Universidade do Minho

2ºSemestre 2019/20

(MIEI, 3ºAno)

Modelos Estocásticos de Investigação Operacional

Trabalho Prático

Identificação do Grupo 39

Número:	*Nome completo:*	*Rubrica:*
A86617	Gonçalo Pinto Nogueira	
A82529	Carlos Manuel Marques Afonso	
A83581	João Miguel Fernandes Cerqueira	

Data de entrega: 2020- 05 - 11

Parte 1

Depois de uma análise cuidada ao problema descrito no enunciado, constatamos que, de facto, trata-se de um problema relacionado com programação dinâmica estocástica de um número de estágios infinitos com alternativas.
Depois de observado o tipo de problema a resolver tivemos de optar por um método para resolver o problema em questão e começamos por elaborar as seguintes considerações iniciais:

Estados: Número de carros presente em cada filial ao fim do dia. Por exemplo, o estado 3-2 representa que ao fim do dia estão 3 carros presentes na filial 1 e 2 carros na filial 2.

Estágios: Dias. Ou seja, as mudanças de estados verificam-se de 24 em 24 horas.

Capacidade: Conforme apresentado no enunciado cada filial pode conter no máximo 12 carros.

Probabilidades: Probabilidades de pedidos e entregues fornecidas. (Anexo A1)

Decisões: Existem 7 ações alternativas neste problema e foram definidas das seguintes formas.

k=1 – Não se realizam transferências entre filiais.
k=2 – Transferir 1 carro da filial 1 para a filial 2.
k=3 – Transferir 2 carros da filial 1 para a filial 2.
k=4 – Transferir 3 carros da filial 1 para a filial 2.
k=5 – Transferir 1 carro da filial 2 para a filial 1.
k=6 – Transferir 2 carros da filial 2 para a filial 1.
k=7 – Transferir 3 carros da filial 2 para a filial 1.

Após definidas todas estas considerações iniciais o próximo passo a ser dado foi a construção das matrizes de transição (probabilidades e contribuições) para cada filial individualmente, para isso desenvolvemos a função presente no Anexo A3-Figura 6, que percorre todas as combinações possíveis de entregas e pedidos e vai preenchendo cada linha individualmente dependendo do estado final que é determinado seguindo a seguinte forma:

Estado Final=Estado Inicial – (min(Estado Inicial, Pedidos)) + Entregas;
Estado Final=min(12,Estado Final)

A última parte serve para garantir que o estado final é sempre 12, pois se der mais que 12 é referido no enunciado que os carros vão para outras filiais do grupo, mas que não geridas pelo empresário e por isso não contam para a resolução do problema.
Relativamente á parte das contribuições apenas multiplicamos o valor obtido em (min(Estado Inicial, Pedidos)) por 30 que corresponde ao ganho pelo aluguer de um carro e multiplicamos (max(0, Estado Inicial-8)) por -10 que representa o preço a pagar por espaço extra para armazenamento de carros quando a filial tem mais que 8 carros armazenados durante a noite e multiplicamos este valor dado pela sua probabilidade de acontecer. Por fim foi necessário dividir estes valores obtidos pelos valores na mesma linha e coluna da matriz P de forma a normalizar os valores acumulados para uma probabilidade total 1.

Após obtidas estas matrizes individuais por filiais foi preciso encontrar uma forma de transformá-las numa matriz 169x169 de forma a representar os estados definidos inicialmente .
Para isso foi necessário criar uma função que traduzisse o número de linha ou coluna para um estado do problema, por exemplo transformar a linha 14 no estado 1-0 (Anexo A3- Figura 7). Depois para obter os valores para preencher a matriz fizemos uso da seguinte expressão: $P[(a\rightarrow b)(c\rightarrow d)]= p1(a\rightarrow c)*p2(b\rightarrow d)$, e utilizamos a mesma expressão para obter a matriz R usando as matrizes r1 e r2 individuais previamente definidas (Anexo A3- Figura 8).

Até ao momento todos cálculos foram efetuados para o caso de a decisão ser não transferir nenhum carro entre filiais, no entanto é necessário calcular as matrizes P e R das outras 6 decisões.
Para o cálculo dessas matrizes a forma é bastante parecida com a de não transferir, é necessário apenas somar e subtrair nos valores de carros presentes do estado inicial de cada filial o numero de transferências da decisão a considerar sendo feito da seguinte forma: P_k2[(a→b)(c→d)]= p1(a-1→c)*p2(b+1→d) sendo este exemplo para a decisão k=2 transferir 1 carro da filial 1 para a filial 2(Anexo A3 Figura 8). No entanto, é preciso considerar alguns casos específicos em que a transferência de carros de uma filial para a outra é impossível, tendo em conta o exemplo anterior se por exemplo a=0, seria impossível transferir um carro da filial 1 para a filial 2, quando estes casos acontecem definimos a probabilidade como 9999 e a contribuição como -1 de forma a que no calculo da matriz Q este valor nunca seja selecionado para o Fn.
De salientar que como cada transferência tem um custo de 7 euros para o empresário foi necessário introduzir esse custo acrescido na matriz R de cada decisão dependendo de quantos carros seriam transferidos.

Depois de todas as matrizes P e R de cada decisão estarem preenchidas, foi necessário obter as matrizes Q e para isso criamos a função encontrada em Anexo A3 – Figura 11 para calcular essa matriz através da multiplicação das matrizes P e R.

Finalmente após obtidas as 7 diferentes matrizes Q correspondentes às 7 decisões, tínhamos já tudo o que era preciso para implementarmos o algoritmo para a obtenção do máximo ganho esperado.

Foi preciso então implementar o algoritmo que recebendo as 7 matrizes Q após um conjunto de operações e de um certo número de iterações obteria um conjunto de valores muito próximos que limitariam o valor de ganho máximo (Anexo A3 Figura 12).
Na iteração 0 os valores de Fn são todos 0 e não acontece mais nada, de seguida na próxima iteração multiplica-se a respetiva matriz de probabilidade pela matriz Fn-1 e soma-se á respetiva matriz Q, fazendo isto para as 7 diferentes matrizes. Depois de obtido este resultado, percorremos cada valor de todas as matrizes Qn e comparamos linha a linha, selecionamos o maior valor e colocamo-lo na matriz Fn e fazemos isso até percorrer todas as linhas. Após isso o cálculo de Dn é feito através da diferença entre Fn e Fn-1.
Na nossa implementação do algoritmo optamos para realizar 1000 iterações em vez de continuar sempre a iterar até chegarmos a valores exatamente iguais, devido a presença de muitas casas decimais nos valores utilizados conseguindo chegar a diferenças nos valores apenas em poucas décimas.

Após chamado o programa os valores obtidos em Dn foram os que constam no Anexo A4 e podemos então concluir e interpretar os resultados da seguinte forma:

Podemos concluir que o ganho é limitado por 71,0<=gn=1000~~g*<=71,2 e após verificar de quais decisões vêm os valores presentes no Dn na última iteração, o nosso programa deu como resposta que eram todos da decisão 0, não transferir nenhum carro entre filial sendo essa a política ótima para todos os estados iniciais possíveis.

No entanto tendo em conta o contexto do problema, concluímos que talvez nos possamos ter enganado em alguma parte da resolução, na nossa opinião, sem efetuar qualquer cálculo parece-nos que por exemplo se o estado fosse 3-3 a melhor decisão seria não transferir nenhum carro mas se o estado fosse 12-6 seria melhor transferir um carro ao mais da filial 1 para a filial 2.

Esboço parcial da rede de programação dinâmica com apenas alguns estados aleatórios onde cada transição de estado tem o seu custo e probabilidade associada, neste caso esses valores não estão representados no esboço, mas são os valores presentes nas matrizes 169x169 P e R que determinamos no programa desenvolvido.
O custo será R(estado inicial-> estado final) e probabilidade da mesma forma P(estado inicial-> estado final).

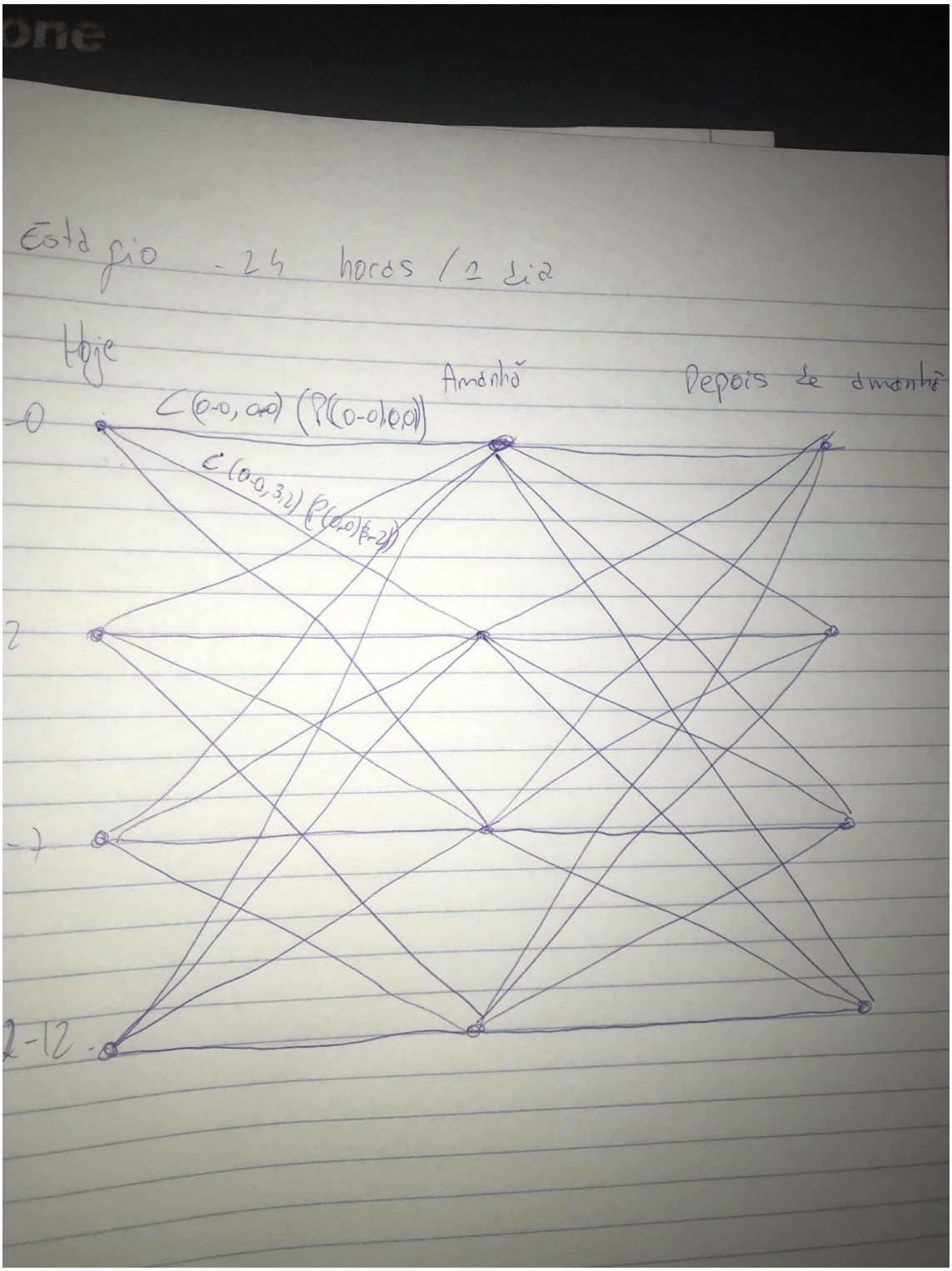


Figura 1Esboço parcial da rede de programação dinâmica

Parte 2

O artigo que escolhemos foi o intitulado "A Driving Intention Prediction Method Based on Hidden Markov Model for Autonomous Driving" por Shiwen Liu, Kan Zheng, Long Zhao e Pingzhi Fan.

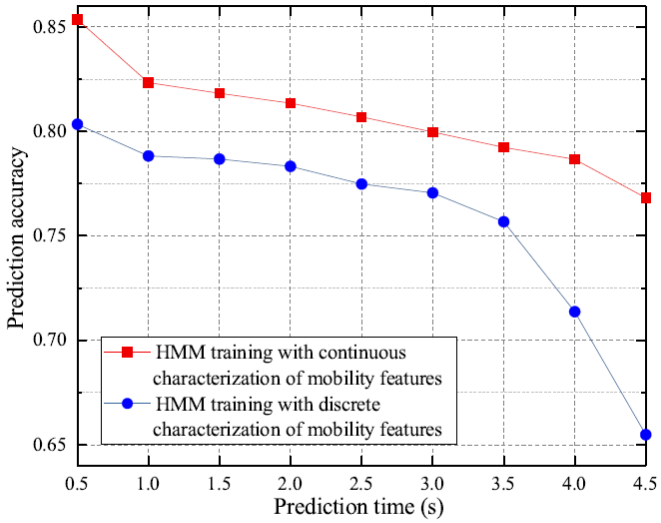
O estudo baseia-se em usar hidden markov models para ajudar a compreender intenções e decisões tomadas por condutores humanos de veículos na estrada de modo a tornar estradas mais seguras.
Com o gradual aparecimento de carros que se guiam autonomamente, como este é um processo que demora muito tempo a tornar-se global, é preciso preocuparmo-nos com a presença de veículos usados por humanos a conviverem com estes novos carros.
Enquanto que carros autônomos conseguem comunicar entre si e sabem o comportamento e as regras que todos partilham, sendo mais fácil prevenir acidentes ou desastres nas estradas, adicionar um elemento humano torna tudo muito mais complicado.
Para manter a segurança nas estradas é preciso com que os carros consigam reagir corretamente às decisões e aos erros humanos dos outros veículos.
Isto é difícil porque seres humanos são imprevisíveis, não existem regras certas que guiam comportamentos humanos.
Para tal, é decidido usar Hidden Markov Models, pois estes são bons para caracterizar a relação escondida entre observações e os estados escondidos que as criam.

Um modelo de markov escondido é um modelo de markov X em que os estados deste são escondidos, desconhecidos. Nestes modelos também existe outro processo de modelo de markov Y que depende do comportamento do original.
O objetivo destes modelos é de aprender sobre o primeiro processo de markov X a partir do segundo, Y.

Na experiência dividam o uso de Hidden Markov Models em dois. Usando caracterização discreta dos dados de mobilização dos veículos e com caracterização contínua destes dados.

Em geral, a experiência baseia-se em, a partir de vários dados obtidos de múltiplos veículos, conseguir adivinhar os próximos passos que um veículo vai fazer de seguida.
Por isso os investigadores usaram dados de uma estrada verdadeira, onde têm dados de um carro e os à sua volta. Os dados escolhidos foram as direções dos veículos, a velocidade e a distância às faixas da estrada onde estão. É guardada uma lista destes a cada intervalo de tempo.

No caso estudado pelo artigo, são usados dados gravadas a cada 0,5 segundos, 9 vezes onde o carro muda de faixa nos últimos dados gravados. Aos 4,5 segundos.
O estudo foi efetuado como data de 3910 veículos que passaram sobre o JianGuoMen Flyover em Pequim, uma estrada chinesa.
Ao tirar conclusões de dados de decisões de um veículo é também tido em conta os dados de veículos em oito direções à volta desse. Para melhorar ainda mais os Markov Models escondidos.
Depois de treinar os Hidden Markov Models com os dados obtidos, quer com dados discretos como com contínuos, os investigadores usaram-nos para tentar prever as decisões de vários veículos para verificar a precisão do estudo.
Obtendo assim vários gráficos para estudar e retirar várias conclusões.
Como por exemplo este gráfico do artigo, onde perceberam que com Hidden Markov Models com caracterização contínua obtemos uma precisão maior que com caracterização discreta. Chegando a até 85% de precisão com meio segundo para fazer a predição, e mantendo-se mais fiável com tempos maiores. Com caracterização discreta e 4 segundos e meio de tempo de fazer a predição descia até 65%.



(b) Prediction accuracy when discrete characterization or continuous characterization is applied.

E também concluíram que com mais variáveis guardadas e usadas no modelo de markov, ainda mais precisos eram as previsões feitas.

Este artigo fala de algo que pode ser extremamente importante para o nosso futuro. É muito provável que carros autônomos sejam introduzidos nas nossas estradas num futuro próximo e para tal é preciso termos a certeza de que iria ser seguro. E para essa conclusão vemos que o uso de modelos de Markov é uma ajuda muito grande, conseguindo chegar com eles a uma precisão de 80 por cento em média (entre 0,5 a 4,5 segundos de dados obtidos).

Referências:

Shiwen Liu, Kan Zheng, Long Zhao, Pingzhi Fan (2019). "A Driving Intention Prediction Method Based on Hidden Markov Model for Autonomous Driving" https://arxiv.org/abs/1902.09068

Anexo A1

Grupo que inclui o Aluno com o Nº 86617
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):	0.0516	0.0828	0.1332	0.1380	0.1248	0.1120	0.0984	0.0752	0.0612	0.0480	0.0416	0.0248	0.0084
Probabilidade (entregas):	0.0184	0.0632	0.1432	0.1940	0.1828	0.1652	0.1112	0.0620	0.0352	0.0136	0.0076	0.0012	0.0024

FILIAL 2

Número de clientes:	0	1	2	3	4	5	6	7	8	9	10	11	12
Probabilidade (pedidos):	0.0360	0.1216	0.2080	0.2216	0.1816	0.1156	0.0692	0.0304	0.0096	0.0036	0.0020	0.0004	0.0004
Probabilidade (entregas):	0.0312	0.0564	0.0920	0.1176	0.1460	0.1176	0.1204	0.0996	0.0792	0.0628	0.0432	0.0240	0.0100

Figura 2Dados de probabilidades fornecidos

Anexo A2

Cada linha corresponde ao estado inicial e cada coluna ao estado final

```
0.018400 0.063200 0.143200 0.194000 0.182800 0.165200 0.111200 0.062000 0.035200 0.013600 0.007600 0.001200 0.002400 somatorio = 1.000000
0.017451 0.060888 0.139072 0.191379 0.183378 0.166108 0.113986 0.064539 0.036583 0.014715 0.007910 0.001530 0.002462 somatorio = 1.000000
0.015927 0.056229 0.130136 0.183044 0.181684 0.168143 0.119366 0.071399 0.041341 0.017886 0.009521 0.002370 0.002953 somatorio = 1.000000
0.013476 0.048739 0.114821 0.167342 0.174842 0.168794 0.128594 0.083332 0.051771 0.025521 0.013492 0.004834 0.004444 somatorio = 1.000000
0.010937 0.040105 0.096291 0.145017 0.160685 0.164380 0.136696 0.099349 0.067402 0.038932 0.021954 0.009687 0.008565 somatorio = 1.000000
0.008641 0.031975 0.077673 0.120146 0.139757 0.152420 0.139022 0.113592 0.086764 0.057258 0.036114 0.018949 0.017688 somatorio = 1.000000
0.006580 0.024661 0.060583 0.095839 0.116141 0.133464 0.133110 0.121428 0.104008 0.079040 0.055113 0.033825 0.036208 somatorio = 1.000000
0.004769 0.018192 0.045397 0.073750 0.092936 0.111579 0.119467 0.120357 0.114481 0.098409 0.077485 0.053454 0.069723 somatorio = 1.000000
0.003386 0.013012 0.032912 0.054744 0.071690 0.089698 0.101643 0.110414 0.115426 0.110507 0.097305 0.076307 0.122956 somatorio = 1.000000
0.002260 0.008887 0.022836 0.039150 0.053369 0.069528 0.083067 0.095601 0.107123 0.112773 0.109770 0.096520 0.199115 somatorio = 1.000000
0.001376 0.005611 0.014871 0.026636 0.038312 0.052052 0.065489 0.079387 0.093597 0.105507 0.112324 0.109291 0.295545 somatorio = 1.000000
0.000611 0.002864 0.008267 0.016558 0.026264 0.037728 0.050260 0.063856 0.078497 0.092880 0.105308 0.112112 0.404797 somatorio = 1.000000
0.000155 0.000987 0.003536 0.008693 0.016464 0.026117 0.037275 0.049846 0.063631 0.078315 0.092829 0.105254 0.516899 somatorio = 1.000000
```

Figura 4 Matriz P individual da filial 1

```
0.031200 0.056400 0.092000 0.117600 0.146000 0.117600 0.120400 0.099600 0.079200 0.062800 0.043200 0.024000 0.010000 somatorio = 1.000000
0.030077 0.055493 0.090718 0.116678 0.144978 0.118622 0.120299 0.100349 0.079934 0.063390 0.043906 0.024691 0.010864 somatorio = 1.000000
0.026283 0.051305 0.085482 0.112284 0.140603 0.121053 0.120981 0.102777 0.083164 0.066119 0.046879 0.027732 0.015338 somatorio = 1.000000
0.019793 0.042270 0.073890 0.101723 0.130301 0.122586 0.122830 0.107786 0.089836 0.072760 0.053685 0.034699 0.027844 somatorio = 1.000000
0.012879 0.030196 0.056965 0.084458 0.113446 0.118577 0.123741 0.114244 0.099364 0.083066 0.064669 0.045759 0.052635 somatorio = 1.000000
0.007213 0.018706 0.038427 0.062884 0.091024 0.106880 0.119225 0.118932 0.109527 0.095573 0.078534 0.060230 0.092845 somatorio = 1.000000
0.003607 0.010127 0.022821 0.041386 0.066167 0.087741 0.107204 0.116820 0.116574 0.107631 0.093307 0.076315 0.150301 somatorio = 1.000000
0.001448 0.004776 0.011778 0.024009 0.042704 0.064849 0.087871 0.106239 0.115874 0.115813 0.106722 0.092416 0.225502 somatorio = 1.000000
0.000499 0.001851 0.005346 0.012188 0.024463 0.042249 0.064894 0.087538 0.105912 0.115611 0.115500 0.106415 0.317534 somatorio = 1.000000
0.000200 0.000660 0.002079 0.005509 0.012370 0.024281 0.042267 0.064761 0.087407 0.105808 0.115486 0.115377 0.423795 somatorio = 1.000000
0.000087 0.000270 0.000760 0.002150 0.005589 0.012290 0.024289 0.042209 0.064704 0.087361 0.105753 0.115432 0.539105 somatorio = 1.000000
0.000025 0.000108 0.000299 0.000781 0.002173 0.005566 0.012292 0.024273 0.042193 0.064691 0.087346 0.105737 0.654517 somatorio = 1.000000
0.000012 0.000035 0.000122 0.000309 0.000792 0.002162 0.005567 0.012284 0.024264 0.042186 0.064683 0.087338 0.760245 somatorio = 1.000000
```

Figura 6 Matriz P individual da filal 2

```
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
1.000000 0.984407 0.976551 0.961390 0.945411 0.943215 0.925216 0.911093 0.912549 0.876563 0.911277 0.743726 0.924547
2.000000 2.000000 1.985408 1.964368 1.918660 1.880930 1.841957 1.761220 1.722408 1.642266 1.618459 1.407738 1.608646
3.000000 3.000000 3.000000 2.982979 2.944045 2.868672 2.766464 2.660424 2.506033 2.325501 2.288620 1.872692 2.136192
4.000000 4.000000 4.000000 4.000000 3.976365 3.920645 3.783780 3.596961 3.440223 3.124173 2.954575 2.679021 2.554055
4.999999 5.000000 5.000000 5.000000 5.000000 4.968854 4.882713 4.674751 4.423125 4.176323 3.819801 3.485954 3.220537
6.000000 6.000000 6.000000 6.000000 6.000000 5.999999 5.957204 5.838861 5.573739 5.240099 4.973107 4.487937 4.006018
7.000000 6.999999 7.000000 7.000001 7.000000 7.000000 7.000000 6.944780 6.800598 6.474402 6.095655 5.764779 4.936077
8.000000 8.000000 8.000000 8.000000 8.000000 8.000000 8.000000 8.000000 7.934196 7.763916 7.392501 6.950514 6.048733
-1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.075771 -1.267378 -1.689002 -2.808012
-9.999999 -10.000000 -10.000000 -10.000000 -10.000000 -10.000000 -10.000001 -10.000001 -10.000001 -9.999999 -9.999999 -10.084526 -10.298388 -11.603462
-19.000000 -18.999998 -19.000000 -19.000002 -19.000000 -19.000000 -19.000002 -18.999998 -19.000000 -18.999998 -19.000000 -19.093157 -20.376390
-28.000000 -28.000000 -28.000000 -28.000000 -28.000004 -27.999998 -28.000002 -28.000004 -27.999998 -28.000002 -27.999996 -28.000000 -29.197912
```

Figura 3 Matriz R individual da filial 1

```
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
1.000000 0.979759 0.977619 0.971614 0.970798 0.955691 0.964808 0.956807 0.955143 0.955022 0.948508 0.937014 0.887334
2.000000 2.000000 1.973721 1.963835 1.952889 1.930054 1.913110 1.917616 1.895762 1.891541 1.878360 1.836951 1.637597
3.000000 3.000000 3.000000 2.966875 2.953254 2.918946 2.896599 2.853709 2.858622 2.821285 2.799630 2.753490 2.456966
4.000000 4.000000 4.000000 4.000000 3.960397 3.931508 3.892938 3.851769 3.788415 3.796132 3.731903 3.686567 3.400310
5.000000 5.000001 5.000000 5.000000 5.000000 4.947455 4.914850 4.860762 4.806734 4.725025 4.730461 4.640179 4.381940
6.000000 5.999999 6.000000 6.000000 6.000000 6.000000 5.937137 5.895716 5.829533 5.763994 5.662018 5.667148 5.368820
7.000000 7.000000 6.999998 7.000001 6.999999 7.000000 7.000000 6.925993 6.877344 6.799815 6.722313 6.601887 6.377772
8.000000 8.000000 8.000000 8.000000 8.000000 8.000000 8.000000 8.000000 7.915160 7.859501 7.770597 7.681728 7.362567
-1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.095539 -1.158233 -1.258353 -1.627212
-10.000000 -10.000000 -10.000000 -10.000001 -9.999999 -9.999998 -10.000000 -10.000001 -10.000000 -9.999998 -10.106212 -10.175897 -10.609271
-19.000000 -19.000000 -19.000004 -19.000002 -19.000004 -18.999996 -19.000000 -18.999998 -19.000000 -19.000004 -19.000002 -19.116850 -19.586142
-28.000000 -27.999998 -27.999998 -28.000000 -28.000000 -28.000004 -27.999994 -27.999998 -28.000000 -28.000000 -28.000002 -28.000000 -28.568251
```

Figura 5 Matriz R individual da filial 2

Anexo A3

```
//funcao que calcula as matrizes 13x13 P e C, por filial, sem efetuar transferencias
void calculaPs(float pe[13], float pp[13],float m[13][13], float custos[13][13]){


    float prob;
    float probCustos;
    float aux[13][13];

    for(int i=0;i<13;i++){
        for(int j=0;j<13;j++){
            aux[i][j]=0;
        }
    }

    int j;

    for(int i=0; i<13;i++){
        for(int k=0; k<13; k++){ // pedidos                    j e objetivo
            for(int l=0;l<13;l++){ //entregas                  i e hoje
                prob=pp[k]*pe[l];
                probCustos=(minimo(i,(k*30))+((maximo(0,i-8))*-10));
                j=i-(minimo(i,k))+l;
                if(j>12) j =12;
                aux[i][j]+=probCustos*prob;

                m[i][j]+=prob;



            }



        }

    }

    for(int i=0;i<13;i++){
        for(int j=0;j<13;j++){
            custos[i][j]=(aux[i][j]/m[i][j]);
        }
    }

}
```

Figura 7- Função de calculo de matrizes individuais P e R

```c
//estrutura para guardar posiccoes depois de transformadas num estado
typedef struct Par{

    int fil1;
    int fil2;
}Par;

//funcao para transformar posicoes da matriz 169 num estado
Par traduz (int x){

    Par a;

    a.fil1= x/13;


    a.fil2= x%13;


    return a;
}
```

Figura 8- Função que transforma número de linha ou coluna em estado do problema

```c
//funcao que calcula as matrizes 169 Rn e Pn
//1-nao se faz nada
//2-um carro da filial 1 para filial 2
//3-dois carros da filial 1 para a filial 2
//4-tres carros da filial 1 para a filial 2
//5-um carro da filial 2 para a filial 1
//6-dois carros da filial 2 para a filial 1
//7-tres carros da filial 2 para a filial 1
void calculaMatriz169(float p1[13][13], float p2[13][13],  float pk1[169][169], float pk2[169][169], float pk3[169][169], float pk4[169][169], float pk5[169][169], float pk6[169][169], float pk7[169][169],


   Par x1;

   Par x2;


   for(int i=0; i<169;i++) {
      for (int j = 0; j < 169; j++) {


         x1=traduz(j);


         x2=traduz(i);

         //decisao 1
         ck1[i][j] = c1[x2.fil1][x1.fil1] * c2[x2.fil2][x1.fil2];
         pk1[i][j] = p1[x2.fil1][x1.fil1] * p2[x2.fil2][x1.fil2];

         //decisao 2
         if(x2.fil1==0){
            ck2[i][j] = -1;
            pk2[i][j] = 9999;
         }
         else {

            ck2[i][j] = (c1[(x2.fil1) - 1][x1.fil1] * c2[minimo(((x2.fil2) + 1),12)][x1.fil2])-7;
            pk2[i][j] = p1[(x2.fil1) - 1][x1.fil1] * p2[minimo(((x2.fil2) + 1),12)][x1.fil2];

         }
```

Figura 9 Cálculo de matriz P e R da decisão 1 e 2

```
        //decisao 3
        if(x2.fil1==0||x2.fil1==1){
            ck3[i][j] = -1;
            pk3[i][j] = 9999;
        }
        else {

            ck3[i][j] = (c1[(x2.fil1) - 2][x1.fil1] * c2[minimo(((x2.fil2) + 2),12)][x1.fil2])-14;
            pk3[i][j] = p1[(x2.fil1) - 2][x1.fil1] * p2[minimo(((x2.fil2) + 2),12)][x1.fil2];

        }
        //decisao 4

        if(x2.fil1==0||x2.fil1==1||x2.fil1==2){
            ck4[i][j] = -1;
            pk4[i][j] = 9999;
        }
        else {

            ck4[i][j] = (c1[(x2.fil1) - 3][x1.fil1] * c2[minimo(((x2.fil2) + 3),12)][x1.fil2])-21;
            pk4[i][j] = p1[(x2.fil1) - 3][x1.fil1] * p2[minimo(((x2.fil2) + 3),12)][x1.fil2];

        }

        //decisao 5

        if(x2.fil2==0){
            ck5[i][j] = -1;
            pk5[i][j] = 9999;
        }
        else {

            ck5[i][j] = (c1[minimo(((x2.fil2) + 1),12)][x1.fil1] * c2[(x2.fil2) - 1][x1.fil2])-7;
            pk5[i][j] = p1[minimo(((x2.fil2) + 1),12)][x1.fil1] * p2[(x2.fil2) - 1][x1.fil2];

        }
        //decisao 6
```

Figura 11Cálculo das matrizes P e R das decisões 3, 4 e 5

```
        }
        //decisao 6
        if(x2.fil2==0||x2.fil2==1){
            ck6[i][j] = -1;
            pk6[i][j] = 9999;
        }
        else {

            ck6[i][j] = (c1[minimo(((x2.fil2) + 2),12)][x1.fil1] * c2[(x2.fil2) - 2][x1.fil2])-14;
            pk6[i][j] = p1[minimo(((x2.fil2) + 2),12)][x1.fil1] * p2[(x2.fil2) - 2][x1.fil2];

        }
        //decisao 7
        if(x2.fil2==0||x2.fil2==1||x2.fil2==2){
            ck7[i][j] = -1;
            pk7[i][j] = 9999;
        }
        else {

            ck7[i][j] = (c1[minimo(((x2.fil2) + 3),12)][x1.fil1] * c2[(x2.fil2) - 3][x1.fil2])-21;
            pk7[i][j] = p1[minimo(((x2.fil2) + 3),12)][x1.fil1] * p2[(x2.fil2) - 3][x1.fil2];

        }
    }
```

Figura 10 Cálculo das matrizes P e R das decisões 6 e 7

```
//funcao que calcula a matriz Qn
void calculaMatrizQ(float a[169][169], float c[169][169], float q[169][1]){

    float count=0;

    for(int i=0;i<169;i++){

        for(int j=0;j<169;j++){

            count+=a[i][j]*c[i][j];


        }
        q[i][0]=count;
        count=0;
    }
```

Figura 12 Cálculo da matriz Q

```
while(flag<1000){



    multiply(p1,FnMinus1,resultaux);
    sumMatrix(q1,resultaux,result1);

    multiply(p2,FnMinus1,resultaux);
    sumMatrix(q2,resultaux,result2);

    multiply(p3,FnMinus1,resultaux);
    sumMatrix(q3,resultaux,result3);

    multiply(p4,FnMinus1,resultaux);
    sumMatrix(q4,resultaux,result4);

    multiply(p5,FnMinus1,resultaux);
    sumMatrix(q5,resultaux,result5);

    multiply(p6,FnMinus1,resultaux);
    sumMatrix(q6,resultaux,result6);

    multiply(p7,FnMinus1,resultaux);
    sumMatrix(q7,resultaux,result7);


    atualizaFn(result1,result2,result3,result4,result5,result6,result7,Fn);

    minusMatrix(Fn,FnMinus1,Dn);

    flag++;


    for(int x=0;x<169;x++) FnMinus1[x][0]=Fn[x][0];
```

Figura 13 Algoritmo

Anexo A4

71.125000	71.101562	71.117188	71.125000
71.085938	71.085938	71.109375	71.117188
71.085938	71.140625	71.148438	71.093750
71.132812	71.117188	71.109375	71.109375
71.148438	71.070312	71.117188	71.101562
71.109375	71.093750	71.109375	71.101562
71.132812	71.101562	71.101562	71.109375
71.070312	71.109375	71.109375	71.093750
71.132812	71.117188	71.117188	71.101562
71.093750	71.125000	71.085938	71.101562
71.125000	71.093750	71.117188	71.117188
71.117188	71.093750	71.078125	71.101562
71.070312	71.117188	71.117188	71.093750
71.109375	71.085938	71.125000	71.117188
71.101562	71.109375	71.109375	71.093750
71.109375	71.109375	71.085938	71.101562
71.140625	71.078125	71.078125	71.101562
71.179688	71.125000	71.117188	71.093750
71.085938	71.125000	71.117188	71.093750
71.078125	71.078125	71.117188	71.117188
71.054688	71.093750	71.117188	71.117188
71.171875	71.093750	71.093750	
71.117188	71.117188	71.101562	
71.085938	71.093750	71.101562	
71.117188	71.117188	71.109375	
71.093750	71.078125	71.109375	
71.109375	71.109375	71.093750	
71.140625	71.109375	71.117188	
71.132812	71.062500	71.101562	
71.171875	71.101562	71.125000	
71.148438	71.101562	71.093750	
71.101562	71.085938	71.109375	
71.046875	71.117188	71.117188	
71.039062	71.117188	71.125000	
71.156250	71.132812	71.101562	
71.148438	71.125000	71.117188	
71.109375	71.093750	71.101562	
71.085938	71.101562	71.125000	
71.085938	71.085938	71.117188	
71.070312	71.078125	71.109375	
71.070312	71.125000	71.093750	
71.078125	71.093750	71.109375	
71.125000	71.109375	71.101562	
71.156250	71.125000	71.109375	
71.085938	71.085938	71.101562	
71.125000	71.117188	71.109375	
71.156250	71.117188	71.101562	
71.117188	71.085938	71.093750	
	71.117188	71.078125	
	71.093750	71.101562	

Figura 14 Valores obtidos de Dn após 1000 iterações

```
float p1e[13]={0.0184 , 0.0632 , 0.1432 , 0.1940 , 0.1828 , 0.1652 , 0.1112 , 0.0620 , 0.0352 , 0.0136 , 0.0076 , 0.0012 , 0.0024};
float p1p[13]={0.0516 , 0.0828 , 0.1332 , 0.1380 , 0.1248 , 0.1120 , 0.0984 , 0.0752 , 0.0612 , 0.0480 , 0.0416 , 0.0248 , 0.0084};
float p2e[13]={0.0312 , 0.0564 , 0.0920 , 0.1176 , 0.1460 , 0.1176 , 0.1204 , 0.0996 , 0.0792 , 0.0628 , 0.0432 , 0.0240 , 0.0100};
float p2p[13]={0.0360 , 0.1216 , 0.2080 , 0.2216 , 0.1816 , 0.1156 , 0.0692 , 0.0304 , 0.0096 , 0.0036 , 0.0020 , 0.0004 , 0.0004};
```

Figura 15 Único Input, incluído no próprio programa são os dados probabilísticos