticos de Investigação Operacional

Trabalho Prático

Identificação do Grupo

Número:	*Nome completo:*	*Rúbrica:*
A86012	Eduardo José Azevedo Ferreira Araújo	
A85731	Gonçalo José Azevedo Esteves	
A84306	João Miguel Vasconcelos Ferreira Mesquita de Araujo	
A83610	Rui Nuno Borges Cruz Oliveira	

Data de entrega: 2020-05-11

Conteúdo

1 Introdução **3**

2 Parte 1 **3**
2.1 Descrição do Problema 3
2.2 Formulação do Problema 4
2.3 Algoritmo de iteração de valor 4
2.4 Resultados 6
2.4.1 Matrizes de transição e contribuições 6
2.4.2 Solução ótima 6

3 Parte 2 **7**
3.1 Apresentação da Problemática 7
3.2 Utilidade do Modelo 7
3.3 Condições de Aplicação do Modelo 8
3.4 Aplicação do Modelo 8
3.5 Questões Respondidas pelo Modelo 9
3.6 Referências 9

4 Conclusão **9**

5 Anexos **10**
5.1 A1 - Dados fornecidos 10
5.2 A2 - Matrizes de transição 10
5.3 A3 - Código Java 10
5.4 A4 - Resultados obtidos para a última iteração 19

1 Introdução

Integrado na unidade curricular MEIO - Modelos Estocásticos de Investigação Operacional, foi desenvolvido o trabalho prático que consistiu na resolução de dois problemas, sendo um fornecido pela equipa docente e outro pela nossa investigação.

Este relatório encontra-se divido em duas partes, sendo que na primeira parte tem como objectivo consolidar os assuntos abordados nas aulas lecionadas através de um problema de Programação Dinâmica Estocástica - problema de decisão com número indeterminado de estágios. Assim, para o resolver, optamos por escrever um algoritmo na linguagem de programação Java, de forma a auxiliar-nos com a iteração de valores.

2 Parte 1

2.1 Descrição do Problema

Um empresário gere duas filiais de um grupo internacional de aluguer de automóveis. Durante o dia, clientes chegam às filiais e são atendidos enquanto há disponibilidade de automóveis para alugar, sendo que o empresário é creditado em 30 euros por cada veículo alugado. Uma vez que as filiais estão localizadas em cidades diferentes, se qualquer uma delas deixa de ter automóveis disponíveis, não pode utilizar o stock da outra, pelo que os clientes que chegam entretanto acabam por se dirigir a empresas concorrentes e o empresário incorre numa situação de "perda de vendas". No entanto, são entregues automóveis diariamente nas filiais pelos clientes, sendo que estes ficam apenas disponíveis para alugar no dia seguinte.

Cada filial não pode acumular mais do que 12 automóveis no final de cada dia, sendo que o eventual excesso de veículos é reencaminhado para outras filiais cuja gerência não pertence ao empresário. Por outro lado, o empresário pode também reajustar os stocks de automóveis nas suas filiais, transferindo de uma para a outra, tendo esta operação um custo de 7 euros por automóvel e com limte máximo de 3 transferêcias por dia.

Se mais do que 8 automóveis tiverem de ser guardados durante a noite, em cada filial, é necessário utilizar um espaço extra que custa ao empresário uma taxa de 10 por noite de utilização (taxa fixa), i.e. não depende do número de viaturas guardadas nesse espaço.

O nosso objetivo é, do ponto de vista do empresário, determinar a política ótima de transferência diária de automóveis entre as duas filiais, de modo a minimizar o prejuízo de situações como a "perda de vendas".

2.2 Formulação do Problema

Este trabalho é relativo a um problema com alternativas e um número infinito de estágios, modelado com base num modelo de Programação Dinâmica Estocástica. Portanto, foi necessário, inicialmente, definir os estágios, estados e ações alternativas do problema, para proceder à resolução do mesmo.

- **Estados:** número de carros em cada filial em formato de dupleto (A,B), sendo A o nº de carros na filial 1 e B o nº de carros na filial 2.
- **Estágios:** final de cada dia(ilimitados).
- **Ações Alternativas:** não fazer transferência, 1 transferência feita pela filial, 1 transferência feita pela filial 2, 2 transferências feitas pela filial 1, 2 transferências feitas pela filial 2, 3 transferências feitas pela filial 1 e 3 tranferências feitas pela filial 2.

Foram contabilizados um total de 169 estados.

Após isto, definimos a matriz de transição, contribuições e esperanças. Selecionando o valor ótimo (neste caso o mínimo, pois queremos minimizar o prejuízo das "perdas de vendas", podemos obter as melhores decisões a ser tomadas, tendo em conta o estado anterior. Para chegar a esse valor, recorremos a um algoritmo de iteração de valor.

2.3 Algoritmo de iteração de valor

O algoritmo utilizado, faz com que se itere até que a variação do ganho seja inferior a 0.1. A cada iteração é calculada uma nova matriz Fn, seleccionando para isso as alternativas que o minimizam, sendo que quando o erro é ultrapassado, as decisões podem ser tidas em conta. Apresentamos agora em baixo o código responsável pela iteração, sendo que a função *verifica* trata de verificar se o erro foi ou não ultrapassado.

```
1         do{
              for(int j = 0; j <= 168; j++)
3                 Fn_anterior[j][0] = Fn[j][0];

5             Vn_0T = soma_matrizes(this.Qn_0T, multiplica_matrizes(this.P,
              Fn_anterior));
7             Vn_1T_from1 = soma_matrizes(this.Qn_1faz1, multiplica_matrizes(
    this.P,
              Fn_anterior));
9             Vn_1T_from2 = soma_matrizes(this.Qn_2faz1, multiplica_matrizes(
    this.P,
              Fn_anterior));
11            Vn_2T_from1 = soma_matrizes(this.Qn_1faz2, multiplica_matrizes(
    this.P,
              Fn_anterior));
```

```
13              Vn_2T_from2 = soma_matrizes(this.Qn_2faz2, multiplica_matrizes(
    this.P,
                Fn_anterior));
15              Vn_3T_from1 = soma_matrizes(this.Qn_1faz3, multiplica_matrizes(
    this.P,
                Fn_anterior));
17              Vn_3T_from2 = soma_matrizes(this.Qn_2faz3, multiplica_matrizes(
    this.P,
                Fn_anterior));
19
                for(int j = 0; j <= 168; j++) {
21                  Fn[j][0] = minimo(Vn_0T[j][0], Vn_1T_from1[j][0], Vn_1T_from2[
    j][0],
                    Vn_2T_from1[j][0], Vn_2T_from2[j][0], Vn_3T_from1[j][0],
    Vn_3T_from2[j][0]);
23                  decisoes[j] = index(Vn_0T[j][0], Vn_1T_from1[j][0],
    Vn_1T_from2[j][0],
                    Vn_2T_from1[j][0], Vn_2T_from2[j][0], Vn_3T_from1[j][0],
    Vn_3T_from2[j][0]);
25                  Dn[j][0] = Fn[j][0] - Fn_anterior[j][0];
                }
27
                System.out.println("*** ITERACAO n"+ i + " ****");
29              System.out.println("* Fn *");
                for(int j = 0; j <= 168;j++){
31                  System.out.println("[" + df.format(Fn[j][0]) + "]");
                }
33              System.out.println("* Dn *");
                for(int j = 0; j <= 168;j++){
35                  System.out.println("[" + df.format(Dn[j][0]) + "]");
                }
37              int contador = 0;
                for(int f1 = 0; f1 <= 12; f1++) {
39                  for(int f2 = 0; f2 <= 12; f2++){
                        System.out.println("Para o estado (" + f1 + ", " + f2 + ")
    : " +
41                      decisoes[contador]);
                        contador++;
43                  }
                }
45
                i++;
47          } while(verifica(Dn));
        }
```

2.4 Resultados

2.4.1 Matrizes de transição e contribuições

Foi utilizada uma matriz de transição de estados, **P**, em que cada duplo ao transitar para outro foi considerado que, sendo *(A,B)* de onde transita e *(C,D)* para onde trasita, teríamos de multiplicar a soma das probabilidades de haver '*0 ou (C-A) até 12 pedidos*' e '*0 ou (A-C) até C ou 12 entregas*', com a soma das probabilidades de haver ' *0 ou (B-D) até 12 pedidos*' e '*0 ou (D-B) até D ou 12 entregas*', dependendo da transição.

Para a matriz de contribuições, isto é, o ganho entre estados, **R**, consideramos que, não havendo transferências, o ganho entre estados é nulo para quando o nº de carros ao final do dia em cada filial é inferior a 8. Quando tal não acontece, o ganho, por cada filial, é 10. No total, foram 7 as matrizes de contreibuições, uma por cada ação alternativa.

Caso haja transferência(s), verificamos se a filial que faz a(s) transferência(s) tem carros suficientes para realmente ser(em) executada(s) e se a filial que a(s) recebe(s) pode efetivamente recebê-la(s), ou seja, se tem espaço. Reunidas as condições necessárias, por cada transferência feita por determinada filial o ganho é 7, sendo que, assim como para quando não há transferências, temos que considerar também a taxa por utilização do espaço extra de estacionamento, ou seja, 10 + 7*nºtransferências, havendo transferência, e 10, não existindo transferência.

Podemos analisar estas matrizes com maior detalhe no excel fornecido.

2.4.2 Solução ótima

Ao fim de 12 iterações(utilizando o algoritmo apresentado em 2.3), obtivemos a solução do problema, sugerindo que para todos os estados não deverão ser efetuadas transferências entre filiais.

De um ponto de vista inicial, faria sentido considerar outras soluções, como a solução de 1 transferência da filial 1 para 2, por exemplo, quando terminamos o dia com 9 carros na filial 1 e 7 na filial 2, porém, com a resolução do problema, podemos concluir que, sendo esse um dos casos que à partida poderia fazer sentido como sendo a decisão válida a tomar, tem uma probabilidade baixa.

Na secção dos Anexos A4) apresentamos a solução ótima com maior detalhe, podendo observar a matriz *Fn* e *Dn*, através dos resultados obtidos.

3 Parte 2

3.1 Apresentação da Problemática

As auroras são fenómenos naturais que ocorrem nas camadas mais elevadas da atmosfera devido à colisão de partículas elétricas (provenientes dos ventos solares e atraídas pelo campo magnético da Terra) com átomos de oxigénio e azoto. Estas colisões geram a emissão de radiações com variados comprimentos de onda, que por sua vez são o motivo da existência das diversas cores características das auroras. Quer estejamos no Pólo Norte e observemos as auroras bureais, ou verifiquemos a existência das auroras astrais aquando de uma visita ao Pólo Sul, é inegável toda a beleza que este espetáculo de luzes naturais proporciona.

Por forma a analisar melhor o que será o comportamento e a evolução esperados de uma aurora, é muito mais relevante o uso de informação temporal em deterimento do uso de imagens capturadas de forma estática. Tendo isto em conta, e considerando também que a análise desta forma de informação é mais complexa, este trabalho serve para apresentar um método de representação baseado num *Hidden Markov Model* (HMM) que se pretende usar para caracterizar sequências de imagens de auroras capturadas por *all-sky imagers* (ASIs).

Podemos considerar um HMM como um autómato de estados finitos onde ocorrem dois processos estocásticos concorrentemente: um é o gerador de uma cadeia oculta de Markov; o outro é o gerador de processos aleatórios.

De modo a tornar o modelo representado mais eficiente, recorreu-se ao uso de certas características de textura espacial bem como a técnicas de evolução dinámica.

3.2 Utilidade do Modelo

Está comprovado que as auroras e os seus tipos estão intrinsecamente relacionados com a magnetosfera e toda a sua atividade dinâmica, bem como com variações nos ventos solares. Assim, o estudo das auroras e a sua consequente classificação são importantes uma vez que providenciam um método de observação do plasma ionosférico da Terra.

Devido à natureza duplamente estocástica dos HMMs (são compostos por um processo estocástico não observável que pode ser inferido através de outro processo estocástico observável), é útil o uso de um destes para a resolução do problema, uma vez que a evolução de uma aurora envolve dois processos estocásticos: a colisão das partículas da magnetosfera da Terra com os ventos solares, que não é diretamente visível mas pode ser inferida através da luz polar que aparece no céu.

A importância do desenvolvimento deste modelo recai no facto de que uma aurora é um processo dinâmico, que evolui com o passar do tempo e, como tal, o uso de imagens sequenciais é em tudo superior ao uso de imagens estáticas, uma vez que estas não providenciam informações relativas ao contexto temporal, que são fulcrais na distinção dos diferentes tipos de auroras.

3.3 Condições de Aplicação do Modelo

Por forma a validar o modelo, foram aplicados três testes diferentes, de crescente complexidade. Inicialmente, foram realizadas experiências mais simples capazes de demonstrar as potencialidades do método proposto. De seguida, realizaram-se experiências supervisionadas capazes de classificar quantitativamente imagens obtidas em 2003 por um ASI na estação de Yellow River, na China (a comparação dos resultados obtidos, com aqueles obtidos por outros métodos já utilizados, evidencia não só uma maior precisão como também uma menor taxa de rejeição). Por fim, utilizou-se o modelo desenvolvido por forma a classificar 21787 sequências, obtidas entre 2004 e 2009 por ASI's, usando como base imagens obtidas em 2003 (estas foram utilizadas antes das mais recentes por forma a dotar o modelo de uma melhor capacidade de reconhecimento dos diversos padrões e caracaterísticas das auroras). Os resultados obtidos eram concordantes com os obtidos por outros modelos já válidos, o que é uma potencial prova da validade do modelo.

3.4 Aplicação do Modelo

Uma *feature* é um propriedade simbólica ou numérica de um dado objeto. Assim, a extração de *features* estáticos é importante para o desenvolvimento do modelo, uma vez que permite a captura de dados estáticos e estruturais.

Por forma a representar as imagens captadas pelos ASIs recorreu-se a *uniform Local Binary Patterns* (uLBPs), guardando cada um a informação de uma *feature* específica dessa imagem. Uma imagem será então representada na totalidade através de um vetor de *features* uLBP.

A definição dos estados ocultos do modelo, que recorreu a conhecimentos de estudos previamente elaborados, foi realizada em duas partes: primeiro, determinou-se que 3 seria o número ótimo de estados; em seguida, distribuiram-se todas as imagens, ou seja, todos os vetores de *features*, por *clusters*, tendo em conta a ordem cronológica das imagens, e considerou-se cada *cluster* como sendo um estado.

Relativamente aos estados observáveis, dependentes dos estados ocultos, estes irão corresponder a observações feitas a um dado estado oculto, baseando-se numa função de probabilidade.

3.5 Questões Respondidas pelo Modelo

Uma classificação sequencial como a implementada neste modelo permite uma obtenção de resultados mais eficaz, tornando a classificação da auroras mais precisa, já que um processo dinâmico e que está em constante evolução é melhor avaliado através do recurso a informação temporal, em deterimento do uso de imagens estáticas, por exemplo.

Comparativamente a métodos previamente criados, tais como métodos baseados em *frames*, o modelo desenvolvido evidencia uma performance superior a nível de classificação de auroras tendo em conta os diversos testes realizados. Para além disto, este método não permite apenas a captura da estrutura espacial das imagens obtidas pelos ASIs, mas também tem em consideração as ocorrências temporais que ocorrem com o passar do tempo numa aurora.

3.6 Referências

- Yang Q., Liang J., Hu Z., and Zhao H. (2012). Auroral Sequence Representation and Classification Using Hidden Markov Models. IEEE Transactions on Geoscience and Remote Sensing, 50(12), 5049-5060. doi: https://doi.org/10.1109/TGRS.2012.2195667

4 Conclusão

Através deste trabalho foi possível enriquecer e consolidar conhecimentos relativamente a Programação Dinâmica - neste caso referente a um problema de cesião com nº indeterminado de estágios. Utilizando a linguagem de programação Java conseguimos implementar o algoritmo de iteração solicitado como auxiliar do problema.

Em suma, após alguma dificuldade inicial em delinear quais os estados e estágios, e, consequentemente, as matrizes de transição e de contribuições, após várias formulações, consideramos ter delineado de forma correta a base do problema e chegado ao obejtivo pretendido.

5 Anexos

5.1 A1 - Dados fornecidos

```
Grupo que inclui o Aluno com o Nº 85731
MEIO-TP1 - Tabelas de probabilidades de pedidos e entregas de automóveis

FILIAL 1
Número de clientes:   ;      0 ;      1 ;      2 ;      3 ;      4 ;      5 ;      6 ;      7 ;      8 ;
9 ;     10 ;     11 ;     12
Probabilidade (pedidos):  ; 0.0488 ; 0.0860 ; 0.1396 ; 0.1432 ; 0.1200 ; 0.1076 ; 0.1012 ; 0.0860 ; 0.0568 ;
0.0436 ; 0.0344 ; 0.0264 ; 0.0064
Probabilidade (entregas): ; 0.0396 ; 0.0816 ; 0.1188 ; 0.1484 ; 0.1388 ; 0.1116 ; 0.0892 ; 0.0832 ; 0.0676 ;
0.0616 ; 0.0336 ; 0.0208 ; 0.0052

FILIAL 2
Número de clientes:   ;      0 ;      1 ;      2 ;      3 ;      4 ;      5 ;      6 ;      7 ;      8 ;
9 ;     10 ;     11 ;     12
Probabilidade (pedidos):  ; 0.0800 ; 0.2020 ; 0.2396 ; 0.2152 ; 0.1392 ; 0.0728 ; 0.0316 ; 0.0132 ; 0.0052 ;
0.0008 ; 0.0000 ; 0.0004 ; 0.0000
Probabilidade (entregas): ; 0.0116 ; 0.0644 ; 0.1228 ; 0.1756 ; 0.1896 ; 0.1632 ; 0.1188 ; 0.0820 ; 0.0392 ;
0.0192 ; 0.0092 ; 0.0028 ; 0.0016
```

Figura 1: Dados obtidos a partir da folha de cálculo "meio_tp1_g35.dat"com o número de aluno 85731

5.2 A2 - Matrizes de transição

As matrizes de transição (probabilidades e contribuições) relativas a cada uma das filiais (em separado) e a cada uma das decisões alternativa encontram-se no excel fornecido.

5.3 A3 - Código Java

```
import java.text.DecimalFormat;
import java.util.Arrays;

public class GIAluguer {
    /* Probabilidade(matriz transicao) apos um dia de trabalho */
    private final double[][] P;

    /* Ganhos(matriz transicao) apos um dia de trabalho */
    private final double[][] R_0T;
    private final double[][] R_1faz1;
    private final double[][] R_2faz1;
    private final double[][] R_1faz2;
    private final double[][] R_2faz2;
    private final double[][] R_1faz3;
    private final double[][] R_2faz3;

    /* Esperanca de Custos */
```

```
18      private final double[][] Qn_0T;
        private final double[][] Qn_1faz1;
20      private final double[][] Qn_2faz1;
        private final double[][] Qn_1faz2;
22      private final double[][] Qn_2faz2;
        private final double[][] Qn_1faz3;
24      private final double[][] Qn_2faz3;

26      private DecimalFormat df = new DecimalFormat("#.####");
        private DecimalFormat df2 = new DecimalFormat("#.##");
28      private final double ERRO = 0.1;


30
        public GIAluguer(double[] p1_pedidos, double[] p1_entregas, double[]
    p2_pedidos,
32      double[] p2_entregas) {
            /* Matrizes transicao */
34          this.P = matrizTransicao(p1_pedidos, p1_entregas, p2_pedidos,
    p2_entregas);

36          /* 0 transferencias(R's e Q's) */
            this.R_0T = matrizGanhos(0,1);
38          this.Qn_0T = matriz_Q(this.P, R_0T);

40          /* 1 transferencia(R's e Q's)  */
            this.R_1faz1 = matrizGanhos(1,1);
42          this.R_2faz1  = matrizGanhos(1, 2);
            this.Qn_1faz1 = matriz_Q(this.P, this.R_1faz1);
44          this.Qn_2faz1  = matriz_Q(this.P, this.R_2faz1);

46          /* 2 transferencias(R's e Q's)  */
            this.R_1faz2 = matrizGanhos(2,1);
48          this.R_2faz2  = matrizGanhos(2, 2);
            this.Qn_1faz2 = matriz_Q(this.P, this.R_1faz2);
50          this.Qn_2faz2  = matriz_Q(this.P, this.R_2faz2);

52          /* 3 transferencias(R's e Q's)  */
            this.R_1faz3 = matrizGanhos(3,1);
54          this.R_2faz3  = matrizGanhos(3, 2);
            this.Qn_1faz3 = matriz_Q(this.P, this.R_1faz3);
56          this.Qn_2faz3  = matriz_Q(this.P, this.R_2faz3);
        }
58
        public double[][] matrizTransicao(double[] p1_pedidos, double[]
    p1_entregas,
60      double[] p2_pedidos, double[] p2_entregas){
            int x, y, i, j;
62          double[][] matriz = new double[169][169];
```

```
        double p1 = 0, p2 = 0;
64      for(int duplosX = 0; duplosX <= 12; duplosX++) {
            int desdeX = duplosX * 13;
66          int ateX   = desdeX + 12;
            for (x = desdeX; x <= ateX; x++) {
68              for (int duplosY = 0; duplosY <= 12; duplosY++) {
                    int desdeY = duplosY * 13;
70                  int ateY   = desdeY + 12;
                    for(y = desdeY; y <= ateY; y++){
72
                        if(duplosY == 12)
74                          for(i = 0; i <= duplosX; i++)
                                for (j = duplosY-duplosX+i; j<=12;j++)
76                                  p1 += p1_pedidos[i] *p1_entregas[j];
                        if(duplosX < duplosY && duplosY != 12){
78                          j = duplosY - duplosX;
                            for(i = 0; i <= 12; i++){
80                              p1 += p1_pedidos[i] * p1_entregas[j];
                                if(j < duplosY) j++;
82                          }
                        }
84                      if(duplosX >= duplosY && duplosY != 12){
                            j = 0;
86                          for(i = duplosX - duplosY; i <= 12; i++){
                                p1 += p1_pedidos[i] * p1_entregas[j];
88                              if(j < duplosY) j++;
                            }
90                      }

92                      if((y-desdeY) == 12)
                            for(i = 0; i <= (x - desdeX) ; i++)
94                              for(j = (y-desdeY)-(x-desdeX)+i; j <= 12; j++)
                                    p2 += p2_pedidos[i] * p2_entregas[j];
96                      if((x-desdeX) < (y-desdeY) && (y-desdeY) != 12){
                            j = (y-desdeY) - (x-desdeX);
98                          for(i = 0; i <= 12; i++){
                                p2 += p2_pedidos[i] * p2_entregas[j];
100                             if(j < (y-desdeY)) j++;
                            }
102                     }
                        if((x-desdeX) >= (y-desdeY) && (y-desdeY) != 12){
104                         j = 0;
                            for(i = (x-desdeX) - (y-desdeY); i <= 12; i++){
106                             p2 += p2_pedidos[i] * p2_entregas[j];
                                if(j < (y-desdeY)) j++;
108                         }
                        }
110
```

```
                        matriz[x][y] = p1 * p2;
                        p1 = 0; p2 = 0;
                    }
                }
            }
        }
        return matriz;
    }

    public double[][] matrizGanhos(int transferencias, int faz){
        int x, y;
        double[][] matriz = new double[169][169];
        double p1 = 0, p2 = 0;
        if(transferencias == 0){
            for(int duplosX = 0; duplosX <= 12; duplosX++) {
                int desdeX = duplosX * 13;
                int ateX   = desdeX + 12;
                for (x = desdeX; x <= ateX; x++) {
                    for (int duplosY = 0; duplosY <= 12; duplosY++) {
                        int desdeY = duplosY * 13;
                        int ateY   = desdeY + 12;
                        for(y = desdeY; y <= ateY; y++) {
                            if(duplosY > 8)
                                p1 = 10;
                            if((y-desdeY) > 8)
                                p2 = 10;
                            matriz[x][y] = p1 + p2;
                            p1 = 0; p2 = 0;
                        }
                    }
                }
            }
        }
        else{
            if(faz == 1) {
                for (int duplosX = 0; duplosX <= 12; duplosX++) {
                    int desdeX = duplosX * 13;
                    int ateX = desdeX + 12;
                    for (x = desdeX; x <= ateX; x++) {
                        for (int duplosY = 0; duplosY <= 12; duplosY++) {
                            int desdeY = duplosY * 13;
                            int ateY = desdeY + 12;
                            for (y = desdeY; y <= ateY; y++) {
                                p1 = 0;
                                p2 = 0;
                                if(duplosY == 0 || (duplosY == 1 &&
                                (transferencias == 2 || transferencias == 3))
    ||
```

```
158                                 (duplosY == 2 && transferencias == 3))
                                        p1 = 0;
160                                 else{
                                        if((y-desdeY) > 12-transferencias) {
162                                         if (duplosY > 8)
                                                p1 = 10;
164                                         else
                                                p1 = 0;
166                                     }
                                        else{
168                                         if(duplosY > 8 + transferencias)
                                                p1 = 10 + 7*transferencias;
170                                         else
                                                p1 = 7 * transferencias;
172                                     }
                                    }
174                                 if ((p1 != 0 && p1 != 10)) {
                                        if ((y - desdeY) > 8 - transferencias)
176                                         p2 = 10;
                                    }
178                                 else {
                                        if ((y - desdeY) > 8)
180                                         p2 = 10;
                                    }
182                                 matriz[x][y] = p1 + p2;
                                }
184                         }
                        }
186                 }
            }
188         else{
                for (int duplosX = 0; duplosX <= 12; duplosX++) {
190                 int desdeX = duplosX * 13;
                    int ateX = desdeX + 12;
192                 for (x = desdeX; x <= ateX; x++) {
                        for (int duplosY = 0; duplosY <= 12; duplosY++) {
194                         int desdeY = duplosY * 13;
                            int ateY = desdeY + 12;
196                         for (y = desdeY; y <= ateY; y++) {
                                p1 = 0;
198                             p2 = 0;
                                if((y-desdeY) == 0 || ((y-desdeY) == 1 &&
200                             (transferencias == 2 || transferencias == 3))
    ||
                                ((y-desdeY) == 2 && transferencias == 3))
202                                 p2 = 0;
                                else{
```

```
204                                     if(duplosY > 12-transferencias) {

        /* Transferencia nao acontece( a outra filial nao tem capacidade para
    receber) */
                                         if ((y-desdeY) > 8)
206                                          p2 = 10;
                                         else
208                                          p2 = 0;
                                     }
210                                  else{
                                         if((y-desdeY) > 8 + transferencias)
212                                          p2 = 10 + 7*transferencias;
                                         else
214                                          p2 = 7 * transferencias;
                                     }
216                                }
                                   if ((p2 != 0 && p2 != 10)) {
218                                    if (duplosY > 8 - transferencias)
                                           p1 = 10;
220                                }
                                   else {
222                                    if (duplosY > 8)
                                           p1 = 10;
224                                }
                                   matriz[x][y] = p1 + p2;
226                           }
                          }
228                   }
              }
230       }
      }
232
      return  matriz;
234   }

236   public double[][] matriz_Q(double[][] P, double [][] R){
        int x, y;
238     double[][] matriz = new double[169][1];
        for(x = 0; x <= 168; x++)
240       for(y = 0; y <= 168; y++)
            matriz[x][0] += P[x][y] * R[x][y];
242     return matriz;
      }
244   public double[][] soma_matrizes(double[][] Q1, double[][] Q2){
        int x;
246     double[][] matriz = new double[169][1];
        for(x = 0; x <= 168; x++)
248       matriz[x][0] = Q1[x][0]+ Q2[x][0];
```

```
        return matriz;
250     }
        public double[][] multiplica_matrizes(double[][] P, double[][] F){
252         int x;
            double[][] matriz = new double[169][1];
254         for(x = 0; x <= 168; x++)
                for(int i = 0; i <= 168; i++)
256                 matriz[x][0] += P[x][i] * F[i][0];
            return matriz;
258     }

260     public void resolver(){
            double[][] Fn = new double[169][1];
262         double[][] Fn_anterior = new double[169][1];
            double[][] Vn_0T;
264         double[][] Vn_1T_from1;
            double[][] Vn_1T_from2;
266         double[][] Vn_2T_from1;
            double[][] Vn_2T_from2;
268         double[][] Vn_3T_from1;
            double[][] Vn_3T_from2;
270         double[][] Dn = new double[169][1];
            String[] decisoes = new String[169];
272
            int i = 0;
274         do{
                for(int j = 0; j <= 168; j++)
276                 Fn_anterior[j][0] = Fn[j][0];

278             Vn_0T = soma_matrizes(this.Qn_0T, multiplica_matrizes(this.P,
                Fn_anterior));
280             Vn_1T_from1 = soma_matrizes(this.Qn_1faz1, multiplica_matrizes(
    this.P,
                Fn_anterior));
282             Vn_1T_from2 = soma_matrizes(this.Qn_2faz1, multiplica_matrizes(
    this.P,
                Fn_anterior));
284             Vn_2T_from1 = soma_matrizes(this.Qn_1faz2, multiplica_matrizes(
    this.P,
                Fn_anterior));
286             Vn_2T_from2 = soma_matrizes(this.Qn_2faz2, multiplica_matrizes(
    this.P,
                Fn_anterior));
288             Vn_3T_from1 = soma_matrizes(this.Qn_1faz3, multiplica_matrizes(
    this.P,
                Fn_anterior));
290             Vn_3T_from2 = soma_matrizes(this.Qn_2faz3, multiplica_matrizes(
    this.P,
```

```
            Fn_anterior));
292
            for(int j = 0; j <= 168; j++) {
294             Fn[j][0] = minimo(Vn_0T[j][0], Vn_1T_from1[j][0], Vn_1T_from2[
    j][0],
                Vn_2T_from1[j][0], Vn_2T_from2[j][0], Vn_3T_from1[j][0],
296             Vn_3T_from2[j][0]);
                decisoes[j] = index(Vn_0T[j][0], Vn_1T_from1[j][0],
    Vn_1T_from2[j][0],
298             Vn_2T_from1[j][0], Vn_2T_from2[j][0], Vn_3T_from1[j][0],
                Vn_3T_from2[j][0]);
300             Dn[j][0] = Fn[j][0] - Fn_anterior[j][0];
            }
302
            System.out.println("*** ITERACAO n"+ i + " ****");
304         System.out.println("* Fn *");
            for(int j = 0; j <= 168;j++){
306             System.out.println("[" + df.format(Fn[j][0]) + "]");
            }
308         System.out.println("* Dn *");
            for(int j = 0; j <= 168;j++){
310             System.out.println("[" + df.format(Dn[j][0]) + "]");
            }
312         int contador = 0;
            for(int f1 = 0; f1 <= 12; f1++) {
314             for(int f2 = 0; f2 <= 12; f2++){
                    System.out.println("Para o estado (" + f1 + ", " + f2 + ")
    : " +
316                 decisoes[contador]);
                    contador++;
318             }
            }
320         i++;
        } while(verifica(Dn));
322 }

324 public boolean verifica(double[][] Dn){
        for(int i=1; i <= 168;i++){
326         if(Math.abs(Dn[i][0] - Dn[i-1][0]) > ERRO)
                return  true;
328     }
        return false;
330 }

332 public double minimo(double a, double b, double c, double d, double e,
    double f, double g){
        return Math.min(a, Math.min(b, Math.min(c, Math.min(d,
334     Math.min(e, Math.min(f, g))))));
```

```
336     }

338     public String index(double a, double b, double c, double d, double e,
        double f, double g){
            double m = minimo(a,b,c,d,e,f,g);
340         if(a == m) return "0 transferencias";
            else if(b == m) return "1 transferencia da filial 1 para a 2";
342         else if(c == m) return "1 transferencia da filial 2 para a 1";
            else if(d == m) return "2 transferencias da filial 1 para a 2";
344         else if(e == m)  return "2 transferencias da filial 2 para a 1";
            else if(f == m) return "3 transferencias da filial 1 para a 2";
346         else if(g == m) return "3 transferencias da filial 2 para a 1";
            return "erro";
348     }
    }
350
    public class Main {
352     public static void main(String args[]){
            double[] filial1_pedidos = new double[]{0.0488, 0.0860, 0.1396,
        0.1432, 0.1200,
354         0.1076, 0.1012, 0.0860, 0.0568, 0.0436, 0.0344,0.0264,0.0064};
            double[] filial1_entregas = new double[]{0.0396, 0.0816, 0.1188,
        0.1484 ,
356         0.1388 , 0.1116, 0.0892, 0.0832, 0.0676, 0.0616 , 0.0336, 0.0208,
        0.0052};
            double[] filial2_pedidos = new double[]{0.0800, 0.2020, 0.2396,
        0.2152, 0.1392,
358         0.0728, 0.0316, 0.0132, 0.0052, 0.0008, 0.0000, 0.0004, 0.0000};
            double[] filial2_entregas = new double[]{0.0116, 0.0644, 0.1228,
        0.1756,
360         0.1896, 0.1632, 0.1188, 0.0820, 0.0392, 0.0192, 0.0092, 0.0028,
        0.0016};

362         GIAluguer x = new GIAluguer(filial1_pedidos, filial1_entregas,
            filial2_pedidos, filial2_entregas);
364         x.resolver();
        }
366 }
```

5.4 A4 - Resultados obtidos para a última iteração

```
 *** ITERAÇÃO n°12 ****
* Fn *
[146.3342]
[146.6861]
[147.8434]
[150.0287]
[153.2087]
[157.0992]
[161.3246]
[165.4825]
[169.2036]
[172.2095]
[174.3806]
[175.7834]
[176.596]
[146.5149]
[146.8671]
[148.025]
[150.2113]
[153.3929]
[157.2853]
[161.5127]
[165.6725]
[169.3954]
[172.4027]
[174.5748]
[175.9782]
[176.7912]
[146.9433]
[147.2958]
[148.4548]
[150.6431]
[153.8273]
[157.7228]
[161.9536]
[166.1167]
[169.8425]
[172.8522]
```

```
[175.0259]
[176.4305]
[177.2442]
[147.7102]
[148.0632]
[149.2235]
[151.414]
[154.6012]
[158.5003]
[162.7348]
[166.9015]
[170.6305]
[173.6428]
[175.8184]
[177.2242]
[178.0385]
[148.8339]
[149.1876]
[150.3493]
[152.542]
[155.7322]
[159.6346]
[163.8726]
[168.0427]
[171.7746]
[174.7892]
[176.9665]
[178.3734]
[179.1884]
[150.3028]
[150.6573]
[151.8205]
[154.0155]
[157.2086]
[161.1144]
[165.3558]
[169.5291]
[173.2639]
[176.2807]
[178.4596]
```

[179.8675]
[180.6831]
[152.1066]
[152.4619]
[153.6267]
[155.8243]
[159.0204]
[162.9295]
[167.1743]
[171.3508]
[175.0884]
[178.1074]
[180.2879]
[181.6969]
[182.5131]
[154.2024]
[154.5587]
[155.7253]
[157.9253]
[161.1243]
[165.0366]
[169.2845]
[173.464]
[177.2041]
[180.2251]
[182.407]
[183.817]
[184.6337]
[156.4852]
[156.8424]
[158.0107]
[160.2128]
[163.4144]
[167.3292]
[171.5797]
[175.7615]
[179.5036]
[182.5262]
[184.7092]
[186.1199]

[186.9371]
[158.8068]
[159.165]
[160.3347]
[162.5389]
[165.7427]
[169.6599]
[173.9125]
[178.0963]
[181.84]
[184.8639]
[187.048]
[188.4592]
[189.2768]
[161.03]
[161.3891]
[162.5603]
[164.7663]
[167.9721]
[171.8912]
[176.1457]
[180.3312]
[184.0763]
[187.1013]
[189.2861]
[190.6979]
[191.5158]
[163.055]
[163.4148]
[164.5873]
[166.7949]
[170.0023]
[173.9231]
[178.1792]
[182.3661]
[186.1124]
[189.1383]
[191.3238]
[192.736]
[193.5541]

[164.8032]
[165.1637]
[166.3372]
[168.5459]
[171.7546]
[175.6765]
[179.9336]
[184.1212]
[187.8683]
[190.8947]
[193.0805]
[194.493]
[195.3113]
* Dn *
[14.0689]
[14.0745]
[14.0835]
[14.0946]
[14.1061]
[14.1168]
[14.1261]
[14.1338]
[14.1398]
[14.1442]
[14.1473]
[14.1492]
[14.1504]
[14.076]
[14.0816]
[14.0905]
[14.1017]
[14.1132]
[14.1239]
[14.1332]
[14.1409]
[14.1469]
[14.1513]
[14.1543]
[14.1563]
[14.1575]

[14.0882]
[14.0938]
[14.1028]
[14.1139]
[14.1255]
[14.1362]
[14.1455]
[14.1532]
[14.1592]
[14.1636]
[14.1666]
[14.1686]
[14.1698]
[14.1027]
[14.1083]
[14.1172]
[14.1284]
[14.1399]
[14.1507]
[14.16]
[14.1677]
[14.1737]
[14.1781]
[14.1812]
[14.1832]
[14.1844]
[14.1173]
[14.1229]
[14.1319]
[14.143]
[14.1546]
[14.1653]
[14.1747]
[14.1823]
[14.1883]
[14.1928]
[14.1959]
[14.1979]
[14.1991]
[14.1324]

[14.138]
[14.147]
[14.1582]
[14.1697]
[14.1805]
[14.1898]
[14.1975]
[14.2035]
[14.208]
[14.2111]
[14.2131]
[14.2143]
[14.1481]
[14.1537]
[14.1627]
[14.1739]
[14.1855]
[14.1963]
[14.2056]
[14.2133]
[14.2193]
[14.2237]
[14.2268]
[14.2288]
[14.23]
[14.1634]
[14.169]
[14.178]
[14.1893]
[14.2008]
[14.2116]
[14.221]
[14.2287]
[14.2347]
[14.2391]
[14.2422]
[14.2442]
[14.2454]
[14.1769]
[14.1825]

[14.1915]
[14.2028]
[14.2144]
[14.2252]
[14.2345]
[14.2422]
[14.2483]
[14.2527]
[14.2558]
[14.2578]
[14.259]
[14.1891]
[14.1948]
[14.2038]
[14.215]
[14.2266]
[14.2374]
[14.2468]
[14.2545]
[14.2606]
[14.265]
[14.2681]
[14.2701]
[14.2713]
[14.2]
[14.2057]
[14.2147]
[14.226]
[14.2376]
[14.2484]
[14.2578]
[14.2655]
[14.2715]
[14.276]
[14.2791]
[14.2811]
[14.2823]
[14.2095]
[14.2151]
[14.2242]

```
[14.2354]
[14.247]
[14.2579]
[14.2672]
[14.2749]
[14.281]
[14.2855]
[14.2885]
[14.2906]
[14.2918]
[14.2164]
[14.2221]
[14.2311]
[14.2424]
[14.254]
[14.2648]
[14.2742]
[14.2819]
[14.2879]
[14.2924]
[14.2955]
[14.2975]
[14.2987]
Para o estado (0, 0): 0 transferências
Para o estado (0, 1): 0 transferências
Para o estado (0, 2): 0 transferências
Para o estado (0, 3): 0 transferências
Para o estado (0, 4): 0 transferências
Para o estado (0, 5): 0 transferências
Para o estado (0, 6): 0 transferências
Para o estado (0, 7): 0 transferências
Para o estado (0, 8): 0 transferências
Para o estado (0, 9): 0 transferências
Para o estado (0, 10): 0 transferências
Para o estado (0, 11): 0 transferências
Para o estado (0, 12): 0 transferências
Para o estado (1, 0): 0 transferências
Para o estado (1, 1): 0 transferências
Para o estado (1, 2): 0 transferências
Para o estado (1, 3): 0 transferências
```

```
Para o estado (1, 4): 0 transferências
Para o estado (1, 5): 0 transferências
Para o estado (1, 6): 0 transferências
Para o estado (1, 7): 0 transferências
Para o estado (1, 8): 0 transferências
Para o estado (1, 9): 0 transferências
Para o estado (1, 10): 0 transferências
Para o estado (1, 11): 0 transferências
Para o estado (1, 12): 0 transferências
Para o estado (2, 0): 0 transferências
Para o estado (2, 1): 0 transferências
Para o estado (2, 2): 0 transferências
Para o estado (2, 3): 0 transferências
Para o estado (2, 4): 0 transferências
Para o estado (2, 5): 0 transferências
Para o estado (2, 6): 0 transferências
Para o estado (2, 7): 0 transferências
Para o estado (2, 8): 0 transferências
Para o estado (2, 9): 0 transferências
Para o estado (2, 10): 0 transferências
Para o estado (2, 11): 0 transferências
Para o estado (2, 12): 0 transferências
Para o estado (3, 0): 0 transferências
Para o estado (3, 1): 0 transferências
Para o estado (3, 2): 0 transferências
Para o estado (3, 3): 0 transferências
Para o estado (3, 4): 0 transferências
Para o estado (3, 5): 0 transferências
Para o estado (3, 6): 0 transferências
Para o estado (3, 7): 0 transferências
Para o estado (3, 8): 0 transferências
Para o estado (3, 9): 0 transferências
Para o estado (3, 10): 0 transferências
Para o estado (3, 11): 0 transferências
Para o estado (3, 12): 0 transferências
Para o estado (4, 0): 0 transferências
Para o estado (4, 1): 0 transferências
Para o estado (4, 2): 0 transferências
Para o estado (4, 3): 0 transferências
Para o estado (4, 4): 0 transferências
```

```
Para o estado (4, 5): 0 transferências
Para o estado (4, 6): 0 transferências
Para o estado (4, 7): 0 transferências
Para o estado (4, 8): 0 transferências
Para o estado (4, 9): 0 transferências
Para o estado (4, 10): 0 transferências
Para o estado (4, 11): 0 transferências
Para o estado (4, 12): 0 transferências
Para o estado (5, 0): 0 transferências
Para o estado (5, 1): 0 transferências
Para o estado (5, 2): 0 transferências
Para o estado (5, 3): 0 transferências
Para o estado (5, 4): 0 transferências
Para o estado (5, 5): 0 transferências
Para o estado (5, 6): 0 transferências
Para o estado (5, 7): 0 transferências
Para o estado (5, 8): 0 transferências
Para o estado (5, 9): 0 transferências
Para o estado (5, 10): 0 transferências
Para o estado (5, 11): 0 transferências
Para o estado (5, 12): 0 transferências
Para o estado (6, 0): 0 transferências
Para o estado (6, 1): 0 transferências
Para o estado (6, 2): 0 transferências
Para o estado (6, 3): 0 transferências
Para o estado (6, 4): 0 transferências
Para o estado (6, 5): 0 transferências
Para o estado (6, 6): 0 transferências
Para o estado (6, 7): 0 transferências
Para o estado (6, 8): 0 transferências
Para o estado (6, 9): 0 transferências
Para o estado (6, 10): 0 transferências
Para o estado (6, 11): 0 transferências
Para o estado (6, 12): 0 transferências
Para o estado (7, 0): 0 transferências
Para o estado (7, 1): 0 transferências
Para o estado (7, 2): 0 transferências
Para o estado (7, 3): 0 transferências
Para o estado (7, 4): 0 transferências
Para o estado (7, 5): 0 transferências
```

```
Para o estado (7, 6): 0 transferências
Para o estado (7, 7): 0 transferências
Para o estado (7, 8): 0 transferências
Para o estado (7, 9): 0 transferências
Para o estado (7, 10): 0 transferências
Para o estado (7, 11): 0 transferências
Para o estado (7, 12): 0 transferências
Para o estado (8, 0): 0 transferências
Para o estado (8, 1): 0 transferências
Para o estado (8, 2): 0 transferências
Para o estado (8, 3): 0 transferências
Para o estado (8, 4): 0 transferências
Para o estado (8, 5): 0 transferências
Para o estado (8, 6): 0 transferências
Para o estado (8, 7): 0 transferências
Para o estado (8, 8): 0 transferências
Para o estado (8, 9): 0 transferências
Para o estado (8, 10): 0 transferências
Para o estado (8, 11): 0 transferências
Para o estado (8, 12): 0 transferências
Para o estado (9, 0): 0 transferências
Para o estado (9, 1): 0 transferências
Para o estado (9, 2): 0 transferências
Para o estado (9, 3): 0 transferências
Para o estado (9, 4): 0 transferências
Para o estado (9, 5): 0 transferências
Para o estado (9, 6): 0 transferências
Para o estado (9, 7): 0 transferências
Para o estado (9, 8): 0 transferências
Para o estado (9, 9): 0 transferências
Para o estado (9, 10): 0 transferências
Para o estado (9, 11): 0 transferências
Para o estado (9, 12): 0 transferências
Para o estado (10, 0): 0 transferências
Para o estado (10, 1): 0 transferências
Para o estado (10, 2): 0 transferências
Para o estado (10, 3): 0 transferências
Para o estado (10, 4): 0 transferências
Para o estado (10, 5): 0 transferências
Para o estado (10, 6): 0 transferências
```

```
Para o estado (10, 7): 0 transferências
Para o estado (10, 8): 0 transferências
Para o estado (10, 9): 0 transferências
Para o estado (10, 10): 0 transferências
Para o estado (10, 11): 0 transferências
Para o estado (10, 12): 0 transferências
Para o estado (11, 0): 0 transferências
Para o estado (11, 1): 0 transferências
Para o estado (11, 2): 0 transferências
Para o estado (11, 3): 0 transferências
Para o estado (11, 4): 0 transferências
Para o estado (11, 5): 0 transferências
Para o estado (11, 6): 0 transferências
Para o estado (11, 7): 0 transferências
Para o estado (11, 8): 0 transferências
Para o estado (11, 9): 0 transferências
Para o estado (11, 10): 0 transferências
Para o estado (11, 11): 0 transferências
Para o estado (11, 12): 0 transferências
Para o estado (12, 0): 0 transferências
Para o estado (12, 1): 0 transferências
Para o estado (12, 2): 0 transferências
Para o estado (12, 3): 0 transferências
Para o estado (12, 4): 0 transferências
Para o estado (12, 5): 0 transferências
Para o estado (12, 6): 0 transferências
Para o estado (12, 7): 0 transferências
Para o estado (12, 8): 0 transferências
Para o estado (12, 9): 0 transferências
Para o estado (12, 10): 0 transferências
Para o estado (12, 11): 0 transferências
Para o estado (12, 12): 0 transferências
```