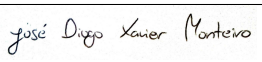
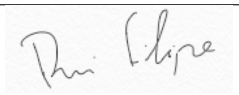
Universidade do Minho
2ºSemestre 2019/20
(MIEI, 3ºAno)

Modelos Estocásticos de Investigação Operacional

Trabalho Prático

Identificação do Grupo

Número:	Nome completo:	Rubrica:
84442	Diogo Pinto Ribeiro	
83638	José Diogo Xavier Monteiro	
83712	Rui Filipe Moreira Mendes	
84930	Rui Pedro Neto Reis	

Data de entrega: 2020-05-10

CONTEÚDO

1

INTRODUÇÃO & CONTEXTUALIZAÇÃO

No intuito do estudo de processos estocásticos, foi nos proposta a resolução de um problema em torno desta temática. Com o objectivo de ser um problema no intermédio entre simplicidade e complexidade do mundo real, é apresentado pela Unidade Curricular de Métodos Estocásticos de Investigação Operacional um enunciado com este mesmo propósito.

Desta forma, o enunciado proposto visa transmitir as capacidades essenciais em termos de formulação inicial de problemas desta tipicidade. Por essa razão, é fornecido um problema representativo de um problema de decisão Markoviano com um número infinito de estágios e diferentes decisões. Com o objetivo final de estabelecer, segundo um algoritmo de iteração de valor, um conjunto de decisões adequadas para cada filial. Posteriormente, sendo este mesmo resultados debatidos e as suas implicações enunciadas.

Este quebra-cabeças rege-se à volta de um grupo internacional de aluguer de automóveis, o grupo submetido a este estudo possui em sua posse 2 filiais distintas. Em cada uma das filiais podem, a qualquer dado momento, chegar um número variável de clientes, tanto para alugar como para entregar viaturas previamente alugada. Porém, considera-se que no máximo ambas as filiais só podem possuir 12 automóveis no final do dia, no máximo. Qualquer excesso deverá ser redistribuído para filiais externas à alçada do grupo. Naturalmente, quando uma filial possui muitas mais unidades que outra, pode decidir transferir unidades para a filial vizinha, sendo que no máximo apenas 3 podem ser transferidas. Este mecanismo de transferência perfaz portanto o conjunto de decisões que podem vir a ser tomadas. Nomeadamente, é possível identificar um conjunto de 7 decisões diferentes emergentes deste contexto.

Ademais, consideram-se também condições de contexto adicionais representativas de um problema real. Em especifico, considera- se durante a noite houverem mais do que 8 automóveis numa filial, então essa filial deverá ter de pagar 10 €, no total, para obter espaços adicionais. Também se considera-se que cada transferência acarreta um custo de

7 € por unidade transferida, e também verificamos que cada cada automóvel alugado custa, ao cliente, um valor de 30 € que entram positivamente nas contas do grupo.

Começamos por explicitar este problema em separado, focando em cada uma das filiais individualmente, por simplicidade, e de seguida passamos ao algoritmo utilizado para unir estas duas filiais num sistema composto. Seguido pela respectiva implementação do algoritmo de iteração de valor, e finalmente debatemos os resultados obtidos por este meio.

Por fim, realizamos um estudo comparativo sobre a temática do problemas de decisão Markovianos.

2

PARTE 1

Como indicamos anteriormente, este problema visa atacar a problemática de um grupo internacional, agregando um conjunto de 2 filiais. Necessariamente, estas 2 filiais perfazem, entre si, um sistema composto. Sendo que no algoritmo utilizado é necessário que ambas as matrizes, de contribuição e transição, contemplem ambas as filiais.

Por essa razão, e de forma a manter o problema a um nível tangível. Cada uma das filiais é enfrentada de forma individual, sendo posteriormente emparelhadas consoante as decisões tomadas.

2.1 FORMULAÇÃO INICIAL

Considera-se que o estado desta sistema corresponde a um estado composto por cada uma das filiais. Por isso, o estado pode ser identificado como um tuplo (x, y) em que x representa o número de automóveis da 1^{a} filial, e y o número de automóveis na 2^{a} filial.

Ademais, considera-se o estágio como sendo fixo à noite, diariamente. Esta decisão assenta na ideia de que faz sentido que o estado seja fixo no momento imediatamente anterior ao instante da decisão de transferência. Permitindo assim um mecanismo de controlo das decisões tomadas.

Porém, de forma a manter a simplicidade do mecanismo, decide-se separar este sistema nas suas partes atómicas. Olhando, primeiro, para cada uma das filiais, com os mesmos estágios, e de seguida unir ambas no estado composto.

É necessário considerar que a transição de uma dada filial pode estar associada a vários valores. Nomeadamente, deve-se considerar:

- ψ_i^k : Probabilidade de haver i pedidos na k-ésima filial.
- μ_i^k : Probabilidade de haver i entregas na k-ésima filial.

2.1.1 *Matriz de Transição*

Com isto em conta, é possível provar que a probabilidade associada à transição $i \to j$ na filial k é dada por:

$$p_k(i,j) = \sum_{N=max(0,i-j)}^{i} \psi_N^k \mu_{N-(i-j)} + \sum_{N=i}^{C} \psi_N^k \mu_j^k \tag{1}$$

Podemos verificar isto com um exemplo. Por exemplo, se tentarmos identificar, exaustivamente, os casos em que é possível a transição $1 \to 3$, obteremos o seguinte resultado:

$$[\psi_0\mu_2 + \psi_1\mu_3] + [\psi_2\mu_3 + \psi_3\mu_3 + \cdots + \psi_{12}\mu_3] \tag{2}$$

De onde é possível verificar que a primeira parte da equação é referente ao caso de pedidos totalmente satisfeitos, e a segunda referente a casos em que os pedidos são apenas parcialmente satisfeitos.

Porém, devido à limitação de capacidade em cada filial, limitada a C, há uma acumulação de probabilidades quando $j = C$. Por exemplo, se começarmos com 7 veículos e 12 forem entregues, apesar de nenhum ser requisitado. Então, o resultado final seria de 19. Logo, estas ficam fixas em C.

Pelo que $\phi^k(i)$ representa a probabilidade de haver um transbordo na filial k, partindo de um estado inicial com i veículos. Sendo definida da seguinte forma:

$$\phi_k(i) = \sum_{N=0}^{i-1} \sum_{w=1}^{i-k} \psi_N^k \mu_{C-i+w+k}^k \tag{3}$$

Então, considerando o transbordo de veículos, obtemos que probabilidade final da transição $i \to j$ é dada por:

$$\delta_k^*(i,j) = \begin{cases} p_k(i,j) & j < C \\ p_k(i,j) + \phi_k(i) & j = C \end{cases} \tag{4}$$

Onde é possível verificar que sempre que alcançamos o estado $j = C$, há uma inerente acumulação de probabilidades.

O que permite uma definição total da matriz de transição, sem considerar transferências de veículos. Assumindo E como sendo o conjunto das transições possíveis, então verifica-se que

$$P_0^k(i,j) = \delta_k^*(i,j) \tag{5}$$

2.1.2 *Matriz de Contribuição*

As contribuições seguem inevitavelmente a mesma lógica que as transições. Porém, tendo em conta que é impossível especificar ao certo qual será o contributo de uma determinada transição, visto haver um número grande de diferentes casos possíveis, é imperioso o inteiramento sobre a média aritmética ponderada, que será empreendido nesta formulação.

Desta forma, considera-se a contribuição de uma transição $i \to j$ como sendo a média aritmética ponderada relativa a cada um dos casos. Por isso, começamos por identificar os casos em que os pedidos são satisfeitos, total ou parcialmente, sendo esse contributo não ponderado, dado por $r^k(i,j)$, como se segue.

$$r^k(i,j) = 30 \cdot \left[\sum_{N=max(0,i-j)}^{i} N\psi_N^k \mu_{N-(i-j)} + \sum_{N=i}^{C} N\psi_N^k \mu_j^k \right] \tag{6}$$

Onde se verifica que, para cada pedido, é fornecido um lucro pesado de 30 €. Continuamos sujeitos à situação em que há uma sobrelotação, essa probabilidade é dada por $\phi_k(i)$, o contributo respectivo será então dado pela função $\theta_k(i)$.

$$\theta_k(i) = 30 \cdot \left[\sum_{N=0}^{i-1} \sum_{w=1}^{i-k} N\psi_N^k \mu_{C-i+w+k}^k \right] \tag{7}$$

De forma a contabilizar estas contribuições individuais, usamos a função $\xi_k(i,j)$.

$$\xi_k(i,j) = \begin{cases} r_k(i,j) & j < C \\ r_k(i,j) + \theta_k(i) & j = C \end{cases} \tag{8}$$

Porém, este resultado corresponde à média não-pesada, por isso se quisermos contabilizar o seu peso, bem como o custo de estacionamento adicional, ficamos com a

seguinte situação, na qual é possivel definir totalmente a matriz de contribuição, para 0 transferências.

$$R_0^k(i,j) = \begin{cases} \dfrac{r_k(i,j)}{P_k(i,j)} & i \leq 8 \\ \dfrac{r_k(i,j)}{P_k(i,j)} - 10 & i > 8 \end{cases} \tag{9}$$

2.1.3 *Decisões Disponíveis*

Para introduzir complexidade, sabe-se que estamos constantemente perante 7 decisões. Nomeadamente, a decisão de não transferir nenhuma unidade, e transferir até um máximo de 3 unidades entre filiais.

Estas transferências ocorrem no período noturno, pelo que quando ocorre uma transferências, na manha seguinte, o numero de unidades já terá sido afetado por esta mesma transferência. Por exemplo, se uma das filiais possuir 1 unidade no inicio do estágio, e receber 2 unidades, então na manhã seguinte ira possuir 3 unidades. Isto faz com que aconteça um *shift* ao nível de probabilidades e contribuições. Pois, notoriamente, se de noite recebermos unidades extras, a probabilidade de acabarmos no estado j vai ser, inevitavelmente alterada.

Olhando primeiro para o caso de quem envia unidades, verifica-se que deverá ser impossível transferir N unidades caso essa filial passa-se de um estado i tal que $i < N$. Caso seja impossível, mantemos o próprio estado, apesar de na prática isto não afectar o algoritmo de iteração de valor. Isto porque, de forma a evitar que esta decisão impossível alguma vez seja seleccionada, atribuímos-lhe uma contribuição M, suficientemente pequena tal que a decisão nunca seja selecionada. Nesta caso, usamos $M = -1000$.

Então, quando uma dada filial envia unidades, as suas probabilidades de transição e contribuição ficam definidas da seguinte forma. Sendo N o número de unidades transferidas.

$$P_N^k(i,j) = \begin{cases} 0 & i < N \wedge i \neq j \\ 1 & i < N \wedge i = j \\ P_0(i-N,j) & i \geq N \end{cases} \tag{10}$$

$$R_N^k(i,j) = \begin{cases} M & i < N \\ R_0(i-N,j) - 7N & i \geq N \end{cases} \tag{11}$$

Analogamente, quando durante a noite recebemos viaturas, temos a garantia que na noite seguinte possuiremos pelo menos essa quantidade de viaturas. De igual forma, será impossível realizar a transferências de N viaturas caso se parta de um estado $i > C - N$. Isto porque, implicaria uma ultrapassagem da capacidade máxima das filiais. Então, resulta a seguinte alteração das respectivas matrizes.

$$P_N^k(i,j) = \begin{cases} P_0(i+N,j) & i \leq C-N \\ 0 & i > C-N \wedge i \neq j \\ 1 & i > C-N \wedge i = j \end{cases} \tag{12}$$

$$R_N^k(i,j) = \begin{cases} M & i < N \\ R_0(i-N,j) & i \geq N \end{cases} \tag{13}$$

2.2 Conversão para sistema agregado

Até agora, todo o comportamento, respectivo a cada filial, tem vindo a ser desenvolvido em separado. Olhando para a implicação das transferências inter-filiais de forma separada, focando numa situação *per* filial.

Porém, no contexto do estudo desenvolvido é necessário expandir o mecanismo desenvolvido até agora de forma a encapsular, também, a união de ambas as filiais. Obtendo-se assim um sistema cujo estado é a união dos estado de cada uma das filiais.

Desta forma, no sistema todas as transições possuem o seguinte formato. Assumindo i_k como sendo o número de veículos inicial na k-ésima filial, f_k como sendo o número de veículos na k-ésima filial na manhã seguinte, e L como sendo número máximo de filiais.

$$X = (i_0, i_1, \cdots, i_{L-1}) \longrightarrow Y = (f_0, f_1, \cdots, f_{L-1}) \tag{14}$$

Até agora desenvolvemos mecanismos necessários a cada um das transições individualmente, por exemplo, descobrir a probabilidade associada à transição $i_k \longrightarrow f_k$. Porém, é a união destas peças que completa o sistema. Assumindo a existência das seguintes funções $P_k^N(i,j)$ e $R_k^N(i,j)$, referentes às matrizes de transição e contribuição, respectivamente, da N-ésima filial assumindo que foi tomada a k-ésima decisão.

Relativamente à matriz de contribuição do estado composto, é fácil prever que a contribuição da transição $X \longrightarrow Y$ (originários da equação 14) é a média aritmética

ponderada das contribuições de cada uma das filiais individualmente. Assumindo um $R^*(X,Y)$ como sendo a contribuição da transição do sistema composto, então temos que:

$$R^*(X,Y) = \frac{\sum_{N=0}^{L-1}\left[R_k^N(i_N,f_N)\cdot P_k^N(i_N,f_N)\right]}{\prod_{N=0}^{L-1}P_k^N(i_N,f_N)} \tag{15}$$

A probabilidade associada à transição $X \longrightarrow Y$ corresponde à composição de probabilidades individuais de cada uma das filiais. Neste caso, associada à transição da equação 14, teríamos a probabilidade $P(X \to Y)$ que resulta em:

$$P(X \to Y) = P(i_0 \to f_0) \times P(i_1 \to f_1) \times \cdots \times P(i_{L-1} \to f_{L-1}) \tag{16}$$

Logo, podemos simplesmente escrever a probabilidade desta transição, no sistema global, como sendo descrita por:

$$P^*(X,Y) = \prod_{N=0}^{L-1} P_k^N(i_N,f_N) \tag{17}$$

Continua a ser necessário garantir que a soma das probabilidades elementares é 1, o que corresponde a garantir que a soma das probabilidades de cada linha da matriz de transição deve ser 1. Assumindo C como sendo a máxima capacidade das filiais, então temos a garantia, estudada acima, de que:

$$\sum_{j=0}^{C} P_k^N(i,j) = 1, \forall(i,N) \tag{18}$$

No contexto do sistema global, continuamos a ter necessidade deste axioma. Pretende-se provar que a soma destas linhas continua a ser 1. Analisando o caso de $L = 2$ filiais, apesar de ser possível generalizar para qualquer número de filiais, verificamos que:

$$\begin{aligned}\sum_{f_1=0}^{C}\sum_{f_0=0}^{C} P^*(X,(f_0,f_1)) &= \sum_{f_1=0}^{C}\sum_{f_0=0}^{C}[P_k^0(i_0,f_0)\cdot P_k^1(i_1,f_1)]\\ &= \sum_{f_1=0}^{C}\left[P_k^1(i_1,f_1)\cdot\sum_{f_0=0}^{C}P_k^0(i_0,f_0)\right]\\ &= \sum_{f_1=0}^{C}P_k^1(i_1,f_1)\\ &= 1\end{aligned}$$

Pelo que se prova que é possível, a partir da composição de diferentes filiais num só sistema, manter todas as condições e requisitos de aplicação. Permitindo assim, a transição para um sistema completo, sobre o qual é possível aplicar algoritmos, nomeadamente algoritmos destinados a sistema com estágios infinitos e com decisão, como o que se apresenta no presente estudo.

2.3 Análise de Resultados

Tendo em conta todo o conteúdo desenvolvido acima, os dados fornecidos pelo docente e a nossa implementação, é nos possível obter o seguinte resultado.

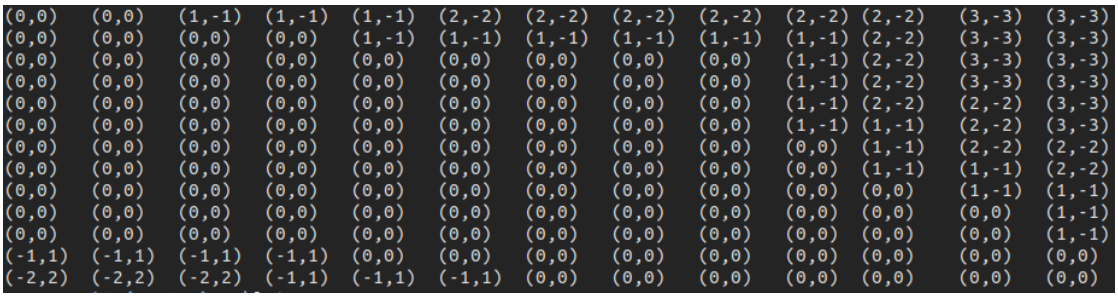


Figura 1: Resultado obtido com a implementação.

Nesta imagem, as linhas representam o número de unidades na primeira filial, e o número da coluna representa o número de unidades da segunda filial. Por exemplo, na 1ª linha e 13ª coluna, verificamos o valor $(3,-3)$. Isto quer dizer que no estado em que possuímos 0 veículos na 1ª filial e 12 veículos na 2ª filial, deveremos tomar a decisão de transferir 3 unidades da filial 2 para 1.

Como se pode verificar os resultados obtidos são promissores, visto que o estado global do sistema, tenta sempre encontrar um equilíbrio, consoante as probabilidades fornecidas. O que é exactamente o que se pretende.

Para testar esta tese, e possuir algum termo de comparação, iremos tentar emular este mesmo mecanismo, mas agora assumindo que as probabilidade possuem todas o mesmo valor. Ou seja, qualquer probabilidade de pedidos ou entrega deve ser igual a $\frac{1}{13} \approx 0.0769$. Intuitivamente, deve-se verificar que a matriz gerada corresponde a uma matriz completamente simétrica. Pois há uma igual e constante distribuição de probabilidades ao longo do sistema.

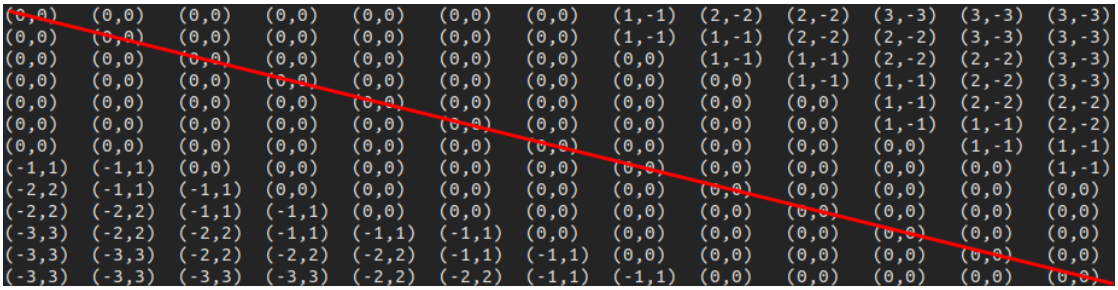


Figura 2: Resultado obtido com iguais probabilidades.

Como se verifica, a matriz é espalhada segundo a linha vermelha. O que se apresenta como uma prova de suporte à nossa tese inicial.

Concluímos, desta forma, que a implementação originada é suficientemente descritiva, captando as componentes necessárias a este funcionamento.

3

PARTE 2

Após efectuarmos uma pesquisa e um debate entre os elementos do nosso grupo, escolhemos o seguinte artigo para explorar nesta parte do trabalho prático - 'Traffic congestion prediction based on Hidden Markov Models and contrast measure' [1].

O principal foco neste artigo é a previsão do tráfego automóvel num futuro próximo. Na sociedade atual, a necessidade de deslocações quer por parte de indivíduos, quer por parte de empresas e mercadorias é muito elevada, o que por sua vez cria grandes concentrações de tráfego automóvel em determinadas zonas e alturas do dia. Estas elevadas concentrações de tráfego automóvel levantam certas questões como quais os recursos a alocar, quais os limites de velocidade a escolher, que desvios podem ser feitos para minimizar o congestionamento, bem como muitas outras.

O modelo proposto neste artigo assenta principalmente em Modelos de Markov Ocultos (*Hidden Markov Models* - HMM), sendo que é acrescentado a este modelo a noção de Contraste (*Contrast*). O Modelo de Markov Oculto é utilizado para fazer a previsão dos estados do tráfego automóvel em horas de "ponta", sendo que o Contraste serve para podermos distinguir diferentes estados uns dos outros. Assim sendo, o modelo proposto neste artigo é um modelo *HMM-Contrast* (HMM-C). Este modelo resulta da fusão dos dois conceitos anteriormente vistos.

Para a previsão do tráfego automóvel é utilizado o HMM pois este é aquele que tem uma estrutura que representa melhor o comportamento do tráfego automóvel. O tráfego automóvel é algo extremamente complexo, pois as variáveis que condicionam o tráfego são desconhecidas. Isto é, variáveis como a velocidade por exemplo, servem apenas como a medição de um ponto e não são representativas do estado do tráfego. É esta situação que faz com que os estados neste modelo sejam considerados ocultos. É devido à natureza oculta dos estados que estes necessitam de ser definidos ao longo de intervalos de tempo recorrendo a estatísticas. Estas estatísticas são definidas ao longo de uma janela de tempo composta por um determinado número de observações. Algumas destas estatísticas como

médias e desvio padrão são consideradas bem definidas. No entanto, o contraste não o é. O contraste é uma medida estatística de segunda ordem que nos permite efetuar a diferenciação entre diversos estados. O contraste permite-nos saber em que nível ocorrem as variações locais nos estados, sendo que por sua vez nos indicam o nível de perturbação numa sequência de observações. Um valor de contraste baixo permite-nos caracterizar situações em que não existem variações no fluxo do tráfego, ou baixos níveis de tráfego por exemplo. Um valor de contraste alto, indica-nos exatamente o contrário, nomeadamente alterações repentinas do tráfego devido a uma baixa capacidade de fluxo da via, um número muito elevado de viaturas, ou mesmo fatores como conduções mais agressivas.

A principal função do modelo proposto neste artigo é a prestação de apoio a entidades responsáveis pela gestão de vias rodoviárias. O modelo permite que sejam efetuadas previsões do estado do tráfego numa via para o qual tenham sido fornecidos determinados conjuntos de dados, sendo que no final, estas previsões podem ser efetuadas para intervalos que se estendem desde alguns minutos até 90 minutos. Decidimos mencionar os dados de input do problema devido à sua extrema importância. Dada a natureza estocástica do modelo, este depende muito da qualidade dos dados fornecidos. Um conjunto de dados relevante para este modelo passa por escolher um determinado traçado entre 2 cruzamentos, e, dividir um dia em intervalos, por exemplo, de 10 minutos e identificá-los. Para cada um destes intervalos é-nos relevante ter dados como o comprimento do traçado, o tempo de viagem médio, a velocidade, e uma média do fluxo. Com um conjunto de dados seguindo esta estrutura seríamos capazes de iniciar o modelo.

Para entendermos melhor o funcionamento e utilidade deste modelo convém abordarmos os resultados que são produzidos por este, bem como o seu respetivo formato. O estados que descrevem o tráfego em horas que ocorrem picos são bi-dimensionais (2D). Estes estados bi-dimensionais são compostos por velocidade e contraste. Deste modo, a magnitude do contraste, quer seja positiva ou negativa é influenciada pelas alterações nas condições do tráfego. Sempre que existe um decréscimo na velocidade, o contraste assume valores negativos, sendo que podemos concluir que existe uma alteração negativa nas condições do tráfego. No sentido oposto, um aumento na velocidade terá associado um valor de contraste positivo, o que por sua vez nos indica uma melhoria nas condições do tráfego. Através do modelo proposto neste artigo é possível melhorar substancialmente a eficácia da gestão do tráfego automóvel bem como de recursos necessários para o gerir. Ao prever o tráfego em determinadas zona ou alturas do dia, é possível antecipadamente efetuar determinadas melhorias, isto é, resolver questões concretas do nosso quotidiano. Algumas das melhorias que podem ser feitas passam pelo ajuste de velocidades em

determinadas zonas, alteração do número de reboques disponível numa determinada zona e até à utilização de polícias de trânsito.

REFERÊNCIAS

[1] John F.Zaki et al. “Traffic congestion prediction based on Hidden Markov Models and contrast measure”. Em: *Ain Shams Engineering Journal* (2019). DOI: https://doi.org/10.1016/j.asej.2019.10.006.

A

DADOS FORNECIDOS

```
Grupo que inclui o Aluno com o Nº 84930
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):  ; 0.0356 ; 0.0904 ; 0.1380 ; 0.1400 ; 0.1224 ;
    0.1292 ; 0.0952 ; 0.0820 ; 0.0560 ; 0.0496 ; 0.0324 ; 0.0216 ; 0.0076
Probabilidade (entregas): ; 0.0448 ; 0.1632 ; 0.2220 ; 0.2092 ; 0.1620 ;
    0.1056 ; 0.0556 ; 0.0236 ; 0.0100 ; 0.0036 ; 0.0000 ; 0.0000 ; 0.0004

FILIAL 2
Número de clientes:   ;      0 ;      1 ;      2 ;      3 ;      4 ;
        5 ;      6 ;      7 ;      8 ;      9 ;     10 ;     11 ;     12
Probabilidade (pedidos):  ; 0.0612 ; 0.1204 ; 0.1476 ; 0.1228 ; 0.1080 ;
    0.1100 ; 0.0788 ; 0.0776 ; 0.0576 ; 0.0516 ; 0.0328 ; 0.0236 ; 0.0080
Probabilidade (entregas): ; 0.0192 ; 0.0848 ; 0.1540 ; 0.1956 ; 0.2040 ;
    0.1528 ; 0.0884 ; 0.0556 ; 0.0284 ; 0.0100 ; 0.0040 ; 0.0024 ; 0.0008
```

B

MATRIZES DE TRANSIÇÃO

C

CÓDIGO CRIADO

```
import numpy as np

"""
Calculates the values of (i,j), without overflow.
Returns (X,Y). Where X represents the probability
of (i,j). Y represents resulting un-weighted profit.
"""
def prob(i, j, C, requests, deliveries):
  X = 0
  Y = 0

  for p in range( max(0, i-j), i ):
    prob = requests[p] * deliveries[p - (i - j)]
    X += prob
    Y += prob * p

  for p in range( i, C + 1 ):
    prob = requests[p] * deliveries[j]
    X += prob
    Y += prob * i

  return (X,Y*30)

"""
Calculates the vales of (i,j), considering overflow.
Returns (X,Y). Where X represents the probability
of (i,j). Y represents resulting un-weighted profit.
"""
def phi(i, C, requests, deliveries):
  X = 0
  Y = 0
```

```
  for k in range(0, i):
    for w in range(1, i - k + 1):
      prob = requests[k] * deliveries[C - i + w + k]
      X += prob
      Y += prob * k

  return (X,Y*30)

"""
Builds both the transation matrix and contribution
matrix.
Returns (tMat,cMat). Where tMat represente the transition
matrix and cMat represents the contribution matrix.

nTrans indicates the number of transfered vehicles to that
branch. If nTrans > 0, then the branch is the reciever of
vehicles. If nTrans = 0, the branch does not provide nor
recieve any vehicles. If nTrans < 0, then the branch is
the sender of the vehicles.
"""
def buildMatrixes(C, nTrans, requests, deliveries):
  X = np.zeros((C+1,C+1))
  Y = np.zeros((C+1,C+1))

  for i in range(0, C + 1):

    if abs(min(nTrans,0)) <= i and i <= C - nTrans:

      for j in range( 0, C + 1):
        (a,b) = prob(i+nTrans, j, C, requests, deliveries)
        X[i,j] += a
        Y[i,j] += b

      (a,b) = phi(i+nTrans, C, requests, deliveries)
      X[i,C] += a
      Y[i,C] += b
    else:
      X[i,i] = 1


  for i in range(0, C + 1):
    for j in range(0, C + 1):
      if X[i,j] == 0:
        Y[i,j] = 0
```

```
      else:
        Y[i,j] = Y[i,j] / X[i,j]


  # Price per transfer
  if nTrans < 0:
    Y = np.add(Y, 7 * nTrans)


  if nTrans < 0:
    for i in range(0, abs(nTrans)):
      for j in range(0, C + 1):
        Y[i,j] = -1000
  elif nTrans > 0:
    for i in range(C - nTrans + 1, C + 1):
      for j in range(0, C + 1):
        Y[i,j] = -1000

  return (X,Y)

"""
Binds separate transition and contributions matrixes,
of both matrixes together, aggregating them together.
Returns (X,Y). Where X is the binding of the transitions
and Y is the binding of the contributions.
"""
def bindMatrixes(C, (a1,b1), (a2,b2)):
  k = C + 1
  N = k * k
  X = np.zeros( (N, N) )
  Y = np.zeros( (N, N) )

  for i in range(0, N):

    i0 = i // k
    i1 = i % k

    for j in range(0, N):

      f0 = j // k
      f1 = j % k

      X[i,j] = a1[i0,f0] * a2[i1,f1]
```

```
      if b1[i0,f0] < -700 or b2[i1,f1] < -700:
        Y[i,j] = -1000
      else:
        Y[i,j] = b1[i0,f0] * a1[i0,f0] + b2[i1,f1] * a2[i1,f1]

        if X[i,j] != 0:
          Y[i,j] = Y[i,j] / X[i,j]

  return (X,Y)

# Generates all possibilites
def entryProblem(requests, deliveries):
  tMats = []
  cMats = []
  dict = {}
  k = 1

  F1 = buildMatrixes(12,0,requests[0],deliveries[0])
  F2 = buildMatrixes(12,0,requests[1],deliveries[1])
  (a,b) = bindMatrixes(12, F1, F2)
  tMats.append(a)
  cMats.append(b)
  dict[0] = '(0,0)'

  for i in range(-3, 4):
    if i != 0:
      F1 = buildMatrixes(12,i,requests[0],deliveries[0])
      F2 = (a1,b1) = buildMatrixes(12,-i,requests[1],deliveries[1])
      (a,b) = bindMatrixes(12, F1, F2)
      tMats.append(a)
      cMats.append(b)
      dict[k] = '(' + str(i) + "," + str(-i) + ')'
      k = k + 1

  return (tMats, cMats, dict)

# Estimates, for each decision, the expected contribution.
def estContributions(k, w, tMat, cMat):
  return [np.array(np.sum(np.multiply(tMat[i],cMat[i]), axis=1)).reshape
     ((w,1)) for i in range(k)]

# Estimates, for each decisions, the total expected contribution.
def estTotalContribs(k, cts, tMat, optPolicy):
  return [np.add(cts[i], np.matmul(tMat[i],optPolicy)) for i in range(k)]
```

```
# Applies the value iteration algorithm.
def valueIteration(nDecisions, nStates, transMat, contribMat, iterMax =
    20):
  n = 0
  # Define def. margin.
  epsilon = 0.01

  # Establish F0.
  Fn = F = np.zeros((nStates,1))

  # Decision taken in each state.
  calls = np.zeros(nStates)

  # Establish expected contribution for each decision.
  contribs = estContributions(nDecisions, nStates, transMat, contribMat)

  while n < iterMax:

    # Calculates total expected contribution.
    Vn = estTotalContribs(nDecisions, contribs, transMat, Fn)

    # Intermediate matrix for different decisions.
    tmp = np.concatenate(Vn, axis=1)

    # Specify the chosen policies.
    calls = np.argmax(tmp, axis=1)

    Fn = np.array(np.max(tmp, axis=1)).reshape((nStates,1))
    Dn = Fn.T - F.T
    F = Fn

    # Leave if Verication Condition checks out.
    if np.max(Dn) - np.min(Dn) < epsilon:
      break

    n = n + 1

  return calls

# Dados Relativos a filial 1.
requests1 =  [0.0356, 0.0904, 0.1380, 0.1400, 0.1224, 0.1292, 0.0952,
    0.0820, 0.0560, 0.0496, 0.0324, 0.0216,  0.0076]
```

```
deliveries1 = [0.0448, 0.1632, 0.2220, 0.2092, 0.1620, 0.1056, 0.0556,
    0.0236, 0.0100, 0.0036, 0.0000, 0.0000, 0.0004]


# Dados Relativos a filial 2.
requests2  = [0.0612, 0.1204, 0.1476, 0.1228, 0.1080, 0.1100, 0.0788,
    0.0776, 0.0576, 0.0516, 0.0328, 0.0236, 0.0080]
deliveries2 = [0.0192, 0.0848, 0.1540, 0.1956, 0.2040, 0.1528, 0.0884,
    0.0556, 0.0284, 0.0100, 0.0040, 0.0024, 0.0008]


(tMat,cMat,dict) = entryProblem([requests1,requests2],[deliveries1,
    deliveries2])

k = 169

calls = valueIteration(7, 169, tMat, cMat)

for i in range(0, 13):
  for j in range(0,12):
    print(dict.get(calls[i*13+j]) + '\t'),
  print(dict.get(calls[i*13+12]))
```

D

FUNCIONAMENTO DO PROGRAMA

D.1 INPUTS

```
# Dados Relativos a filial 1.
requests1 =  [0.0356, 0.0904, 0.1380, 0.1400, 0.1224, 0.1292, 0.0952,
    0.0820, 0.0560, 0.0496, 0.0324, 0.0216,  0.0076]
deliveries1 = [0.0448, 0.1632, 0.2220, 0.2092, 0.1620, 0.1056, 0.0556,
    0.0236, 0.0100, 0.0036, 0.0000, 0.0000, 0.0004]


# Dados Relativos a filial 2.
requests2  = [0.0612, 0.1204, 0.1476, 0.1228, 0.1080, 0.1100, 0.0788,
    0.0776, 0.0576, 0.0516, 0.0328, 0.0236, 0.0080]
deliveries2 = [0.0192, 0.0848, 0.1540, 0.1956, 0.2040, 0.1528, 0.0884,
    0.0556, 0.0284, 0.0100, 0.0040, 0.0024, 0.0008]
```

D.2 OUTPUTS

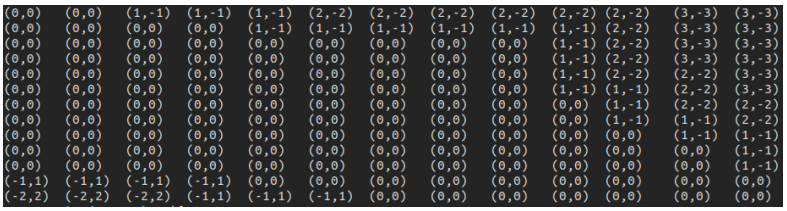


Figura 3: Output obtido.