

Universidade do Minho

2ºSemestre 2020/21

(MIEI, 3ºAno)

Modelos Estocásticos de Investigação Operacional

Trabalho Prático

(Problema de Gestão de Inventários)

Identificação do Grupo

<u>Número:</u>	<u>Nome Completo:</u>
A89542	Carlos Filipe Coelho Ferreira
A89575	Joel Salgueiro Martins
A89471	Manuel João Ferreira Moreira
A89544	Sara João Carvalho Dias

Data de Entrega: 2021-04-26

Conteúdo

1	Introdução	3
2	Descrição e formulação do problema	3
3	Construção, implementação e execução do modelo	5
3.1	Problema 1	5
3.2	Problema 2	7
3.3	Problema 3	8
3.3.1	Inserção de Dados	9
3.3.2	Geração de Probabilidades	10
3.3.3	Simulações	10
3.3.4	Resultado Ideal	11
3.3.5	Simulação do Ano	12
3.3.6	Resultados	14
3.3.7	Resultado em Épocas	14
4	Conclusão	15
A	Dados	16
B	Problema 1	18
C	Problema 2	19
D	Problema 3	20
E	Código do problema 3	22

Lista de Figuras

1	Resultados	14
2	Exemplo 1	14
3	Exemplo 2	14
4	Dados fornecidos	16
5	Tratamento de dados	17
6	Problema 1	18
7	Problema 2	19
8	Simulador Q3	20
9	Divisão da procura	21

1 Introdução

Este relatório tem como objetivo explicar todo o processo de desenvolvimento do projeto da Unidade Curricular Modelos Estocásticos de Investigação Operacional.

Para a resolução do projeto proposto foram precisos todos os conceitos aprendidos nas aulas práticas e teóricas, com o objetivo de os aplicar a uma situação de apoio à decisão da política de gestão de inventários num processo realista.

Todas as etapas de desenvolvimento da solução proposta para o problema serão aqui especificadas e devidamente documentadas.

2 Descrição e formulação do problema

A empresa Café&Afins importa café do Brasil e distribui-o por vários países da Europa. Nos últimos três anos, as vendas da empresa têm aumentado consideravelmente, e o Sr. Gervásio – responsável pela gestão do armazém –, deparando-se com alguns problemas no uso atual da política nível de encomenda, decidiu optar por uma política (s, S) de gestão de inventário.

A política (s, S) tem um comportamento semelhante ao da política do tipo Ciclo de Encomenda. Ao fim de um período constante de tempo (t) , neste caso de 4 semanas, é feita uma revisão e é averiguado se se deve ou não efetuar uma encomenda. No caso da política (s, S) , para a encomenda ser concretizada é também necessário que o stock em mão seja inferior a s . A quantidade a encomendar é a diferença entre um nível preestabelecido (S) e o stock em mão.

O prazo de entrega de uma encomenda (l) pode ser igual a uma, duas ou três semanas, com probabilidades p_1 , p_2 e p_3 , sendo estas calculadas com base nas seguintes equações:

$$p_1 = 0.21 + \frac{d_1}{100}, p_2 = 0.52 + \frac{d_2}{100}, p_3 = 1 - p_1 - p_2$$

, com d_1 e d_2 o antepenúltimo e penúltimo dígitos do maior número mecanográfico do grupo de trabalho, respetivamente. Ou seja, sendo esse número 89575, $d_1 = 5$ e $d_2 = 7$. Desta forma, temos as probabilidades $p_1 = 0.26$, $p_2 = 0.59$ e $p_3 = 0.15$.

Existem vários outros dados conhecidos relativos à Café&Afins, tais como os custos de transporte do café, a taxa de juro anual correspondente à posse de inventário de café, o preço médio de

compra do café, o custo de quebra, entre outros.

Tal como aconteceu com as probabilidades do prazo de entrega, também o custo de quebra (C_2) foi calculado a partir de uma fórmula dada: $C_2 = 20 + 2 * d_3$, com d_3 o último dígito do mesmo número mecanográfico mencionado acima, ou seja, $d_3 = 5$. Assim, $C_2 = 30$ euros/saco.

Uma vez que em todas as empresas é necessário efetuar análises, por exemplo do funcionamento previsto do sistema de gestão, estatísticas gerais e análises de viabilidade, surge a necessidade de implementar um modelo de simulação para esta empresa.

Desta forma, o problema passa por, numa primeira fase, perceber bem a política que a empresa pretende adotar e, conseqüentemente, trabalhar com os dados disponíveis. Para tal, precisamos de construir uma folha de cálculo onde se encontre implementado um modelo de simulação do funcionamento do sistema de gestão pretendido, com o menor número de pressupostos e simplificações possível.

Os dados obtidos com o modelo gerarão estatísticas que irão colaborar na análise das medidas de desempenho do sistema, tais como as quebras, os custos e o lucro, de modo a que possa ser inferida a eficiência da política.

Posteriormente, tendo o modelo de simulação corretamente construído, é ainda necessário simular o funcionamento do sistema tendo como objetivo determinar quais serão os melhores valores recomendados para s e S .

3 Construção, implementação e execução do modelo

3.1 Problema 1

No enunciado foi-nos pedido que estimássemos analiticamente os valores dos parâmetros da política nível de encomenda que teriam sido mais adequados para o ano de 2020. Foi-nos pedido também que calculássemos quanto é que a empresa poderia ter poupado em custos e ou evitado em quebras de *stock*, ao longo desse ano, se tivesse usado parâmetros mais racionais na sua política de gestão.

Para podermos resolver este problema, começamos por calcular o valor da variação do valor de procura (σ_r) para cada semana do ano de 2020, com a ajuda da nossa folha de cálculo, e o valor médio desta variação ($\sigma_{r_{avg}} = 88.67$). Para tal, utilizamos a seguinte fórmula, subtraindo o valor da procura (r_i), em cada semana i , ao valor médio da procura anual ($r_{avg} = 484.46$ sacos de café), resultados que estão explícitos na folha de cálculo:

$$\sigma_r = |r_i - r_{avg}|$$

De seguida, calculamos a probabilidade do prazo de entrega ser igual a uma, duas ou três semanas, através dos valores demonstrados na secção **2**, bem como a média dos vários valores possíveis do prazo de entrega ($LT_{avg} = 1.89$ semanas). Para além disso, calculamos ainda a variação do prazo de entrega (σ_{LT}), para cada semana i , e o seu valor médio ($\sigma_{LT_{avg}} = 0.77$), utilizando a seguinte expressão:

$$\sigma_{LT} = |LT_i - LT_{avg}|$$

Através das fórmulas da procura no prazo de entrega, conseguimos calcular o valor de μ_{DDLT} e σ_{DDLT} .

$$\mu_{DDLT} = r * LT \Leftrightarrow \mu_{DDLT} = r_{avg} * LT_{avg} \Leftrightarrow \mu_{DDLT} = 484.46 * 1.89 = 915.6294 \text{ sacos}$$

$$\begin{aligned} \sigma_{DDLT} &= \sqrt{LT * \sigma_r^2 + r^2 * \sigma_{LT}^2} \Leftrightarrow \sigma_{DDLT} = \sqrt{LT_{avg} * \sigma_{r_{avg}}^2 + r_{avg}^2 * \sigma_{LT_{avg}}^2} \\ &= \sqrt{1.89 * 88.67^2 + 484.46^2 * 0.77^2} = 392.4467 \text{ sacos} \end{aligned}$$

Tendo o valor de $S = 1200$ sacos, dado no enunciado, e dando uso à formula $S = \mu_{DDLT} + Z\sigma_{DDLT}$, conseguimos chegar a $Z = 0.7246$, que, por sua vez, através da formula $Z = \frac{3N}{100}$ nos

dá o valor de $N = 24$. Com este dado, e pela tabela dada nas aulas, chegamos aos valores do 1º e 2º integrais: $1 \text{ integral} = 0.234413$ e $2 \text{ integral} = 0.134665$.

Em caso de distribuições normais de nível de encomenda, sabemos que $P[DDLT > S] = 1 \text{ integral}$ e que $E[DDLT > S] = 2 \text{ integral} * \sigma_{DDLT}$, sendo fácil o seu cálculo.

$$P[DDLT > S] = 0.234413$$

$$E[DDLT > S] = 0.134665 * 392.4467 = 52.8488$$

De seguida, procedemos ao cálculo do custo de posse, C_1 , custo de quebra, C_2 , e o custo de encomenda, C_3 . Sabendo que $C_1 = b * i$, sendo b o preço médio por saco dado no enunciado, $b = 115$ euros, e i a taxa de juro anual, $i = 0.15$ que é convertida para taxa semanal, então temos o valor de $C_1 = 115 * (0.15/50) = 0.345$ euros/saco/semana. Quanto ao valor de C_2 , este foi já calculado na secção 2. Para o custo de encomenda, C_3 , o seu valor é-nos dado no enunciado $C_3 = 1500$ euros. Tendo o valor de q dado no enunciado, $q = 1700$ sacos, temos agora todos os valores necessários ao cálculo de CT .

$$CT = C_1\left(\frac{q}{2} + S - \mu_{DDLT}\right) + C_2\frac{r}{q}E[DDLT > S] + C_3\frac{r}{q}$$

$$\Leftrightarrow CT = 1270.6428 \text{ euros}$$

Por fim, procedemos ao cálculo do valor ótimo. Começamos por calcular o valor de QEE , dado pela fórmula $QEE = \sqrt{\frac{2rC_3}{C_1}} = 2052.4852$ sacos, seguido de $P[DDLT > S] = \frac{C_1q}{C_2r} = 0.0487$. A partir deste valor e com a ajuda das tabelas fornecidas nas aulas, vemos que $N = 58$ e $2 \text{ integral} = 0.014502$, calculando assim o valor de $E[DDLT > S] = 2 \text{ integral} * \sigma_{DDLT} = 6.6877$.

Começando uma nova iteração, podemos agora calcular q seguindo a seguinte fórmula: $q = \sqrt{\frac{2r(C_2E[DDLT > S] + C_3)}{C_1}} = 2185$ sacos. Com este novo valor, conseguimos recalculer $P[DDLT > S] = 0.0519$, e novamente através da tabela vemos que $N = 55$, $2 \text{ integral} = 0.01573$ e podemos recalculer $E[DDLT > S] = 7.2367$.

Avançando para mais uma iteração, começamos por recalculer q com o novo valor de $E[DDLT > S]$. Assim temos que $q = 2196$ sacos e que $P[DDLT > S] = 0.0521$. Dando novamente uso às tabelas, vemos que $N = 55$, logo o sistema converge, dando por terminadas as iterações, com $q = 2196$ sacos.

Para terminar, recalculamos o valor de S utilizando a fórmula dada acima, $S = 1563$ sacos, e o custo total com a fórmula utilizada anteriormente, $CT = 980.9630$ euros. Deste modo, a empresa poderia ter poupado $1270.6428 - 980.9630 = 289.6798$ euros semanais em 2020 se tivesse usado parâmetros mais racionais na sua política de gestão.

3.2 Problema 2

Para a questão 2 começamos por calcular o valor da média dos valores da procura semanal, uma estimativa que consiste na extrapolação do valor segundo a regressão linear dos valores médios homólogos verificados nos últimos anos. Com a ajuda da calculadora gráfica, conseguimos obter as seguintes equações que definem a procura e o desvio:

$$r = 67.15t + 281$$

$$\sigma_r = 10.41t + 58.39$$

No caso deste exercício, como estamos a estudar o ano de 2021 (4^{o} ano estudado, logo $t = 4$), então os valores da procura e do desvio são: $r = 67.15 * 4 + 281 = 549.6$ e $\sigma_r = 10.41 * 4 + 58.39 = 100.03$.

Sabendo que as fórmulas para o μ e σ na política ciclo de encomenda são dadas por

$$\mu_{DDPP} = r * (t + LT)$$

$$\sigma_{DDPP}^2 = (t + LT) * \sigma_r^2 + r^2 * \sigma_{LT}^2$$

então, utilizando os valores calculados e mencionados anteriormente, resta-nos que $\mu_{DDPP} = 3237.144$ e $\sigma_{DDPP} = 487.8799$.

Considerando um ciclo (t) de 4 semanas, os valores de procura e desvio de procura calculados acima e ainda os valores de C_1, C_2 e C_3 provenientes do Problema 1, podemos calcular os valores dos parâmetros da política (s, S).

Começando pelo cálculo de $P[DDPP > S]$, e impondo a condição de uma quebra a cada dois anos, temos que

$$P_{DDPP} \leq \frac{1}{\# \text{ciclos em dois anos}} = \frac{1}{\frac{2*50}{4}} = 0.04$$

Vendo pelas tabelas auxiliares de distribuição normal, como $P[DDPP > S] = 1 \text{ integral}$, fazendo correspondência com o N averiguamos que $N = 58$ e o respetivo 2° integral é 0.014502. A partir do valor de N calculamos o valor de Z , e ficamos com $Z = 1.74$. Deste modo, já nos é possível calcular o valor de S , através da expressão $S = \mu_{DDPP} + Z\sigma_{DDPP}$. Substituindo pelos valores obtidos e mencionados anteriormente, temos que $S = 4086$ sacos. Para além disto, calculamos ainda $E[DDPP > S]$ através da fórmula $E[DDPP > S] = 2 \text{ integral} * \sigma_{DDPP}$, e obtemos o valor 7.0752 .

Conseguimos então calcular o valor de q^* pela expressão $\sqrt{\frac{2rC_3}{C_1}}$ e obter o valor $q^* = 2186$ sacos. Consequentemente, conseguimos obter os valores de s e de CT . Com a equação $S = q + s - \frac{rt}{2}$, conseguimos obter o valor $s = 2999.2 \approx 3000$. Podemos agora calcular o valor de CT , dado pela expressão $CT = C_1(S - rLT - \frac{rt}{2}) + C_2(\frac{1}{t})E[DDPP > S] + C_3\frac{1}{t}$, obtemos $CT = 1100.14$ euros.

Assim, após a realização de todos os cálculos necessários, conseguimos chegar a uma estimativa dos valores dos parâmetros da política (s, S) para o ano 2021, pelo que o valor máximo do *stock* em mão deverá ser inferior a 3000 sacos e o nível de referência máximo deverá ser 4086 sacos de café.

3.3 Problema 3

Resolver analiticamente um problema de gestão de *stocks* nem sempre é possível de forma viável, tal como podemos ver pelo cálculo de s e S nesta política, que não é garantidamente a melhor opção.

No entanto, tirando partido das funcionalidades computacionais, que, como sabemos, são extremamente rigorosas, rápidas e eficazes, consegue-se então simular milhões de situações numa questão de meros minutos.

Em teoria é simples, oferecendo-se vários dados fixos como *input* ao simulador, que por sua vez, baseado numa certa probabilidade, gera vários outros dados, o simulador acaba por retornar inúmeros resultados que podem ser comparados para concluir em média qual a melhor estratégia.

Com esta exata perspetiva em mente, fizemos uso da linguagem de programação Java para pôr em prática o desenvolvimento deste problema.

Durante este capítulo, pretendemos apresentar uma versão mais simplificada do código elaborado, demonstrando as suas características e o seu funcionamento. Pretendemos depois apresentar os resultados que o algoritmo conclui e a razão para estes serem adotados.

3.3.1 Inserção de Dados

O primeiro passo que se deve realizar para utilizar o simulador é oferecer o conjunto de dados iniciais correspondentes à nossa situação.

```
1  int stockRechargeMin = 2500;
2  int stockRechargeMax = 3500;
3  int stockRechargeInt = 2;
4  int stockLevelMin = 3500;
5  int stockLevelMax = 4500;
6  int stockLevelInt = 2;
7  double maxQuebras = 1000;
8  double media = 549.6, desvio = 100.03;
9  double c1 = 0.345, c2 = 30, c3 = 1500;
10 int ciclo = 4;
11 int duracao = 50;
12 int[] probPrazo = {26,59};
13 int tries = 50;
```

Neste caso, deve-se oferecer os intervalos de s e S que gostaríamos de testar, assim como a sua precisão. O exemplo acima corresponde a testar os valores de s de 2500 a 3500 com precisão 2.

Deve-se introduzir um número máximo de quebras por ano, ou seja, a taxa de serviço desejada. A média e o desvio correspondem a uma distribuição normal da procura semanal, e os custos C_1 , C_2 e C_3 são de posse, de quebra e de encomenda, respetivamente.

A variável *ciclo* corresponde ao intervalo no qual se vai fazer a gestão de inventário, que neste caso é de 4 em 4 semanas.

A duração representa o número de semanas num ano, e o array *probPrazo* contém as probabilidades em percentagem da distribuição do prazo de entrega.

Por fim, a variável *tries* corresponde ao número de anos que queremos simular para os mesmos valores de s e S , sendo que cada simulação tem diferentes fluxos de procura e prazos de entrega.

3.3.2 Geração de Probabilidades

```
1 private static void gerarSequencias(int[][] procura, int[][] prazosEntregas,
2     int tries, int duracao, double media, double desvio, int [] probPrazo) {
3
4     Random gaussian = new Random();
5     Random r = new Random();
6     int tempoEntrega; double procura;
7     for (int t1 = 0; t1 < tries; t1++)
8         for (int d = 0; d < duracao; d++) {
9             procura = gaussian.nextGaussian() * desvio + media;
10            procura[t1][d] = (int) procura;
11            tempoEntrega = r.nextInt(100);
12            //Transformar 0-100 em 1,2 ou 3
13            prazosEntregas[t1][d] = tempoEntrega;
14        }
15 }
```

Como se pretende simular para um mesmo *input* (s, S) um número $n = tries$ de simulações, com diferentes procura e prazos de entrega, é importante então gerar um conjunto de dados distintos a cada uma dessas tentativas. Para isso, o objetivo é formar uma matriz de dados, onde cada linha representa uma diferente simulação do ano 2021, e cada coluna uma semana desse ano.

É importante estabelecer estes dados probabilísticos antecipadamente, porque assim garante-se que durante a otimização de diferentes (s, S), estes são sujeitos às mesmas condições de procura e prazo de entrega.

3.3.3 Simulações

```
1 for (int S = Min; S <= Max; S += Int)
2     for (int s = Min; s <= Max; s += Int) {
3         for (int t = 0; t < tries; t++) {
4             Otimizacao o = new Otimizacao(stockLevel, stockRecharge, stockLevel, duracao,
5                 c1, c2, c3, ciclo, procura[t], prazosEntregas[t]);
6             double custoContro l= o.calculaCusto()/duracao;
7             custo+=custoControl;
8             custos[t] = custoControl;
9             artigosQuebra += o.getArtigosQuebra();
10            quebras += o.getQuebras();
11        }
```

```

12     double custoMedio = custo/tries;
13     double quebrasMedias = quebras/tries;
14     artigosQuebra = artigosQuebra/tries;
15     double desvioR = 0;
16     for (int j = 0; j < tries; j++)
17         desvioR += Math.abs(custoMedio - custos[j]);
18     desvioR = desvioR / tries;
19 }

```

Com os dados impostos, pode-se começar a processar as diferentes simulações. Para isso, usamos dois ciclos *'for'* que representam a formação do par (s, S) , onde para cada um se processa um número **n** de diferentes simulações. Posteriormente, podemos obter o desempenho de (s, S) calculando a média dos diferentes resultados obtidos.

Concluindo, conseguimos obter valores para a variável **custoMedio**, que representa o custo por semana em média; **quebrasMedias**, que representa as quebras durante um ano em média; **artigosQuebra**, que representa o número de artigos em situação de quebra em média num ano; e finalmente **desvioR**, que representa o desvio médio dos custos semanais.

3.3.4 Resultado Ideal

```

1  if ((custoMedio + desvioR) <= (custoIdeal + devioIdeal) && quebrasMedias <= maxQuebras)
2  {
3      custoIdeal = custoMedio;
4      stockLevelIdeal = stockLevel;
5      stockRechargeIdeal = stockRecharge;
6      artigosQuebraIdeal = artigosQuebra;
7      quebrasMediaIdeal = quebrasMedias;
8      desvioIdeal = desvioR;
9  }

```

Após rodar todas as simulações referentes a um valor (s, S) e ter os resultados correspondentes, devemos então ver se esse (s, S) é a melhor opção obtida até ao momento. Para isso, devemos indicar qual o critério de comparação desejado.

Neste caso, decidiu-se ter em atenção o custo médio e a sua derivação. Foi importante também garantir um nível de serviço mínimo (ou seja, não ultrapassar o número máximo de quebras). Se a situação atual for então melhor devemos guardá-la, sendo esta comparada a outras no futuro.

3.3.5 Simulação do Ano

```
1 double calculaCusto() {
2     int[] recargas = new int[duracao];
3     Arrays.fill(recargas, 0);
4     double custo = 0;
5     for (int i = 0; i < duracao; i++) {
6         this.verificarInventario(i, recargas); //Contagem de inventário, Encomendar
7         custo += this.recargas(i, recargas); //Receber Encomendas - c3
8         custo += this.custoManutencao(i); //Custo De Posse - c1
9         custo += this.quebras(); //Ocorrência de Quebras - c2
10    }
11    return custo;
12 }
```

A simulação de um ano faz-se percorrendo o número total de semanas. Em cada semana devemos ter em atenção: a contagem de inventário e se é necessário encomendar mais; a chegada de encomendas ao armazém; o custo de manter as unidades no armazém e as ocorrências de situações de quebra. Resumindo, pretende-se simular uma situação de fluxo (chegada e saída) de unidades durante um ano, e calcular o custo resultante.

```
1 private void verificarInventario(int i, int[] recargas) {
2     if ((i + 1) % ciclo == 0)
3         if (stock < stockRecharge) {
4             int tempoEntrega = this.prazosEntrega[this.recargas++];
5             recargas[i + tempoEntrega] += stockMax - stock;
6         }
7 }
```

Assim como a política de ciclo comanda, deve-se verificar o inventário de 4 em 4 semanas e fazer encomendas no caso do *stock* atual ser menor que o *s* estabelecido.

Existe um prazo de entrega associado à encomenda, que podemos obter no *array* dos valores aleatórios. Este *array* foi gerado antecipadamente para o efeito, e a indicação da encomenda é feita através de um *array* de recargas na posição *semana atual + prazo de Entrega*. A quantidade a encomendar é, como sabemos, $S - \text{stock atual}$.

```
1 private double recargas(int i, int[] recargas) {
2     if (recargas[i] != 0) {
```

```

3         stock += recargas[i];
4         return c3;
5     } else return 0;
6 }

```

No caso de receber uma encomenda naquela semana, deve-se registar o aumento do *stock* e pagar o valor associado de C_3 .

```

1 private double custoManutencao(int i) {
2     double custo;
3     if (stock >= procuras[i])
4         custo = c1 * (stock - (this.procuras[i]/2));
5     else
6         custo = c1 * (stock/2);
7
8     stock -= procuras[i];
9
10    return custo;
11 }

```

Deve-se pagar o custo de posse, e para isso devemos, tendo em conta a procura semanal, saber qual o *stock* médio durante essa semana. Tomando um exemplo: para um valor de *stock* no início da semana igual a 500 e procura igual a 100, em média o armazém continha 450 unidades durante a semana. No final, devemos retirar ao nosso *stock* a procura durante essa semana.

```

1 private double quebras(boolean debug) {
2     if (this.stock < 0) {
3         this.quebras++;
4         this.artigosQuebra += -stock;
5         double custo = c2 * (-stock);
6         this.stock = 0;
7         return custo;
8     } else return 0;
9 }

```

No caso de não satisfazer a procura, ou o *stock* ser menor que 0, deve-se registar essa situação, assim como pagar o custo associado igual a $C_2 * nr \text{ de artigos quebrados}$.

3.3.6 Resultados

```
int stockRechargeMin =2750;
int stockRechargeMax =3200;
int stockRechargeInt = 1;
int stockLevelMin =3800;
int stockLevelMax =4200;
int stockLevelInt = 1;
double maxQuebras = 0.5;
double media = 549.6, desvio = 100.03;
double c1 = 0.345, c2 = 30, c3 = 1500;
int ciclo = 4;
int duracao =50;
int[] probPrazo = {26,59};
int tries = 500;

Em 500 simulações,em media e arredondado as unidades:
Ideal: s=3200, S=3984
CustoTotal/Semana=(1074,59)
Artigos Quebrados/Ano=79 Quebras/Ano=0.474
```

Figura 1: Resultados

As simulações rodadas para cada conjunto de dados (s,S) foram 500 (cada uma delas representando o possível ano de 2021). Obtendo o custo médio nessas 500 simulações para cada input (s,S) , e garantindo que o risco é mínimo, foi possível então concluir que a melhor opção a adotar é $(s = 3250, S = 4013)$ com um custo relacionado de 1078 por semana.

Devido à enorme quantidade de simulações, pode-se concluir que este resultado é bastante viável e próximo ao ideal a impor, usando a política descrita, devendo assim ser adotado pelo Sr. Gervásio.

3.3.7 Resultado em Épocas

```
int stockRechargeMin =3400;
int stockRechargeMax =3850;
int stockRechargeInt = 1;
int stockLevelMin =4500;
int stockLevelMax =4900;
int stockLevelInt = 1;
double maxQuebras = 0.5;
double media = 458.84, desvio = 24.33;
double c1 = 0.345, c2 = 30, c3 = 1500;
int ciclo = 4;
int duracao =23;
int[] probPrazo = {26,59};
int tries = 500;

Em 500 simulações,em media e arredondado as unidades:
Ideal: s=3850, S=4615
CustoTotal/Semana=(1139,40)
Artigos Quebrados/Ano=12 Quebras/Ano=0.256
```

Figura 2: Exemplo 1

```
int stockRechargeMin =2000;
int stockRechargeMax =2400;
int stockRechargeInt = 1;
int stockLevelMin =3100;
int stockLevelMax =3500;
int stockLevelInt = 1;
double maxQuebras = 0.5;
double media = 458.06, desvio = 12.27;
double c1 = 0.345, c2 = 30, c3 = 1500;
int ciclo = 4;
int duracao =27;
int[] probPrazo = {26,59};
int tries = 500;

Em 500 simulações,em media e arredondado as unidades:
Ideal: s=2400, S=3205
CustoTotal/Semana=(886,25)
Artigos Quebrados/Ano=8 Quebras/Ano=0.324
```

Figura 3: Exemplo 2

Numa situação real, é preferível adotar diferentes políticas conforme a altura do ano, devido à existência de épocas onde as procuras semanais pelos produtos do cliente variam de forma significativa. Observando os valores de procura, podemos notar que existe uma grande diferença entre as procuras nas semanas de 23 a 46 e as restantes, sendo esta obviamente uma época de grande tráfego no negócio da Café&Afins.

O que deve ser feito é, então, depois de extrapolar as procuras e desvios do ano 2021 para as épocas de 23 a 46, e 1 a 22 e 47 a 50, utilizar a flexibilidade do simulador em Java para alterar o *input* de dados, incluindo a duração (semanas que um ano tem), e fazer assim o cálculo de (s,S) ótimo 2 vezes, fazendo a média do custo de ambos e comparar com os resultados anteriores.

$$CT = 890 * (\frac{27}{50}) + 1145 * (\frac{23}{50}) = 1007.3$$

Pode-se então concluir que utilizando 2 políticas diferentes no mesmo ano o negócio Café&Afins consegue poupar em média $1078-1007.3=70.7$ por semana.

4 Conclusão

O trabalho desenvolvido, permitiu a obtenção de uma política de gestão de inventário aproximadamente ótima com técnicas de simulação.

Embora para problemas pequenos, como o resolvido neste trabalho, o esforço computacional seja reduzido, o recurso a técnicas de simulação para um determinado espaço de soluções, tem baixa escalabilidade, por isso é impensável em problemas reais.

Anexos

A Dados

ANEXO: Tabela de dados

Grupo de Trabalho 1

MIEI-MEIO 2020/21

VALORES DA PROCURA (EMPRESA Café&Afins)

<u>Semana</u>	<u>ANOS</u>		
	<u>2018</u>	<u>2019</u>	<u>2020</u>
1	307	343	404
2	306	345	426
3	278	320	405
4	307	319	415
5	280	357	384
6	283	304	420
7	293	349	396
8	297	334	418
9	315	334	411
10	284	326	399
11	270	338	405
12	294	302	394
13	270	372	393
14	276	336	422
15	284	357	391
16	289	359	390
17	280	334	401
18	271	334	362
19	279	349	404
20	295	338	404
21	296	334	379
22	284	369	397
23	292	353	416
24	407	458	626
25	439	507	570
26	439	486	542
27	424	477	545
28	431	511	579
29	417	492	601
30	420	530	568
31	435	508	616
32	395	520	574
33	411	503	590
34	431	540	589
35	391	514	579
36	432	493	566
37	440	519	568
38	462	501	572
39	441	523	583
40	443	494	617
41	420	489	611
42	416	489	609
43	419	500	589
44	411	490	601
45	396	489	575
46	430	511	584
47	289	327	431
48	265	360	404
49	289	306	394
50	305	344	405

Obs. P.f., anexe estes dados no relatório

Figura 4: Dados fornecidos

Semana	2018	2019	2020				
1	307	43,16	343	70,74	404	80,46	NMax=89575
2	306	44,16	345	68,74	426	58,46	d1= 5
3	278	72,16	320	93,74	405	79,46	d2= 7
4	307	43,16	319	94,74	415	69,46	d3= 5
5	280	70,16	357	56,74	384	100,46	Procura=67.15*ano+281 (ano=4) 549,6
6	283	67,16	304	109,74	420	64,46	Desvio=10.41*ano+58.39 (ano=4) 100,03
7	293	57,16	349	64,74	396	88,46	
8	297	53,16	334	79,74	418	66,46	
9	315	35,16	334	79,74	411	73,46	
10	284	66,16	326	87,74	399	85,46	
11	270	80,16	338	75,74	405	79,46	
12	294	56,16	302	111,74	394	90,46	
13	270	80,16	372	41,74	393	91,46	
14	276	74,16	336	77,74	422	62,46	
15	284	66,16	357	56,74	391	93,46	
16	269	81,16	359	54,74	390	94,46	
17	280	70,16	334	79,74	401	83,46	
18	271	79,16	334	79,74	362	122,46	
19	279	71,16	349	64,74	404	80,46	
20	295	55,16	338	75,74	404	80,46	
21	296	54,16	334	79,74	379	105,46	
22	284	66,16	369	44,74	397	87,46	
23	292	58,16	353	60,74	416	68,46	
24	407	56,84	458	44,26	626	141,54	
25	439	88,84	507	93,26	570	85,54	
26	439	88,84	486	72,26	542	57,54	
27	424	73,84	477	63,26	545	60,54	
28	431	80,84	511	97,26	479	5,46	
29	417	66,84	492	78,26	601	116,54	
30	420	69,84	530	116,26	568	83,54	
31	435	84,84	508	94,26	616	131,54	
32	395	44,84	520	106,26	574	89,54	
33	411	60,84	503	89,26	590	105,54	
34	431	80,84	540	126,26	589	104,54	
35	391	40,84	514	100,26	579	94,54	
36	432	81,84	493	79,26	566	81,54	
37	440	89,84	519	105,26	568	83,54	
38	462	111,84	501	87,26	572	87,54	
39	441	90,84	523	109,26	583	98,54	
40	443	92,84	494	80,26	617	132,54	
41	420	69,84	489	75,26	611	126,54	
42	416	65,84	489	75,26	609	124,54	
43	419	68,84	500	86,26	589	104,54	
44	411	60,84	490	76,26	601	116,54	
45	396	45,84	489	75,26	575	90,54	
46	430	79,84	511	97,26	584	99,54	
47	289	61,16	327	86,74	431	53,46	
48	265	85,16	360	53,74	404	80,46	
49	289	61,16	306	107,74	394	90,46	
50	305	45,16	344	69,74	404	80,46	
	350,16	67,85	413,74	81,12	484,46	88,67	

Figura 5: Tratamento de dados

B Problema 1

Questão 1:					
r(2020)=	$\sigma_r(2020)=$	Px=	LT=	$\sigma_{LT}=$	
404	80,46	0,26	1	0,89	
426	58,46	0,59	2	0,11	
405	79,46	0,15	3	1,11	
415	69,46	Media=		1,89	0,77
384	100,46				
420	64,46				
396	88,46				
418	66,46	Politica Nivel:			
411	73,46	Usado:			
399	85,46	$\mu_{ddlt}=$	915,6294		
405	79,46	$\sigma_{ddlt}=$	392,4466736		
394	90,46	S=	1200		
393	91,46	Z=	0,724609531		
422	62,46	N=	24		
391	93,46	1 int=	0,234413		
390	94,46	2 int=	0,134665		
401	83,46				
362	122,46	P[DDLT>S]=	0,234413		
404	80,46	E[DDLT>S]=	52,8488313		
404	80,46				
379	105,46	C1=	0,345		
397	87,46	C2=	30		
416	68,46	C3=	1500		
626	141,54	q=	1700		
570	85,54				
542	57,54	CT=	1270,642765		
545	60,54				
479	5,46	Otimo:			
601	116,54	QEE=	2052,485238		
568	83,54	P[DDLT>S]=	0,048721422		
616	131,54	N=	56		
574	89,54	2 int=	0,017041		
590	105,54	E[DDLT>S]=	6,687683764		
589	104,54	q=	2185		
579	94,54	P[DDLT>S]=	0,051867027		
566	81,54	N=	55		
568	83,54	2 int=	0,01844		
572	87,54	E[DDLT>S]=	7,236716661		
583	98,54	q=	2196		
617	132,54	P[DDLT>S]=	0,052128143		
611	126,54	N=	55		
609	124,54	S=	1563		
589	104,54	Q=	2196		
601	116,54	CT=	980,9629629		
575	90,54	Resposta =	289,6798025		
584	99,54				
431	53,46				
404	80,46				
394	90,46				
404	80,46				
484,46	88,67				

Figura 6: Problema 1

C Problema 2

PoliticaCiclo:	
Questão 2:	
Formulas Usadas:	
$q = S - s + rt/2$	
$q = \sqrt{2rC3/C1}$	
$S = \sqrt{2rC3/C1} + s - rt/2$	
Dados:	
t=	4
r=	549,6
σ_r =	100,03
μ_{ddpp} =	3237,144
σ_{ddpp} =	487,8799178
C1=	0,345
C2=	30
C3=	1500
Calculos:	
$p[DDPP > S]$ =	0,04
2 int=	0,014502
$e[DDPP]$ =	7,075234568
N=	58
Z=	1,74
S=	4086
q=	2186
s=	2999,2
ct=	1100,143579

Figura 7: Problema 2

D Problema 3

	stockInicial	procura	stockFinal	Encomenda	Aleatorio	PrazoEntreg	A Encomendar	Custo Quebra	Custo Encont	CustoPosse	Custo	
1	4086	440,3941226	3645,605877	0	12	0	0	0	0	1333,702014	1333,702014	
2	3645,605877	498,2076365	3147,398241	0	29	0	0	0	0	1171,79321	1171,79321	s
3	3147,398241	586,8026238	2560,595617	0	22	0	0	0	0	984,6289405	984,6289405	3000
4	2560,595617	441,9196139	2118,676003	1525,404383	93	1	1525,4043828992	0	1500	807,1743545	2307,174355	S
5	3644,080386	716,0440863	2928,0363	0	61	0	1525,4043828992	0	0	1133,690128	1133,690128	4086
6	2928,0363	382,2281231	2545,808177	0	46	0	1525,4043828992	0	0	944,2381722	944,2381722	c1
7	2545,808177	403,8243699	2141,983807	0	17	0	1525,4043828992	0	0	808,6441172	808,6441172	0,345
8	2141,983807	714,995418	1426,988389	1944,016193	76	1	1944,0161931349	0	1500	615,6477038	2115,647704	c2
9	3371,004582	509,957692	2861,04689	0	86	0	1944,0161931349	0	0	1075,028879	1075,028879	30
10	2861,04689	558,098815	2302,948075	0	49	0	1944,0161931349	0	0	890,7891315	890,7891315	c3
11	2302,948075	645,6343867	1657,313688	0	37	0	1944,0161931349	0	0	683,1451542	683,1451542	1500
12	1657,313688	609,8245544	1047,489134	2428,686312	89	3	2428,686311752	0	0	466,5784868	466,5784868	
13	1047,489134	548,3211472	499,1679866	0	15	2	2428,686311752	0	0	266,7983533	266,7983533	CT
14	499,1679866	582,649109	-83,4811223	0	34	1	2428,686311752	2504,433669074	1500	86,1064777	4090,540147	1158,984222
15	2428,686312	682,725211	1745,961101	0	67	0	2428,686311752	0	0	720,1266787	720,1266787	
16	1745,961101	420,0493667	1325,911734	2340,038899	70	2	2340,0388992302	0	0	529,898064	529,898064	
17	1325,911734	492,9156627	832,9960714	0	98	1	2340,0388992302	0	1500	372,4115964	1872,411596	
18	3173,034971	548,8338531	2624,201117	0	59	0	2340,0388992302	0	0	1000,023225	1000,023225	
19	2624,201117	549,7358934	2074,465224	0	91	0	2340,0388992302	0	0	810,5199439	810,5199439	
20	2074,465224	678,6171066	1395,848117	2011,534776	12	2	2011,5347759036	0	0	598,6290514	598,6290514	
21	1395,848117	606,3736667	789,4744508	0	35	1	2011,5347759036	0	1500	376,968143	1876,968143	
22	2801,009227	621,2537945	2179,755432	0	70	0	2011,5347759036	0	0	859,1819037	859,1819037	
23	2179,755432	527,0024019	1652,75303	0	24	0	2011,5347759036	0	0	661,1077098	661,1077098	
24	1652,75303	482,3549364	1170,398094	2433,24697	89	1	2433,2469698654	0	1500	486,9935689	1986,993569	
25	3603,645064	561,4668721	3042,178191	0	13	0	2433,2469698654	0	0	1146,404511	1146,404511	
26	3042,178191	473,7249213	2568,45327	0	41	0	2433,2469698654	0	0	967,8339271	967,8339271	
27	2568,45327	612,0067326	1956,447538	0	63	0	2433,2469698654	0	0	780,5453893	780,5453893	
28	1956,447538	561,9816201	1394,455917	2129,552462	23	2	2129,5524624863	0	0	578,030846	578,030846	
29	1394,455917	460,3895233	934,0663941	0	67	1	2129,5524624863	0	1500	401,6700987	1901,670099	
30	3063,618857	435,6314319	2627,987425	0	95	0	2129,5524624863	0	0	981,8020835	981,8020835	
31	2627,987425	730,5884251	1897,399	0	73	0	2129,5524624863	0	0	780,6291582	780,6291582	
32	1897,399	404,8251917	1492,573808	2188,601	17	2	2188,6010003715	0	0	584,7703093	584,7703093	
33	1492,573808	516,4658252	976,1079828	0	91	1	2188,6010003715	0	1500	425,8476089	1925,847609	
34	3164,708983	437,955776	2726,753206	0	35	0	2188,6010003715	0	0	1016,277228	1016,277228	
35	2726,753206	724,2344422	2002,518763	0	5	0	2188,6010003715	0	0	815,7994146	815,7994146	
36	2002,518763	490,941498	1511,577265	2083,481237	45	1	2083,4812366358	0	1500	606,1815649	2106,181565	
37	3595,058502	430,7394367	3164,319065	0	55	0	2083,4812366358	0	0	1165,99263	1165,99263	
38	3164,319065	454,1157668	2710,203298	0	16	0	2083,4812366358	0	0	1013,355108	1013,355108	
39	2710,203298	497,8933486	2212,30995	0	7	0	2083,4812366358	0	0	849,1335353	849,1335353	
40	2212,30995	585,0713691	1627,238581	1873,69005	64	1	1873,6900500937	0	1500	662,3221216	2162,322122	
41	3500,928631	614,8736836	2886,054947	0	21	0	1873,6900500937	0	0	1101,754667	1101,754667	
42	2886,054947	527,9416906	2358,113257	0	70	0	1873,6900500937	0	0	904,6190152	904,6190152	
43	2358,113257	502,065748	1856,047509	0	34	0	1873,6900500937	0	0	726,9427321	726,9427321	
44	1856,047509	723,1983463	1132,849162	2229,952491	84	2	2229,9524912521	0	0	515,5846758	515,5846758	
45	1132,849162	494,0870487	638,7621138	0	5	1	2229,9524912521	0	1500	305,6029452	1805,602945	
46	2868,714605	565,8974141	2302,817191	0	100	0	2229,9524912521	0	0	892,0892348	892,0892348	
47	2302,817191	461,8839603	1840,933231	0	54	0	2229,9524912521	0	0	714,7969477	714,7969477	
48	1840,933231	498,4528012	1342,480429	2245,066769	20	2	2245,0667694132	0	0	549,1388564	549,1388564	
49	1342,480429	637,736037	704,7443925	0	6	1	2245,0667694132	0	1500	353,1462618	1853,146262	
50	2949,811162	562,3379887	2387,473173	0	82	0	2245,0667694132	0	0	920,6815478	920,6815478	

Figura 8: Simulador Q3

Semana	2018	2019	2020						
47	289	1,666666667	327	11,62962963	431	28,44444444			
48	265	22,33333333	360	21,37037037	404	1,444444444			
49	289	1,666666667	306	32,62962963	394	8,555555556			
50	305	17,66666667	344	5,37037037	404	1,444444444			
1	307	19,66666667	343	4,37037037	404	1,444444444			
2	306	18,66666667	345	6,37037037	426	23,44444444			
3	278	9,333333333	320	18,62962963	405	2,444444444			
4	307	19,66666667	319	19,62962963	415	12,44444444			
5	280	7,333333333	357	18,37037037	384	18,55555556			
6	283	4,333333333	304	34,62962963	420	17,44444444			
7	293	5,666666667	349	10,37037037	396	6,555555556			
8	297	9,666666667	334	4,62962963	418	15,44444444			
9	315	27,66666667	334	4,62962963	411	8,444444444			
10	284	3,333333333	326	12,62962963	399	3,555555556			
11	270	17,33333333	338	0,62962963	405	2,444444444			
12	294	6,666666667	302	36,62962963	394	8,555555556	a=	57,61111111	0,03703704
13	270	17,33333333	372	33,37037037	393	9,555555556	b=	227,617284	12,1179698
14	276	11,33333333	336	2,62962963	422	19,44444444			
15	284	3,333333333	357	18,37037037	391	11,55555556			
16	269	18,33333333	359	20,37037037	390	12,55555556	4	458,061728	12,266118
17	280	7,333333333	334	4,62962963	401	1,555555556			
18	271	16,33333333	334	4,62962963	362	40,55555556			
19	279	8,333333333	349	10,37037037	404	1,444444444			
20	295	7,666666667	338	0,62962963	404	1,444444444			
21	296	8,666666667	334	4,62962963	379	23,55555556			
22	284	3,333333333	369	30,37037037	397	5,555555556			
23	292	4,666666667	353	14,37037037	416	13,44444444			

Figura 9: Divisão da procura

E Código do problema 3

```
1  import java.util.Random;
2  import java.util.Arrays;
3
4  public class Otimizacao {
5      int stock;
6      int stockRecharge;
7      int stockMax;
8      int duracao;
9      int ciclo;
10     int[] procura;
11     int[] prazosEntrega;
12     double c1;
13     double c2;
14     double c3;
15     int artigosQuebra;
16     int quebras;
17     int recargas;
18     boolean quebrou;
19
20     public Otimizacao(int stockInicial,
21                      int stockRecharge, int stockMax, int duracao,
22                      double c1, double c2, double c3, int ciclo,
23                      int[] procura,int[] prazosEntrega) {
24         this.stock = stockInicial;
25         this.stockRecharge = stockRecharge;
26         this.stockMax = stockMax;
27         this.duracao = duracao;
28         this.ciclo = ciclo;
29         this.procura = procura;
30         this.prazosEntrega=prazosEntrega;
31         this.c1 = c1;
32         this.c2 = c2;
33         this.c3 = c3;
34         this.artigosQuebra = 0;
35         this.quebras = 0;
36         this.recargas=0;
37         this.quebrou=false;
38     }
39
40     public int getArtigosQuebra() {
41         return artigosQuebra;
42     }
43
44     public int getQuebras() {
45         return quebras;
```

```

46     }
47
48     double calculaCusto(boolean debug) {
49         int[] recargas = new int[duracao];
50         Arrays.fill(recargas, 0);
51         double custo = 0;
52         for (int i = 0; i < duracao; i++) {
53             if (debug) System.out.println("Semana:" + (i + 1));
54
55             this.verificarInventario(i, recargas, debug); //encomendar se necessário
56
57             if (debug) System.out.println("No inicio da semana stock estava em:" + stock);
58
59             custo += this.recargas(i, recargas, debug); //encomendar e pagar c3
60
61             custo += this.custoManutencao(i, debug); //pagar c1
62
63             custo += this.quebras(debug); //pagar c2
64
65             if (debug) System.out.println("Stock esta em: " + stock);
66             if (debug) System.out.println("Custo esta em:" + custo + "\n");
67         }
68         return custo;
69     }
70     private void verificarInventario(int i, int[] recargas, boolean debug) {
71         if ((i + 1) % ciclo == 0) { //de 4 em 4 semanas {4,8,12...}
72             if (stock < stockRecharge) { //Encomendar ou não?
73                 int tempoEntrega = this.prazosEntrega[this.recargas++]; // prazoEntrega random
74                 if (debug) System.out.println("Detetou recarga! " + stock + "<" + stockRecharge +
75                     ". Encomendou " + (stockMax - stock) + " unidades" +
76                     ". Entrega daqui a " + tempoEntrega + " semana(s)");
77                 //uma simulação(ano) dura 50(duracao) semanas,
78                 //se entrega chegar após 50 (proximo ano) já não a recebe.
79                 if (i + tempoEntrega < duracao)
80                     // Estamos na semana x e vamos receber entrega daqui a 3 semanas,
81                     // vamos receber a entrega na x+3 semana
82                     //encomendamos stockMax-stock Artigos, politica de ciclo
83                     recargas[i + tempoEntrega] += stockMax - stock;
84             } else
85                 if (debug) System.out.println("Não é preciso recarga " + stock + ">" + stockRecharge);
86         }
87     }
88     private double recargas(int i, int[] recargas, boolean debug) {
89         if (recargas[i] != 0) { //Chegou alguma encomenda esta semana?
90             if (debug)
91                 System.out.println("No inicio da semana houve uma recarga de " + recargas[i] +
92                     " pagou-se " + c3);
93             stock += recargas[i]; //por encomenda no stock
94             this.quebrou=false;

```



```

95         return c3;                //pagar custo de encomenda realizada
96     } else return 0;
97 }
98 private double custoManutencao(int i, boolean debug) {
99     if (debug) System.out.println("Procura esta semana foi:" + procuras[i]);
100    double custo;
101    if (stock >= procuras[i]) { //Se não for situação de quebra
102        //Se stock=50 e procura=20, no inicio da semana tera 50 e no fim 30
103        //durante a semana terá em media 40 artigos
104        //aos quais devemos multiplicar c1
105        custo = c1 * (stock - (this.procuras[i] / 2));
106    } else {
107        // Se no inicio da seman tem 50 e no fim 0, em média durante a semana tera 25
108        custo = c1 * (stock/2);
109    }
110    if (debug) System.out.println("Pagou-se " + (custo) + " em custos de manutenção");
111    stock -= procuras[i]; //Retirar procura ao stock
112    return custo;
113 }
114 private double quebras(boolean debug) {
115     if (this.stock < 0) { //Houve quebras de stock?
116         if (debug) System.out.println("Situação de quebra de " + (-stock)
117             + " artigos.Pagou-se" + c2 * (-stock));
118         if(!quebrou){
119             this.quebras++; //Se sim +1 em quebras durante este ano
120             this.quebrou=true;
121         }
122         this.artigosQuebra += -stock; // se stock=-50, houve uma quebra de 50 artigos
123         double custo = c2 * (-stock); //pagar custo de quebra por artigo
124         this.stock = 0;                //resetar stock
125         return custo;
126     } else return 0;
127 }
128
129
130 public static void main(String[] args) {
131
132     int stockRechargeMin =2750; //s minimo
133     int stockRechargeMax =3250; //s maximo    for(s=minimo;s<maximo;s+=Precisao)
134     int stockRechargeInt = 1;    //s precisão
135     int stockLevelMin =3800;    //S minimo
136     int stockLevelMax =42000;    //S maximo    for/S=minimo;S<maximo;S+=Precisao(
137     int stockLevelInt = 1;    //S precisao
138     double maxQuebras = 0.5;    // 1 Em cada 2 anos
139     double media = 549.6, desvio = 100.03; //Media e desvio de procura semanal
140     double c1 = 0.345, c2 = 30, c3 = 1500; //custo posse,quebra,encomenda
141     int ciclo = 4;                //Revisão de x em x semanas
142     int duracao = 50;            //Duração de um ano
143     int[] probPrazo = {26,59};

```

```

144     int tries = 500;
145     boolean debug = false;
146
147
148     //situação valores de (s,S) ideal(final) com custo/desvio min e quebras<quebrasMax
149     double custoIdeal = 1000000;           //Numero Maximo,o mais longe do ideal
150     double desvioIdeal= 1000000;
151     int stockLevelIdeal = 0, stockRechargeIdeal = 0;
152     double artigosQuebraIdeal = 0, quebrasMediaIdeal = 0;
153
154     //Condições de simulação para n tries(simulações/Ano)
155     int[] [] procuras = new int[tries][duracao]; //cada linha é um ano, cada coluna uma semana
156     int[] [] prazosEntregas = new int[tries][duracao];
157
158     //Gerar sequencias de procuras e prazos entrega=>preencher arrays procura,prazosEntrega.
159     //geradas antecipadamente para garantir que diferentes (s,S) simulem as mesmas condições.
160
161     gerarSequencias(procuras,prazosEntregas,tries,duracao,media,desvio,probPrazo);
162
163     for (int stockLevel = stockLevelMin; stockLevel <= stockLevelMax;
164     stockLevel += stockLevelInt) {
165
166         for (int stockRecharge = stockRechargeMin; stockRecharge <= stockRechargeMax;
167         stockRecharge += stockRechargeInt) {
168             double custo = 0;
169             double artigosQuebra = 0;
170             double quebras = 0;
171             double[] custos =new double[tries]; //custos medios por semana numa simulação(ano)
172             for (int t = 0; t < tries; t++) {
173                 Otimizacao o = new Otimizacao(stockLevel, stockRecharge, stockLevel,
174                 duracao, c1, c2, c3, ciclo, procuras[t],prazosEntregas[t]);
175                 double custoControl= o.calculaCusto(debug)/duracao; //custo/ano
176                 custos[t]=(custoControl); //ano->semana
177                 custo+=custoControl; //custo+=Custo/ Ano
178                 artigosQuebra += o.getArtigosQuebra(); //artigosQuebra+=artigosQuebra num Ano
179                 quebras += o.getQuebras(); //quebras+=quebras num ano
180                 //System.out.println("try="+t+" custo="+custoControl);
181             }
182             double custoMedio=custo/tries; //Custos em t anos-> custo por ano->custo por semana
183             double quebrasMedias=quebras/tries; //quebras em t anos->quebras por ano
184             artigosQuebra=artigosQuebra/tries; //artigosQuebra medio por ano
185
186             //Calcular desvioMedio por semana
187             //custos[j] = custoPorSemana numa simulação=j(ano)
188             double desvioR=0;
189             for(int j=0;j<tries;j++){
190                 desvioR+=Math.abs(custoMedio-custos[j]); //somar desvios de cada simulação
191             }
192             desvioR=desvioR/tries; // desvio Medio de n simulações

```

```

193
194         //Se este (s,S) é melhor que o melhor (s,S) de todas as iterações anteriores.
195         //Se sim atualizar variaveisIdeias
196         if ((custoMedio+(0.3*desvioR))<=(custoIdeal+(0.3*desvioIdeal))
197             && quebrasMedias <= maxQuebras) {
198             custoIdeal = custoMedio;
199             stockLevelIdeal = stockLevel;
200             stockRechargeIdeal = stockRecharge;
201             artigosQuebraIdeal = artigosQuebra;
202             quebrasMediaIdeal = quebrasMedias;
203             desvioIdeal=desvioR;
204         }
205
206     }
207 }
208 System.out.println("\nEm "+tries+" simulações,em media e arredondado as unidades:");
209 System.out.println("Ideal: s=" + stockRechargeIdeal + ", S=" + stockLevelIdeal);
210 System.out.println("CustoTotal/Semana=(" + (int)(custoIdeal)+","+ (int)desvioIdeal+");");
211 System.out.println("Artigos Quebrados/Ano=" + (int) artigosQuebraIdeal
212 + " Quebras/Ano=" + quebrasMediaIdeal);
213 }
214
215 private static void gerarSequencias(int[] [] procura, int[] [] prazosEntregas,
216                                     int tries,int duracao,double media,double desvio,
217                                     int [] probPrazo) {
218     //para n "tries" simulações, simular valores para cada semana(duracao)
219     Random gaussian = new Random();
220     Random r = new Random();
221     int tempoEntrega ;
222     double procura;
223     for (int t1 = 0; t1 < tries; t1++) {
224         for (int d = 0; d < duracao; d++) {
225             procura = gaussian.nextGaussian() * desvio + media; //Distribuição normal
226             procura[t1][d] = (int) procura;
227
228             tempoEntrega = r.nextInt(100); //Distribuição aleatória
229             if (tempoEntrega < probPrazo[0]) tempoEntrega = 1;
230             else if (tempoEntrega<probPrazo[1]+probPrazo[0]) tempoEntrega = 2;
231             else tempoEntrega = 3;
232
233             prazosEntregas[t1][d] = tempoEntrega;
234         }
235     }
236 }
237 }

```
