YUICHI TOKUMOTO

SIMULAÇÃO E OTIMIZAÇÃO DO CONTROLE DE ESTOQUES EM PROCESSOS DE PRODUÇÃO CONTÍNUOS

São Paulo
2023

YUICHI TOKUMOTO

SIMULAÇÃO E OTIMIZAÇÃO DO CONTROLE DE ESTOQUES EM PROCESSOS DE PRODUÇÃO CONTÍNUOS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Diploma de Engenheiro de Produção.

Orientador:

Prof. Dr. Marco Aurélio de Mesquita

São Paulo
2023

À minha família e amigos por todo o incentivo e suporte durante esta jornada.

AGRADECIMENTOS

Aos meus pais, Shinichi e Tiemi, e à minha irmã, Yukari, por serem a base sólida que sustentou minha jornada acadêmica, pelo apoio incondicional, amor e crença em meu potencial, mesmo nos momentos mais desafiadores.

À minha companheira, Bruna, por seu amor, compreensão e paciência. Sua presença e incentivo constantes tornaram essa jornada mais significativa e mais leve.

Aos meus amigos, que compartilharam comigo os altos e baixos desta jornada, pelos momentos de diversão, apoio e companheirismo que fizeram toda a diferença.

Ao meu orientador, Prof. Dr. Marco Aurélio de Mesquita, cuja orientação e conhecimento foram fundamentais para o desenvolvimento deste trabalho. Sua dedicação, paciência e disposição em compartilhar conhecimento foram essenciais.

À Escola Politécnica e à CentraleSupélec, por oferecerem o ambiente acadêmico propício para o meu crescimento pessoal e profissional, bem como por todas as oportunidades de aprendizado proporcionadas ao longo da minha graduação.

Agradeço a todos que de alguma forma contribuíram para minha formação e jornada acadêmica. Suas influências e apoio foram cruciais para que eu chegasse até aqui.

“Pour ce qui est de l’avenir, il ne s’agit pas de le prévoir, mais de le rendre possible.”

-- Antoine de Saint-Exupéry

RESUMO

Este Trabalho de Formatura aborda o problema de calibração de parâmetros de controle de estoque no contexto do *Stochastic Economic Scheduling Problem* (SELSP), considerando uma linha de produção em lotes, com múltiplos produtos. Para abordar esse problema, desenvolveu-se um modelo de simulação-otimização em *Python*, com o qual foram testados quatro métodos de busca/otimização distintos (Busca aleatória, *Nelder-Mead*, *Genetic Algorithm* e *Ant Colony Optimization*). Para calibrar esses algoritmos, desenvolveu-se um plano de experimentos de calibração para cada método de busca. Em seguida, com os métodos calibrados, um experimento fatorial foi realizado para avaliar e comparar o desempenho dos algoritmos. Por meio de uma análise de variância dos resultados obtidos nos experimentos, notou-se que o algoritmo *Nelder-Mead* se mostrou o mais eficiente em termos de resultados obtidos (ex: custos e nível de serviço), propondo soluções de custos até 24% inferiores às dos outros métodos, além de apresentar maior facilidade de calibração dos seus hiperparâmetros. A abordagem adotada neste Trabalho, que resultou na elaboração de um modelo de simulação-otimização em *Python* usando o método de otimização *Nelder-Mead*, apresenta benefícios em relações a soluções comerciais existentes, já que estas, muitas vezes, operam como "caixas pretas" e não são ferramentas específicas para este problema. Dessa forma, o presente Trabalho permitiu uma maior transparência e controle sobre o processo de otimização dos parâmetros de controle de estoque. Além disso, a implementação do modelo de simulação-otimização representa uma contribuição significativa para a otimização de parâmetros de estoque em problemas complexos de produção, como o SELSP. Essa abordagem flexível e acessível não apenas oferece uma ferramenta de código aberto para resolver esse desafio, mas também busca oferecer maior acessibilidade e independência de soluções comerciais. Com isso, o trabalho traz contribuições para a comunidade acadêmica e incentivos à indústria na busca por soluções eficazes no controle de estoques, já que o cenário do problema estudado espelha muitas indústrias, como as áreas química, cosmética e têxtil, evidenciando a relevância prática da pesquisa para a indústria.

Palavras-Chave – Controle de estoque, Simulação-otimização, SELSP, Indústria de processos, Processos contínuos

ABSTRACT

This work addresses the stock control parameter calibration problem in the context of the *Stochastic Economic Scheduling Problem* (SELSP), considering a batch production line with multiple products. To address this issue, a simulation-optimization model was developed using *Python*, and four distinct search/optimization methods (*Random Search*, *Nelder-Mead*, *Genetic Algorithm*, and *Ant Colony Optimization*) were tested with this model. To calibrate these algorithms, a calibration experiment plan was developed for each search method. Subsequently, with the calibrated methods, a factorial experiment was conducted to assess and compare the performance of the algorithms. Through an analysis of variance of the results obtained in the experiments, it was observed that the *Nelder-Mead* algorithm proved to be the most efficient in terms of results (e.g., costs and service level), proposing cost solutions up to 24% lower than those of other methods, in addition to showing greater ease in calibrating its hyperparameters. The approach adopted in this work, which resulted in the development of a simulation-optimization model in *Python* using the *Nelder-Mead* optimization method, offers benefits compared to existing commercial solutions, which often operate as "black boxes" and are not specific tools for this problem. Thus, this work allowed for greater transparency and control over the optimization process of stock control parameters. Furthermore, the implementation of the simulation-optimization model represents a significant contribution to the optimization of stock parameters in complex production problems, such as SELSP. This flexible and accessible approach not only provides an open-source tool to solve this challenge but also seeks to offer greater accessibility and independence from commercial solutions. Therefore, this work contributes to the academic community and encourages the industry to seek effective solutions in stock control, as the studied problem scenario mirrors many industries, including the chemical, cosmetic, and textile sectors, highlighting the practical relevance of the research for the industry

Keywords – Inventory control, Simulation-optimization, SELSP, Process industry, Continuous processes

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo, apresenta-se uma contextualização do trabalho, seguida da definição do problema e dos objetivos principais. A seguir, apresentam-se a justificativa para a realização deste trabalho e a sua estrutura.

1.1 Contexto

O controle de estoques é parte essencial para o gerenciamento de diversos negócios. O correto dimensionamento e monitoramento dos níveis de estoque em uma empresa pode afetar diretamente tanto os resultados operacionais quanto os financeiros da organização. Uma tomada de decisão assertiva em relação a políticas de reposição e tamanhos dos lotes pode garantir à empresa elevados níveis de serviço, sem onerar suas finanças de forma desnecessária.

Por outro lado, ineficiências no controle de lotes podem gerar atrasos ou perdas de pedidos e consequente insatisfação de seus clientes, resultando em perda de receita e/ou aumento de custos da operação (i.e. custo de manutenção de estoque, custo de venda perdida, custo de oportunidade, etc).

Por isso, pesquisadores e empresas têm dado grande atenção ao estudo de problemas de controle de estoque, o que tem motivado a produção de estudos acadêmicos e o desenvolvimento de *softwares* voltados para soluções industriais.

Nesse sentido, a calibração dos parâmetros de controle de estoque tem se tornado um importante processo na gestão das organizações. Essa prática visa a ajustar e otimizar os parâmetros que definem os pontos de reposição, as políticas de reabastecimento e a frequência de pedidos de forma a atingir os níveis ideais de estoque médio, equilibrando a disponibilidade de produtos com os custos associados ao seu armazenamento.

Uma possível abordagem para a calibração do controle de estoques é a utilização de métodos de simulação e otimização, principalmente nas situações em que não há uma expressão analítica satisfatória para a função objetivo. Por meio dessa metodologia, que relaciona as variáveis de decisão (parâmetros de controle de estoque) e a função objetivo (custo total de estoque), é possível simular diferentes cenários e verificar, para cada configuração de parâmetros de controle, o impacto no sistema. Com o auxílio do modelo de otimização, buscam-se melho-

res soluções com base nos resultados obtidos anteriormente.

Nesse contexto, o presente estudo abordará a calibração dos parâmetros de controle de estoques de produtos acabados, em uma linha de produção contínua, utilizando a abordagem de simulação-otimização para determinar os valores desses parâmetros de forma a balancear objetivos operacionais e financeiros de um sistema de produção.

1.2 Formulação do problema

O presente Trabalho de Formatura aborda o problema de controle de estoque conhecido na literatura por *Stochastic Economic Lot Scheduling Problem* (SELSP). O problema consiste na programação de uma máquina capaz de produzir N produtos, mas apenas um tipo de cada vez (i.e. uma máquina com produção em lotes). Essa configuração exige que haja uma decisão a ser tomada toda vez que é concluído um lote de produção, sendo preciso decidir qual será o próximo SKU (*Stock Keeping Unit*) a entrar em produção e qual a quantidade a produzir.

Embora, tradicionalmente, o SELSP consista na determinação de uma sequência de produção e no dimensionamento dos lotes (potencialmente fixos) para a minimização do custo médio total, o presente trabalho aborda esse problema de forma diferente. Tendo em vista os modelos clássicos de ponto de pedido, o problema abordado por esse Trabalho de Formatura passa a ser a determinação dos parâmetros de controle de estoque que minimizam o custo médio total. Essa abordagem proposta, embora mais simples, torna mais viável a aplicação do problema em casos reais.

Durante a produção, primeiramente, verificam-se quais são os SKU's que estão abaixo de seus respectivos pontos de reabastecimento, s_i. Em seguida um produto é escolhido para ser produzido com base em seu tempo de cobertura (i.e. Nível de estoque atual $\div$ demanda esperada), sendo os produtos com menor cobertura os mais prioritários (*Lowest Days of Supply* - LDS). O tamanho do lote do produto é definido pelo nível de estoque máximo, S_i, do produto e seu nível de estoque atual. Note que caso ocorram pedidos durante a produção do lote, o nível de estoque máximo não será atingido após a finalização do lote. Ainda, destaca-se que a produção de um lote não pode ser interrompida após seu início e caso não existam itens abaixo do ponto de abastecimento, a produção fica ociosa.

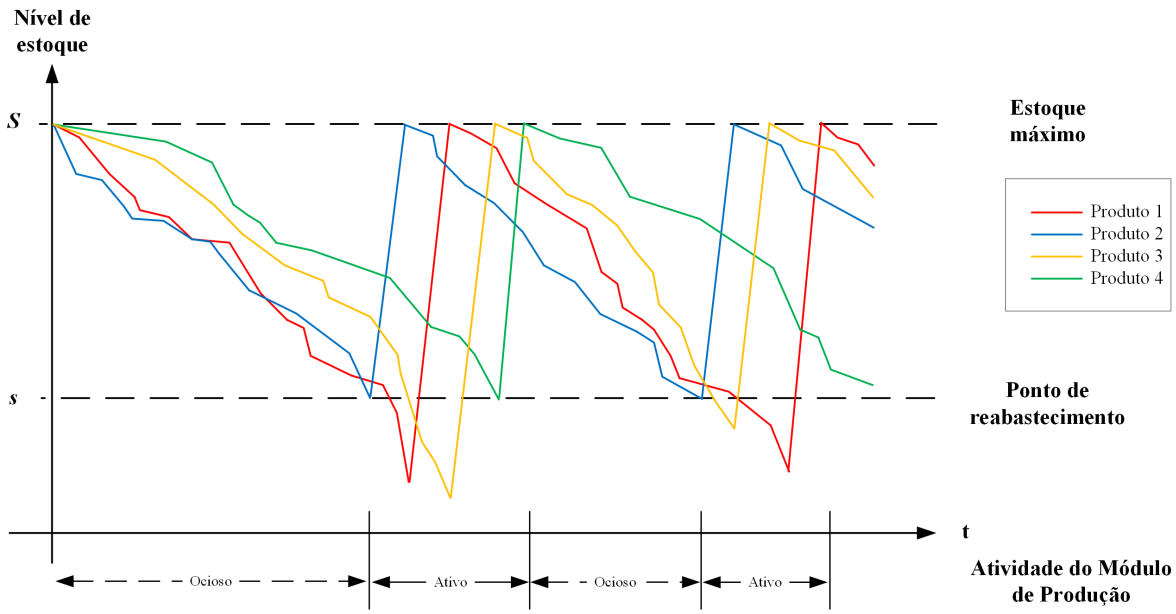
Sendo assim, o problema a ser resolvido é a determinação dos valores dos parâmetros de estoque máximo, S_i, e o ponto de reabastecimento, s_i, para cada um dos N produtos manufaturados na fábrica (i.e. $(s_i, S_i)\ \forall\ i = 1, 2, ..., N$) para minimizar o custo médio total de estoque. A solução desse problema não pode ser encontrada de forma analítica e será abordada pelo autor

por meio de um método de simulação-otimização.

Para sistemas de produção com múltiplos produtos cujas demandas são estacionárias, a determinação dos parâmetros de estoque torna-se complexa (Figura 1), devendo levar em consideração diversos outros fatores como: tempos de *setup*, número de produtos, políticas de priorização de produtos, entre outros. Pela Figura 1, é possível perceber que um dimensionamento equivocado dos parâmetros de controle de estoque pode levar à falta de estoque, mesmo que a taxa de utilização do setor de produção esteja abaixo de seu nível máximo.

Em geral, a calibração dos parâmetros de estoque na indústria é feita por meio de *softwares* desde os mais simples, como o *Excel* (BARRY; JAY; CHUCK, 2017), até *softwares* mais robustos comercializados por diferentes empresas, tais como o *OptQuest*, comercializado pela *OptTek System Inc*. Contudo, as diferentes alternativas restringem a exploração do presente problema estudado devido a limitações dos próprios *softwares* em termos de ferramentas/recursos, à falta de clareza sobre o método de otimização utilizado, a limitações de integrações entre diferentes soluções (ex: *softwares* diferentes para simulação e otimização) e, de forma mais prática, à necessidade de se adquirir licenças para utilizá-los.

Figura 1: Exemplo de monitoramento do nível de estoque de quatro produtos com parâmetros de controle (s, S) para uma demanda estocástica.



Fonte: Elaborado pelo autor.

1.3 Objetivos do trabalho

O objetivo deste Trabalho de Formatura consiste na identificação e implementação de métodos numéricos eficientes para a otimização dos parâmetros de controle de estoque $(s_i, S_i) \; \forall \; i = 1, 2, ..., N$, no contexto do problema descrito na seção 1.2.

Além disso, o trabalho busca oferecer uma solução de implementação em um *software* livre e de código aberto, sem a dependência de *softwares* pagos. Assim, a solução proposta almeja a implementação de um modelo de simulação junto a um método de busca de soluções.

Dessa forma, o modelo de simulação-otimização proposto neste estudo será implementado em *Python*, por meio do qual será possível propor uma solução em código aberto para avaliar os métodos de otimização e busca a partir de indicadores de desempenho dos algoritmos (ex: convergência e tempo de execução) e indicadores desempenho do próprio problema (ex: custos e nível de serviço).

1.4 Relevância

O presente Trabalho de Formatura traz contribuições para o estudo de controle de estoque e pode ser adaptado e aplicado de forma prática na indústria. O problema apresentado na seção 1.2, com múltiplos produtos e uma única linha de produção em lotes é semelhante à realidade enfrentada pela indústria de processos. Nesse tipo de indústria, bens de consumo ou produtos intermediários são produzidos por meio da transformação da matéria-prima via uma série de processos físicos, químicos e biológicos, tipicamente com produções em *batch*. Alguns exemplos de produtos comumente produzidos por meio deste método são:

1. Produtos químicos: resinas plásticas e revestimentos;
2. Produtos alimentícios: produtos de panificação, molhos, condimentos, bebidas alcoólicas e não alcoólicas, etc;
3. Produtos farmacêuticos: medicamentos;
4. Produtos cosméticos: loções, cremes e outros produtos de beleza;
5. Tintas e revestimentos: tintas e revestimentos para fins decorativos e industriais;
6. Produtos têxteis: camisetas, calças, vestidos, uniformes esportivos, etc.

Devido à grande quantidade de *setups* dos equipamentos para a troca de produtos em linhas de produção como essas, percebe-se que há um problema inerente a esse modo de produção relacionado à otimização do dimensionamento dos lotes de produção e ao sequenciamento dos produtos nas máquinas. Dessa forma, diversas contramedidas são implementadas pelas empresas de forma a minimizar custos e o tempo perdido com *setup* dos equipamentos e maximizar o nível de serviço, assim como evidenciado no trabalho de Tomotani e Mesquita (2017), que realizaram um levantamento das práticas na indústria.

Sobre a implementação em *Python*, em código aberto, do modelo de otimização-simulação proposto, apresenta-se um modelo customizado para a resolução do SELSP, eliminando a dependência de *softwares* genéricos e pagos já existentes no mercado, tais como o *AnyLogic* (simulação) e o *OptQuest* (otimização). Esses *softwares* dificultam o acesso a soluções computacionais para o problema estudado e limitam a usabilidade de ferramentas para aprimorar a busca de soluções para o problema.

Dessa forma, o estudo em questão contribui tanto para a melhoria dos métodos de controle de estoque quanto para a disponibilização de soluções computacionais para o problema abordado.

1.5 Estrutura do trabalho

Este Trabalho de Formatura está organizado nos seguintes capítulos:

1. *Introdução*: Define o problema, os objetivos e a relevância do estudo;
2. *Fundamentação teórica*: Traz uma revisão bibliográfica abordando o controle de estoque, modelos de simulação-otimização e métodos numéricos de otimização aplicáveis ao problema de *Stochastic Economic Lot Scheduling Problem* (SELSP);
3. *Método*: Apresenta o método de solução empregado, as lógicas dos modelos de simulação e otimização para a resolução do problema estudado;
4. *Modelo de Simulação*: Detalha os modelos conceitual e computacional de simulação e apresenta a validação e verificação desse modelo;
5. *Modelo de Simulação-Otimização*: Apresenta em detalhes os métodos de busca implementados, seus experimentos de calibração e suas implementações junto ao modelo de simulação;

6. *Planejamento dos Experimentos de Comparação*: Traça o plano dos experimentos de comparação realizados, identificando os parâmetros a serem variados e as demais variáveis do problema;

7. *Discussão dos Resultados*: Discute os resultados obtidos com os experimentos de comparação realizados;

8. *Conclusões*: Este último capítulo traz um resumo dos principais resultados, as limitações do estudo e discute perspectivas futuras de possíveis extensões para o estudo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, apresenta-se uma revisão bibliográfica, na qual foram explorados o tema de controle de estoque de forma mais geral e, de forma mais profunda, os problemas de estoque para múltiplos produtos. Além disso, foram levantados os principais métodos de busca que podem ser aplicados ao problema estudado.

2.1 Controle de estoque

Nesta seção, explora-se os principais conceitos relacionados ao controle de estoque, abordando questões de custo e políticas de controle, que são essenciais para a construção do modelo de simulação proposto no Trabalho.

2.1.1 A importância do controle de estoques

De forma geral, a maioria dos setores econômicos lida com questões ligadas à gestão de cadeias de suprimentos, de forma que o correto controle do fluxo de material é um fator importante para a manutenção de um bom relacionamento entre as empresas, seus fornecedores e seus clientes. Sendo assim, o controle de estoque é parte relevante para as estratégias das empresas, pois além de influenciar diretamente *stakeholders* externos à empresa, também tem forte impacto em diversos setores internos.

É comum que se tenha a visão de que os sistemas de controle de estoques tenham como principal objetivo a gestão de conflitos de interesse em uma fábrica. Se, por um lado, o setor de finanças tende a preferir que a fábrica opere com estoques mínimos para que mais capital esteja disponível para outros investimentos, o setor comercial, em sentido contrário, prefere operações com grandes estoques para garantir entregas de pedidos de forma rápida e completa. Ao mesmo tempo, o gerente de produção dará preferência à produção de grandes lotes para evitar custos e paradas para a realização de *setups* nas máquinas, enquanto o setor comercial prefere estoques com lotes menores para ter níveis de estoque equilibrados entre os produtos de forma a aumentar o nível de serviço da empresa (AXSÄTER, 2015).

Em meio a esses conflitos e dada a importância estratégica do assunto, as empresas rea-

lizam grandes investimentos no setor de estoques, buscando a implementação de ferramentas de controle mais eficientes, de forma a reduzir seus custos e manter ou melhorar seu nível de serviço (BARRY; JAY; CHUCK, 2017).

Além disso, a complexidade do assunto têm aumentado nas últimas décadas, atraindo pesquisadores e empresas a criarem métodos modernos de controle de estoques por meio de modelos de decisão complexos que aliam a experiência e a tecnologia, deixando de lado a aplicação de regras de decisão simples usados há décadas (AXSÄTER, 2015).

2.1.2 Custos de Estoque

Uma abordagem comum do controle de estoques é trabalhar com a minimização do custo total. Nessa abordagem, diversos custos são considerados, mesmo que alguns não se materializem em fluxos de caixa.

A seguir, serão introduzidos os principais custos ligados ao problema de estoques, tais como o custo de manutenção do estoque (*holding cost*), custo de *setup* (*setup cost*) e custo de falta (*shortage cost*).

2.1.2.1 Custo de manutenção do estoque (*Holding cost*)

A ideia do *holding cost* está ligada, principalmente, ao custo de oportunidade de outros investimentos, mas também considera outros fatores como custos de manuseio de cargas, de armazenamento, de danos ao produto, de obsolescência e eventuais taxas. Em suma, todos os custos variáveis devem ser englobados por essa categoria.

A posse de grandes estoques por uma empresa incorre em diferentes custos que oneram a parte financeira da empresa e podem (ou não) aumentar a complexidade operacional, exigindo maiores espaços, mais operadores e mais equipamentos, por exemplo. Além disso, o capital investido para garantir a posse desse estoque poderia estar alocado em outros ativos, seja investimentos visando a um maior retorno financeiro ou mesmo ativos de maior liquidez para a empresa (ex: caixa) (AXSÄTER, 2015).

Em geral, o custo de manutenção é calculado de forma unitária (por produto) e dentro de espaço de tempo. Ainda, de forma a evitar o rateamento dos custos operacionais e simplificar o cálculo, esse valor é comumente relacionado a uma porcentagem do custo do material armazenado.

2.1.2.2 Custo de *setup* (*Setup cost*)

O reabastecimento dos estoques de produtos acabados ou de matéria-prima costuma estar associado a um custo fixo, independente do tamanho do lote. A esse custo é dado o nome de *setup cost*, que pode levar em conta a compra de materiais ou a produção de itens.

Para esse custo são considerados fatores como custos de preparo e treinamento, custos administrativos para gestão de pedidos, custos associados a transporte e manuseio de carga, tempo de *setup*, entre outros (AXSÄTER, 2015).

2.1.2.3 Custo de falta (*Shortage cost*)

Caso uma demanda não seja atendida devido a estoque insuficiente, o cliente tem duas possíveis alternativas: deixar seu pedido como pendente e receber em completude assim que possível ou cancelar o pedido e escolher outro fornecedor. Em ambos os casos, diversos custos são incorridos pela empresa fornecedora.

Um pedido pendente exigirá, em geral, uma mobilização adicional da empresa fornecedora que pode ser traduzida em descontos ao cliente, fretes mais rápidos e mais caros, mão de obra adicional (na operação e no administrativo) etc. Além disso, embora seja difícil mensurar, há um desgaste da relação com o cliente, que pode ser prejudicial para os negócios da empresa no longo termo. Evidentemente, caso o cliente opte por um outro fornecedor, a empresa deixa de receber a remuneração pela venda do produto e o desgaste com o cliente é ainda maior (AXSÄTER, 2015).

Para o setor de produção, a falta de materiais e componentes pode gerar ociosidade de linhas de produção e atrasos, sendo necessário reorganizar as ordens de produção.

A dificuldade de precisar o custo associado à ruptura de estoques faz com que seja comum a associação desse custo a níveis mínimos de serviço da fábrica.

2.1.3 Políticas de revisão de estoque

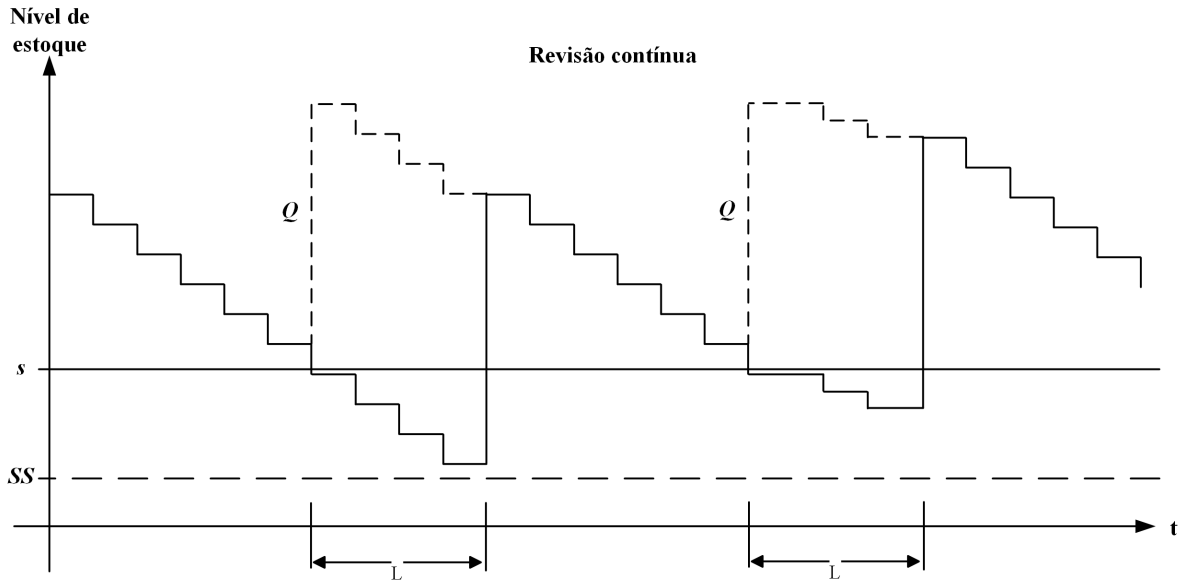
Os modelos de controle de estoque podem ser classificados em duas grandes classes: revisão contínua e periódica.

O modelo de revisão contínua estabelece uma política de acompanhamento contínuo da posição do estoque. Assim, logo que o nível de estoque atinge o nível de reposição, s, um pedido de tamanho fixo Q é feito, conforme representado na Figura 2 (AXSÄTER, 2015).

Note que o o ponto de reabastecimento, deve levar em consideração fatores como o *lead time* de entrega do fornecedor, L, e outros fatores ligados à variação da demanda.

Esse modelo de revisão apresenta como vantagem a possibilidade de trabalhar com estoque de segurança, SS, reduzido quando comparado a modelos de revisão periódica (AXSÄTER, 2015).

Figura 2: Representação da política de revisão contínua.



Fonte: Elaborado pelo autor.

Ainda, além da política de revisão de estoques contínua, podem ser estabelecidas políticas de reabastecimento de estoques, tais como a política (R, Q) e a (s, S).

A política (R, Q) estabelece um ponto de reabastecimento R e um tamanho de lote Q, de forma que assim que o estoque atinge uma posição R, um lote de tamanho Q é produzido.

Note que em um sistema com reabastecimento (R, Q), em geral, o pedido pelo material ocorrerá quando a posição do estoque já está abaixo do nível R, fazendo com que o estoque não atinja o valor de $R + Q$ (AXSÄTER, 2015).

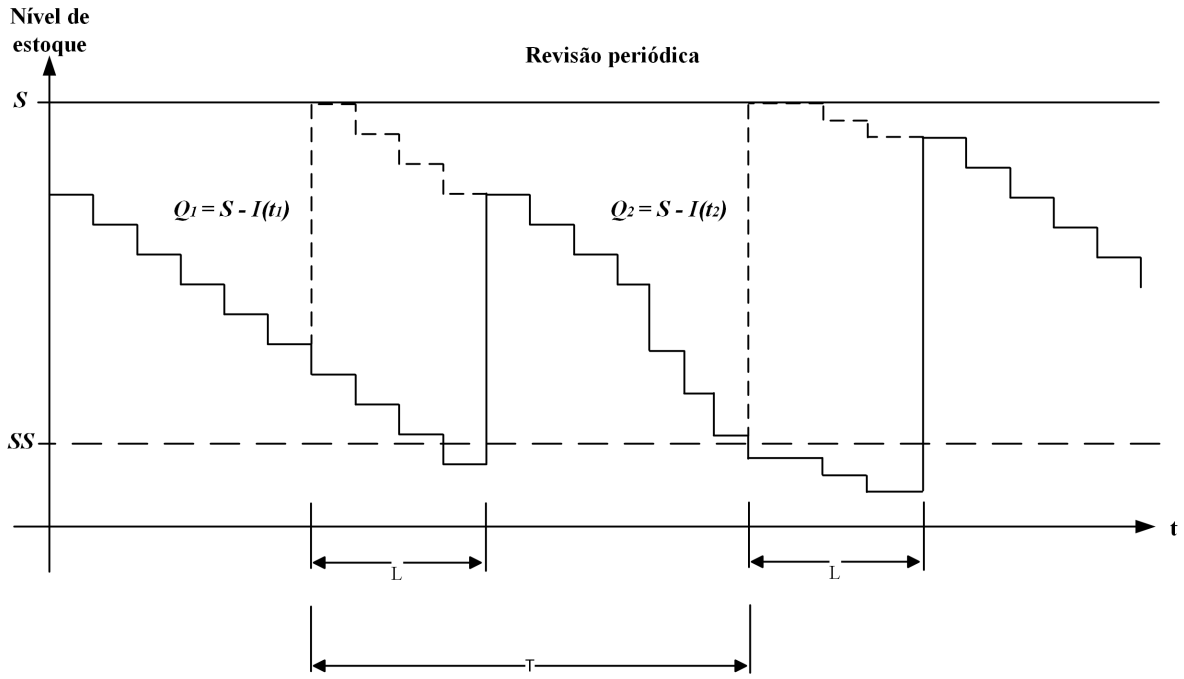
A política de reabastecimento (s, S) é similar à política (R, Q) e também estabelece um ponto de reabastecimento, aqui denominado como s, que ativa o pedido por um novo lote. Porém, nesse modelo, o lote pedido tem tamanho variável e é determinado pela quantidade necessária para que se atinja o nível de estoque máximo S (*order-up-to*) (AXSÄTER, 2015).

Dando continuidade às políticas de revisão de estoque, a outra grande classe é o modelo de revisão periódica. Esse modelo estabelece uma política de acompanhamento com verificações

periódicas da posição do estoque em intervalos fixos de tempo T. Assim, a cada período T, o nível de estoque é verificado e um pedido de tamanho variado Q_i é feito, de forma a restabelecer o estoque até seu nível máximo S, conforme ilustrado na Figura 3 (AXSÄTER, 2015).

Nesse modelo de revisão é preciso estabelecer níveis de estoques de segurança maiores do que no modelo de revisão periódica. Porém, uma das principais vantagens desse modelo de revisão é a escalabilidade do sistema de controle de estoque, uma vez que ele reduz de forma considerável o custo com inspeções da posição do estoque, o que permite o acompanhamento de múltiplos itens de forma mais prática (AXSÄTER, 2015).

Figura 3: Representação da política de revisão periódica.



Fonte: Elaborado pelo autor.

2.2 Problemas de dimensionamento de lotes

Esta seção dedica-se à definição e análise de problemas de estoque, além de levantar os principais métodos que têm sido utilizados para a resolução do problema de *Stochastic Economic Lot Scheduling Problem* (SELSP).

2.2.1 Problema clássico do Lote econômico de compra

O modelo mais antigo (e simples), ligado ao problema de controle de estoque é o problema do Lote Econômico de Compra, também conhecido como *Economic Order Quantity* (EOQ). Esse modelo foi aplicado inicialmente em meados de 1913, por Ford W. Harris, para o dimensionamento de lotes de produção (HOPP; SPEARMAN, 2011).

O modelo proposto traz diversas simplificações, que muitas vezes não são aplicáveis de forma prática, mas que permitem que seja possível descrever de forma analítica o tamanho do lote de compra de um produto.

Para o modelo EOQ, assume-se que (HOPP; SPEARMAN, 2011):

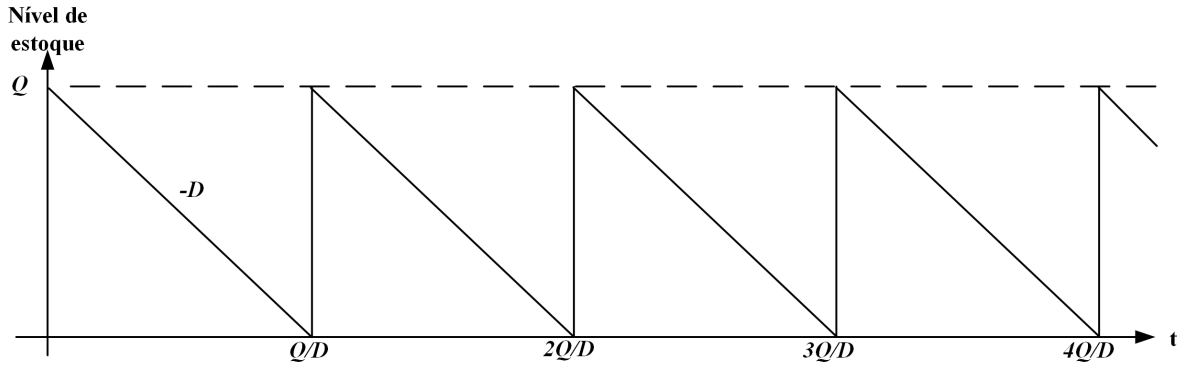
1. A produção ocorre de forma instantânea e não há limitações de capacidade para a produção do lote completo;
2. A entrega do pedido ocorre de forma imediata, não havendo intervalo de tempo entre a produção e a entrega do produto para satisfazer a demanda;
3. A demanda é determinística, desprezando quaisquer incertezas em relação aos tamanhos dos pedidos e seu intervalo de ocorrência;
4. A demanda é constante durante todo o período;
5. O início da produção incorre em um custo fixo de *setup*, independentemente do tamanho do lote (de produção);
6. Os produtos são analisados de forma independente, assim, deve-se considerar a existência de apenas um produto ou assume-se que não há compartilhamento de recursos, como máquinas e operadores, entre os produtos.

Para computar o tamanho ótimo do lote, Q^*, é necessário o conhecimento das seguintes variáveis do problema:

$$D = \text{A taxa de demanda (un./período)}$$
$$c = \text{O custo unitário de produção (R\$/un.)}$$
$$A = \text{O custo fixo de pedido/}\textit{setup}\text{ do lote (R\$)}$$
$$h = \text{O custo unitário de manutenção do estoque}$$

Assim, diante das simplificações mencionadas e adotando a notação indicada acima, pode-se traçar a evolução do nível de estoque conforme mostrado na Figura 4.

Figura 4: Evolução do nível de estoque para o modelo EOQ.



Fonte: Adaptado de Hopp e Spearman (2011).

A linearidade e a natureza determinada do problema permitem descrever de forma analítica a função de custo para o problema. Assim, para um dado período, pode-se deduzir o custo total, Y, da seguinte forma (HOPP; SPEARMAN, 2011):

$$Y(Q) = \underbrace{\frac{hQ}{2}}_{\text{Custo de manutenção do estoque}} + \underbrace{\frac{AD}{Q}}_{\text{Custo de } \textit{setup}} + \underbrace{cD}_{\text{Custo de produção}} \tag{2.1}$$

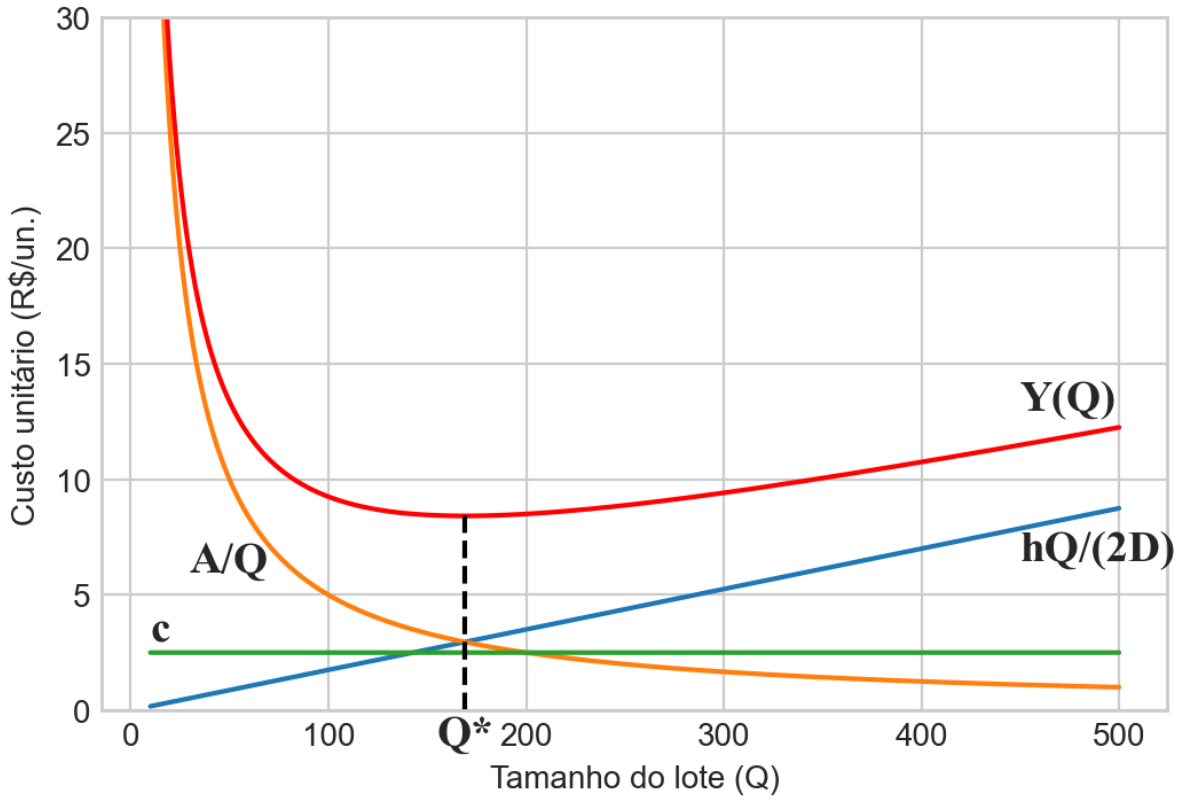
A equação 2.1 permite traçar as curvas para o custo total de estoque, assim como para cada componente do custo, como mostrado na Figura 5. Pelo gráfico, é possível notar que há um *trade-off* entre o custo de *setup* e o custo de manutenção do estoque e, conforme aumenta-se o tamanho do lote, o custo de manutenção fica cada vez mais predominante no custo unitário de estoque do produto.

Além disso, por meio da equação 2.1, é possível derivar a clássica fórmula para o tamanho ideal do lote, Q^*, ao minimizar-se o custo total de estoque:

$$Q^* = \sqrt{\frac{2AD}{h}} \tag{2.2}$$

Duas premissas importantes limitam o modelo proposto por Harris: 1) a existência de um único produto e 2) a demanda constante. Essas hipóteses simplificadores, em muitos casos, comprometem a aplicação da fórmula 2.2 no contexto real das indústrias.

Figura 5: Exemplo de curvas de custos unitários para o modelo EOQ.



Fonte: Adaptado de Hopp e Spearman (2011).

Todavia esse clássico modelo trouxe importantes noções sobre o *trade-off* entre o custo de *setup* e o custo de manutenção de estoque, assim como o *trade-off* entre o tamanho de lote de produção (ou compra) e o tamanho do inventário (HOPP; SPEARMAN, 2011).

Nesse contexto, os avanços no estudo de controle de estoques permitiram o surgimento de novos modelos que permitem o relaxamento de algumas das premissas adotadas pelo modelo EOQ, como será visto a seguir.

2.2.2 Re-Order Point (ROP) model

Dando continuidade ao problema estabelecido pelo modelo EOQ, o modelo ROP também considera o controle de um único item, porém, alternativamente, relaxa-se a premissa de que a demanda do produto é constante.

Assim, para o modelo ROP, são assumidas as seguintes premissas (NAHMIAS; OLSEN, 2015):

1. A demanda é aleatória e estacionária, ou seja, embora não seja possível determinar a

demanda exata para cada período, é possível ter conhecimento da esperança do valor da demanda para certo período de tempo;

2. Rupturas de estoque são permitidas, podendo haver perda de venda ou atraso no pedido;

3. Considera-se um único item e assume-se que as demais premissas são compatíveis com o modelo EOQ.

A preocupação do modelo passa então a ser a determinação de duas variáveis de decisão independentes: a do estoque mínimo para a realização do pedido do lote para que as incertezas da demanda sejam absorvidas; e o tamanho do lote a ser produzido/pedido (HOPP; SPEARMAN, 2011).

Adicionalmente, é possível segmentar o modelo ROP em outros 2 modelos (HOPP; SPEARMAN, 2011):

- *Base stock model*: Nesse modelo, o estoque é reabastecido com uma unidade por vez, conforme a demanda ocorre, e o problema a ser resolvido é a determinação do ponto de reabastecimento.

- *(R, Q) model*: Nesse modelo, considera-se uma política de acompanhamento de estoque contínuo, de forma que quando o estoque atinge um nível R, um lote de tamanho Q é solicitado. Em seguida, após certo período de tempo l, durante o qual rupturas de estoque podem ocorrer, o lote é recebido. Assim, o problema a ser solucionado é a determinação dos parâmetros R e Q.

2.2.3 Economic Lot Scheduling Problem (ELSP)

Talvez uma das simplificações mais limitantes do modelo EOQ seja considerar apenas um tipo de produto ou, caso existam múltiplos produtos, cada SKU pode ser analisado de forma independente, uma vez que não compartilham recursos na fábrica (AXSÄTER, 2015).

Extensamente estudado na literatura, o *Economic Lot Scheduling Problem* (ELSP) relaxa essa premissa e busca determinar programações cíclicas para sistemas de produção com múltiplos itens de demandas constantes, que compartilham um mesmo recurso.

Nesse problema, busca-se não apenas determinar quanto deve ser produzido de cada produto, mas também quando cada um dos itens deve ter sua produção iniciada.

Sendo assim, no ELSP, tem-se o interesse em programar a produção de múltiplos produtos em uma única máquina de forma a minimizar os custos de estoque, tais como os custos de *setup* e de manutenção do estoque.

Assim como feito para o modelo EOQ, listam-se abaixo as principais premissas do ELSP (LARRAÑETA; ONIEVA, 1988):

1. Há apenas um recurso disponível;
2. Um único produto pode ser produzido por vez;
3. Os custos e tempos de *setup* são constantes e podem ser particulares para cada produto;
4. As taxas de produção são conhecidas e constantes, podendo ser particulares para cada produto;
5. As demandas são conhecidas e constantes, podendo ser particulares para cada produto.

Dessa forma, tendo em vista as hipóteses acima e assumindo as restrições de que não há ruptura de estoque e de que toda a demanda é atendida, assume-se que o tamanho das *batches* para cada produto e os tempos de ciclos são mantidos constantes. Além disso, esses fatores dependem dos parâmetros de estoque e das políticas de acompanhamento de estoque adotados pela fábrica.

Ainda, ressalta-se o fato de que o *Economic Lot Scheduling Problem* pode ser modelado com programação matemática e é conhecido como um problema *NP-hard* (CHUNG; CHAN, 2012), ou seja, ainda não é possível encontrar um solução ótima para o problema em tempo polinomial. Essa dificuldade impõe limites computacionais para a resolução do problema, levando à busca de soluções aproximadas por meio da utilização de heurísticas.

2.2.4 Stochastic Economic Lot Scheduling Problem (SELSP)

Naturalmente, o *Stochastic Economic Lot Scheduling Problem* (SELSP) é uma variação do ELSP, no qual incertezas são adicionadas ao problema por meio de fatores estocásticos que podem estar relacionados às demandas, tempos de ciclo, tempos de *setup* ou alguma combinação desses fatores.

Dessa vez, o interesse é de calibrar os valores dos parâmetros de estoque e a programação dos N produtos que compartilham a utilização da única máquina disponível.

A incerteza adicionada ao problema em relação à demanda e aos tempos de processamento aumenta significativamente a complexidade da programação matemática, uma vez que a determinação da quantidade a ser produzida por cada produto e a sequência de produção dos itens deixam de ser fixas. Sendo assim, é necessário estabelecer políticas de estoque para definir priorizações e quantidades a serem produzidas para garantir níveis de serviço e custos satisfatórios (AXSÄTER, 2015).

Embora esse dinamismo implique em um aumento de complexidade, o problema representado pelo SELSP é muito próximo da realidade enfrentada por várias empresas, principalmente para a indústria de processos. Ainda, a resolução do SELSP depende da formulação proposta ao problema, que pode ser orientada à definição de sequenciamentos de produtos ou à calibração de parâmetros de estoque.

No presente estudo, adota-se a segunda abordagem, usando a ideia de ponto de pedido para reposição de estoque e voltando esforços para a calibração dos parâmetros de estoque (s, S).

De toda forma, a busca por métodos eficientes que atinjam bons resultados (i.e. métodos que escolham bons valores para s e S, por exemplo) para a resolução do SELSP é importante para integrar os estudos da literatura com os problemas reais de controle de estoques enfrentados pelas empresas. Esses esforços permitem que as empresas possam tomar decisões informadas/embasadas sobre sua produção e políticas de estoque.

2.3 Algoritmos de busca

Nesta seção, apresentam-se alguns algoritmos de busca de mínimos que podem ser usados na busca de soluções para otimização dos parâmetros do problema definido no capítulo 1.

2.3.1 Busca exaustiva

O método de busca exaustiva, também conhecida no literatura como método de "força bruta", é uma das abordagens mais simples e genéricas para a busca de soluções em problemas de otimização (HAUPT; HAUPT, 2004). Note, porém, que esse método é aplicável em problemas cujo espaço de soluções é enumerável. Dessa forma, para problemas em espaços contínuos, é preciso discretizar o espaço criando malhas discretas com intervalos suficientemente pequenos.

Nesse método, todos os possíveis candidatos do espaço de soluções são testados e avaliados para determinar-se a melhor solução, como ilustrado pelo Algoritmo 1.

Algoritmo 1 Busca exaustiva

1: **função** BUSCA EXAUSTIVA(S) ▷ Onde S é o conjunto de todas as soluções candidatas
2: $s_{best} \leftarrow None$ ▷ Melhor solução
3: $f_{best} \leftarrow \infty$ ▷ Custo da melhor solução
4: **Para** cada s em S **faça**
5: $f_s \leftarrow$ AVALIAR(s) ▷ Calcula o custo para o candidato s
6: **Se** $f_s < f_{best}$ **Então**
7: $f_{best} \leftarrow f_s$
8: $s_{best} \leftarrow s$
9: **Fim Se**
10: **Fim Para**
11: **Imprima** s_{best}, f_{best}
12: **Fim função**

Embora seja facilmente implementada e garanta a identificação da solução-ótima, caso exista, a busca exaustiva é custosa em tempo e poder computacional. Note que para problemas complexos o número de soluções candidatas cresce exponencialmente conforme o tamanho do problema aumenta, tornando a busca por força bruta difícil de ser praticada.

Assim, seja N_{var} o número de variáveis e Q_i os possíveis valores assumidos pela variável i, o número de combinações possíveis, N_{comb}, a serem avaliadas é dado por (HAUPT; HAUPT, 2004):

$$N_{comb} = \prod_{i=1}^{N_{var}} Q_i \tag{2.3}$$

Note ainda que em alguns casos é possível estabelecer um subconjunto do espaço de soluções, onde acredita-se que a solução ótima possa ser encontrada, de forma a diminuir o número de soluções avaliadas. Porém, essa estratégia não garante que a solução encontrada seja um mínimo global, sendo, portanto, um possível mínimo local.

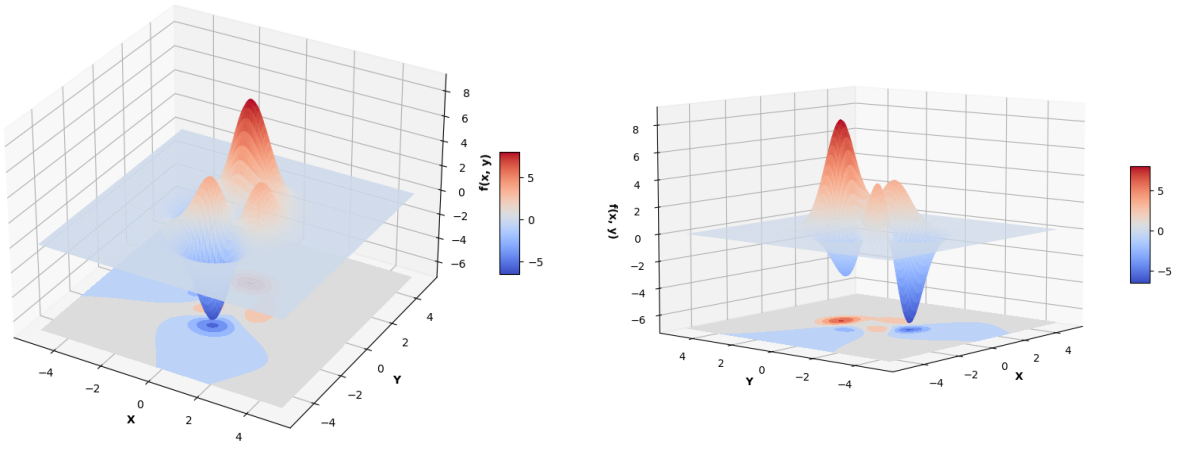
Para exemplificar esse fato, toma-se o exemplo da função seguinte função $f : (\mathbb{R}, \mathbb{R}) \rightarrow \mathbb{R}$:

$$f(x,y) = 3 \cdot (1-x)^2 \cdot e^{-x^2-(y+1)^2} - 10 \cdot (\frac{x}{5} - x^3 - y^5) \cdot e^{-x^2-y^2} - \frac{1}{3} \cdot e^{-(x+1)^2-y^2}$$

Tendo em vista que f é uma função contínua definida sobre todo o plano $(\mathbb{R}, \mathbb{R})$, é possível definir uma malha para o plano (x,y) com intervalos de 0,1 entre dois pontos consecutivos em uma mesma direção e limitados ao intervalo de $[-5, 5]$. A malha construída é formada por 101^2

pontos, todos possíveis candidatos a um mínimo global que estão representados na Figura 6.

Figura 6: Representação da curva f.



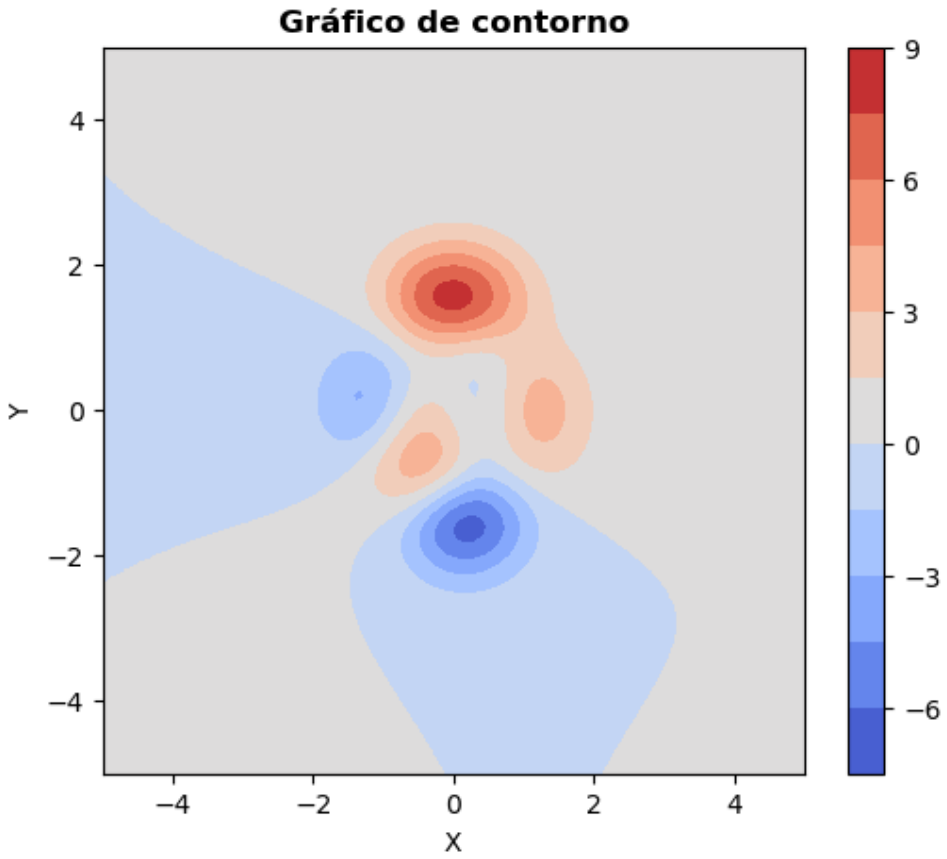
Fonte: Elaborado pelo autor.

A análise gráfica da função permite a identificação de dois pontos críticos de mínimo. O primeiro deles, com coordenadas próximas a (-1,35; 0,20), é um mínimo local e o segundo, de coordenadas (0,23; -1,63) aproximadamente, é um mínimo global. Ambos os pontos são representados no gráfico de contorno da função f na Figura 7.

Assim, caso um subconjunto a ser explorado fosse definido apenas para o semi-plano S_1 tal que $S_1 : \{(x, y) \mid y \geq 0\}$, o mínimo encontrado pela busca exaustiva seria um mínimo local, apesar de reduzir pela metade o tempo de execução do algoritmo.

Portanto, o método busca por força bruta é propício apenas para problemas pequenos, com um número pequenos de variáveis e cujos intervalos de possíveis valores são limitados a um espaço pequeno. Além disso, busca por força bruta é interessante em casos nos quais a facilidade de implementação do método de busca é mais relevante do que a eficiência do método.

Figura 7: Representação do contorno de f.



Fonte: Elaborado pelo autor.

2.3.2 Nelder-Mead

Introduzido em 1965 pelos estatísticos John Ashworth Nelder e Roger Mead, o algoritmo de busca *Nelder-Mead*, também conhecido na literatura como *Downhill Simplex Method*, é um método de busca heurístico que dispensa a necessidade de cálculos de derivadas. Essa característica o torna atrativo para implementação em problemas em que não é possível descrever de forma analítica a função objetivo do problema (HAUPT; HAUPT, 2004).

Apesar do nome, o algoritmo não é relacionado com o método *Simplex*, mas mantém a ideia de construir um *simplex* a cada iteração. Assim, uma forma geométrica elementar formada em N dimensões e possuindo $N + 1$ lados (ou vértices) é produzida para cada iteração, como um triângulo em um problema de duas dimensões.

O objetivo do método é de mover o *simplex* até a região do mínimo buscado e, então, contrair o *simplex* em torno do mínimo até que atinja-se a tolerância do erro pré-estabelecida (HAUPT; HAUPT, 2004).

O método de busca é iniciado com um conjunto de $N + 1$ pontos que formam o *simplex* inicial, de forma que apenas um dos pontos, P_0, é especificado pelo usuário, sendo os outros pontos determinados pela seguinte equação (HAUPT; HAUPT, 2004):

$$P_n = P_0 + c_s \cdot e_n \tag{2.4}$$

Onde:

P_n : Ponto n do *simplex* inicial ($\forall\ n,\ n = 1, 2, ..., N$)

c_s : Constante de escala

e_n : Vetor unitário de dimensão N na direção n

Além disso, define-se os seguintes parâmetros (JIN et al., 2019):

α : Fator de *Reflexão* (tipicamente $\alpha = 1$)

β : Fator de *Expansão* (tipicamente $\beta = 2$)

γ : Fator de *Contração* (tipicamente $\gamma = \frac{1}{2}$)

ρ : Fator de *Encolhimento* (tipicamente $\rho = \frac{1}{2}$)

Para exemplificar o funcionamento do método, toma-se novamente o problema de duas dimensões. Assim, inicia-se a busca com a criação de triângulo inicial de vértices $P_A = (x_A, y_A)$, $P_B = (x_B, y_B)$ e $P_C = (x_C, y_C)$ (Figura 8), com o qual, inicialmente, ordena-se os custos em ordem crescente para cada vértice:

$$f(P_B) \leq f(P_C) \leq f(P_A)$$

Para cada iteração, define-se os índices $h,\ s,\ l$ para os vértices de pior, segundo pior e melhor custo, respectivamente, para o *simplex* de trabalho atual:

$$P_h \mid f_h = max_j\ f_j$$
$$P_s \mid f_s = max_{j \neq h}\ f_j$$
$$P_l \mid f_l = min_{j \neq h}\ f_j$$

Assim:

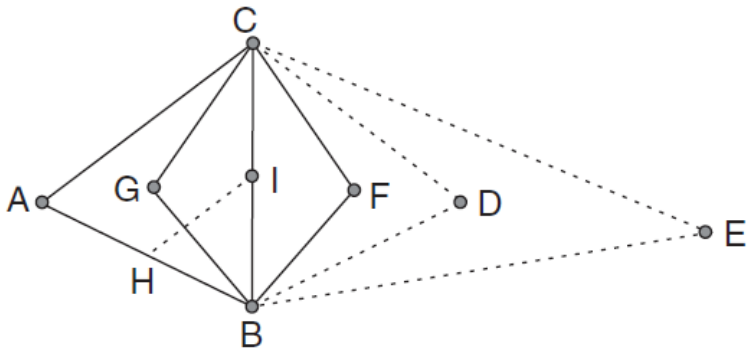
$$f(P_B) \leq f(P_C) \leq f(P_A) \Leftrightarrow f(P_l) \leq f(P_s) \leq f(P_h)$$

Em seguida, ocorre a etapa de Reflexão, em que um novo ponto $P_D = (x_D, y_D)$ é definido como uma reflexão do ponto com o maior custo, P_h (considerado como o ponto P_A nesse exemplo), em relação ao centroide, $P_I = (x_I, y_I)$, formado pelos pontos restantes (P_B e P_C), conforme ilustrado na Figura 8.

$$P_I = \frac{\sum_{i \neq A} P_i}{N},\ N = 2 \tag{2.5}$$

$$P_D = P_I + \alpha(P_I - P_h) \tag{2.6}$$

Figura 8: Representação de uma iteração do Algoritmo de *Nelder-Mead* para um problema de duas dimensões



Fonte: Haupt e Haupt (2004).

Caso o custo do ponto P_D seja tal que:

$$f_l \leq f(P_D) < f_s \Leftrightarrow f(P_B) \leq f(P_D) < f(P_C),$$

substitui-se o ponto de maior custo, P_h, (i.e. P_A no exemplo) pelo ponto P_D para formar-se um novo *simplex* e uma nova iteração é iniciada.

Em seguida, caso o custo de P_D seja menor do que de P_A (ou seja, o melhor custo até o momento: $f(P_D) < f_l$), então é preciso fazer uma Expansão. O ponto de Expansão, $P_E = (x_E, y_E)$, é tal que:

$$P_E = P_I + \beta(P_D - P_I) \tag{2.7}$$

Caso o custo do ponto P_E seja menor do que do ponto P_D (i.e. $f(P_E) < f(P_D)$) , obtém-se um novo *simplex* substituindo o ponto de pior custo, P_h (i.e. P_A), pelo ponto expandido P_E e inicia-se uma nova iteração. Caso o custo de P_E seja maior ou igual do ponto P_D (i.e.

$f(P_D) \leq f(P_E)$), o novo *simplex* é obtido ao fazer a substituição descrita anteriormente pelo ponto refletido P_D.

Porém, caso o custo de P_D seja maior ou igual ao segundo pior custo, f_s (i.e. $f(P_D) \geq f_s$), realiza-se uma Contração. Dois pontos de Contração são estabelecidos, $P_F = (x_F, y_F)$ e $P_G = (x_G, y_G)$, tais que:

$$P_F = P_I + \gamma(P_D - P_I) \tag{2.8}$$

$$P_G = P_I + \gamma(P_h - P_I) \tag{2.9}$$

Apenas o ponto de menor custo entre os pontos F e G é mantido. Caso o ponto escolhido tenha custo menor do que o pior custo encontrado (i.e. $f(P_{F \text{ ou } G}) < f_h$), substitui-se o ponto de pior custo, P_h (i.e. P_A) pelo ponto escolhido. Caso contrário, realiza-se um Encolhimento.

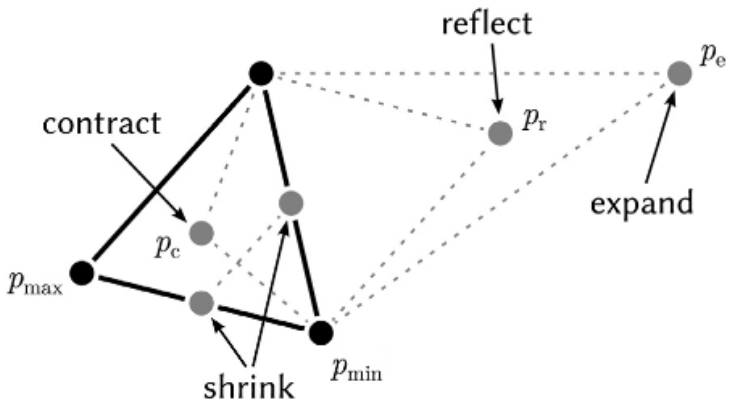
No processo de Encolhimento, todos os pontos, exceto o de melhor custo, P_l, são substituídos e o *simplex* sofre um Encolhimento na direção deste ponto (i.e. P_B para o exemplo). Seja $P_n^{'} = (x_n^{'}, y_D^{'})$ o ponto encolhido do ponto P_n, então os N novos pontos são tais que:

$$P_j^{'} = P_l + \rho(P_j - P_l), \forall j = 1, 2, ..., N \text{ com } j \neq l \tag{2.10}$$

Caso o novo *simplex* formado não atinja os critérios de término do algoritmo (em geral definido como um tamanho mínimo para o *simplex*), uma nova iteração é realizada. A solução proposta como solução ótima é representada pelo ponto de menor custo no último *simplex* formado.

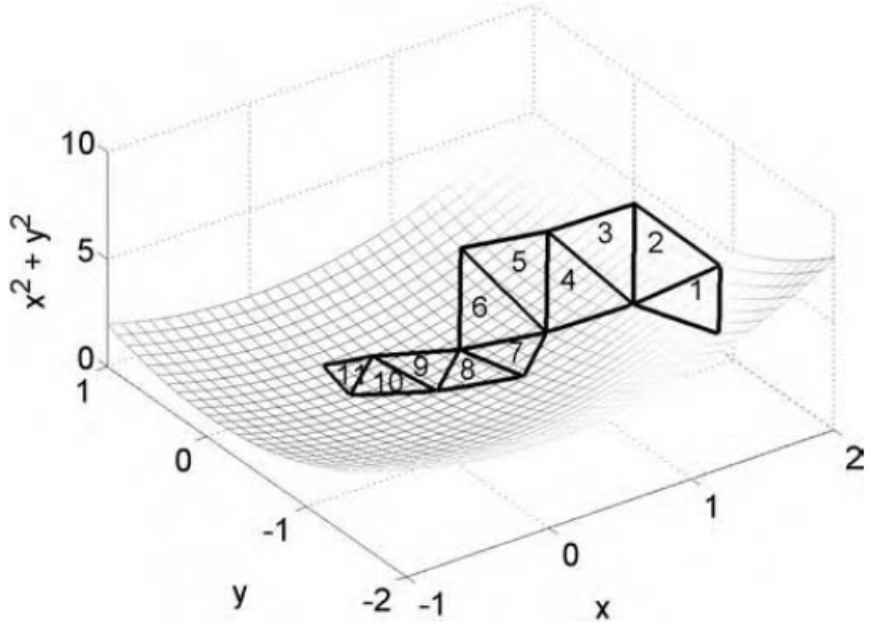
Os processos de Reflexão, Expansão, Contração e Encolhimento descritos acima podem ser observados se forma mais clara na Figura 9. Ainda, um exemplo ilustrativo do funcionamento do algoritmo é representado na Figura 10.

Figura 9: Representação dos processos de Reflexão, Expansão, Contração e Encolhimento do algoritmo de *Nelder-Mead* para um problema de duas dimensões.



Fonte: Adaptado de Cheng e Mailund (2015).

Figura 10: Exemplo de aplicação do algoritmo de *Nelder-Mead*.



Fonte: Haupt e Haupt (2004).

O algoritmo de *Nelder-Mead* não é conhecido por sua velocidade, mas sua robustez o torna atrativo para que seja implementado. Porém, ressalta-se o fato de que o método está exposto ao risco de ficar preso em mínimos locais, o que faz necessário a combinação do método com outros métodos de busca aleatória para diminuir esse risco (HAUPT; HAUPT, 2004).

2.3.3 Genetic Algorithm

O método de busca de Algoritmo Genético (*Genetic Algorithm* - GA) é uma metaheurística para otimização combinatória que foi desenvolvida pelo pesquisador John Holland em 1975 e é inspirada nos princípios de genética, seleção natural, mutação e cruzamento. Nesse método, uma população de indivíduos evolui sob certas restrições e apenas os indivíduos que melhor se adaptam ao problema (i.e. apresentam o menor custo), têm seus genes propagados nas próximas gerações (HAUPT; HAUPT, 2004).

O algoritmo genético, assim como o algoritmo de *Nelder-Mead*, é um método de otimização que dispensa a necessidade de cálculos de derivadas da função objetivo, além de poder ser aplicado tanto para otimização de variáveis contínuas como discretas. Porém, uma desvantagem comum desse algoritmo é o maior tempo necessário para atingir resultados satisfatórios quando comparado a outros métodos de busca mais tradicionais, embora o Algoritmo Genético possa ser aplicado em problemas nos quais esses outros métodos são incapazes de encontrar soluções (HAUPT; HAUPT, 2004).

A cada iteração do algoritmo (também conhecido como geração), um conjunto de indivíduos (a população) tem seus níveis de "adequação" ao problema (i.e. seus custos) computados a partir da configuração de genes particular de cada indivíduo (aqui denominado como o cromossomo).

Cada indivíduo representa um candidato à solução do problema e seu cromossomo é definido por um vetor $1 \times N_{var}$, no qual N_{var} é o número de variáveis do problema:

$$Cromossomo = [x_1, x_2, ..., x_{N_{var}}]$$

$$Custo = f(Cromossomo) = f(x_1, x_2, ..., x_{N_{var}}) \tag{2.11}$$

O algoritmo é iniciado com uma população inicial de N_{pop} indivíduos representada por uma matriz $N_{pop} \times N_{var}$. Em geral, a população inicial é definida de forma aleatória com base nas restrições de valores de cada variável. Assim, seja j um gene (uma variável) qualquer do problema:

$$x_{j,k,0} = (u_j - l_j) \cdot p_{norm} + l_j \tag{2.12}$$

Onde:

$x_{j,k,0}$: Valor inicial do gene j para o indivíduo k

u_j : Valor máximo para a variável j

l_j : Valor mínimo para a variável j

p_{norm} : Número aleatório entre 0 e 1

Com a população definida, os custos de cada indivíduo são computados e ordenados. A definição da próxima geração de indivíduos passa por um processo de Seleção natural para que apenas os cromossomos mais adaptados possam sobreviver e gerar descendentes. Assim, para cada geração, N_{keep} indivíduos sobrevivem ($N_{keep} < N_{pop}$) e seguem para a etapa de acasalamento, enquanto o restante é descartado.

A decisão de quantos indivíduos de cada geração devem ser mantidos é arbitrária. Permitir poucos indivíduos sobrevivam limita a disponibilidade de genes para os descendentes, enquanto um número grande de indivíduos sobreviventes permite que genes com performance baixas sejam propagados. É comum que a taxa de seleção de indivíduos seja próxima a 50% (HAUPT; HAUPT, 2004).

Os indivíduos que sobreviveram são agrupados em pares de forma aleatória para que possam gerar descendentes. Note que a formação desses pares pode ocorrer de outras formas, como por exemplo fazendo priorizações a partir dos custos. Cada par de pai e mãe gera pelo menos um descendente (em geral, cada casal produz 2 filhos) e o processo é repetido até que $N_{pop} - N_{keep}$ filhos sejam gerados.

Durante a etapa de acasalamento, a geração dos filhos depende dos genes dos pais que geraram o descendente. Diferentes abordagens podem ser tomadas para definir a combinação dos cromossomos dos pais.

Uma das abordagens mais simples é a escolha aleatória de alguns genes (pontos de acasalamento) para serem trocados entre os pais. Assim, um exemplo de acasalamento é:

$$Pai = [x_{1,p}, x_{2,p}, x_{3,p}, x_{4,p}, ..., x_{N_{var},p}]$$
$$Mãe = [x_{1,m}, x_{2,m}, x_{3,m}, x_{4,m}, ..., x_{N_{var},m}]$$
$$Filho_1 = [x_{1,m}, x_{2,p}, x_{3,m}, x_{4,m}, ..., x_{N_{var},p}]$$
$$Filho_2 = [x_{1,p}, x_{2,m}, x_{3,p}, x_{4,p}, ..., x_{N_{var},m}]$$

Outra possível abordagem é, para cada gene, a escolha aleatória de qual dos pais irá passar

seu material genético ao filho.

O problema das abordagens mencionadas acima é que elas não adicionam informações novas à prole, sendo realizadas apenas diferentes combinações entre os genes. Para remediar esse problema, são sugeridos alguns métodos de mistura (*blending methods*) que contam com a ideia de mutações para adicionar novos materiais genéticos à população.

Assim, seja j um gene (uma variável) para um descendente:

$$x_{Filho,j} = \beta x_{Pai,j} + (1 - \beta) x_{Mãe,j} \tag{2.13}$$

Onde:

$x_{Filho,j}$: Valor do gene j do descendente

$x_{Pai,j}$: Valor do gene j do Pai

$x_{Mãe,j}$: Valor do gene j da Mãe

β : Fator de contribuição $\in [0, 1]$

Dessa forma, o gene do filho passa a ser uma combinação linear dos genes de seus geradores. A seleção de quantas e quais variáveis devem ser misturadas é arbitrária e fica a critério do usuário. Além disso, os valores de β podem ser particulares para cada gene (i.e. $\beta_j, \forall\, j = 1, 2, ..., N_{var}$), mas em geral costuma-se adotar o valor de $\beta = 0,5$ (HAUPT; HAUPT, 2004).

Além da própria mistura dos genes dos pais, o Algoritmo Genético considera a possibilidade de mutações na população, um recurso importante para que seja evitado que a população convirja para mínimos locais, sem que consiga sair. Em geral, define-se uma taxa de mutação, τ, cujo valor costuma ser estabelecido em 20% (HAUPT; HAUPT, 2004). Assim, com essa taxa, 20% dos genes ($N_{pop} \times N_{var}$) estão sujeitos a mutações. Note que é comum que seja adotada uma prática de "elitismo", na qual os N_{elite} melhores cromossomos não são submetidos a mutações, fazendo com que o número total de genes modificados seja:

$$N_{\text{Mutações}} = (N_{pop} - N_{elite}) \times N_{var} \tag{2.14}$$

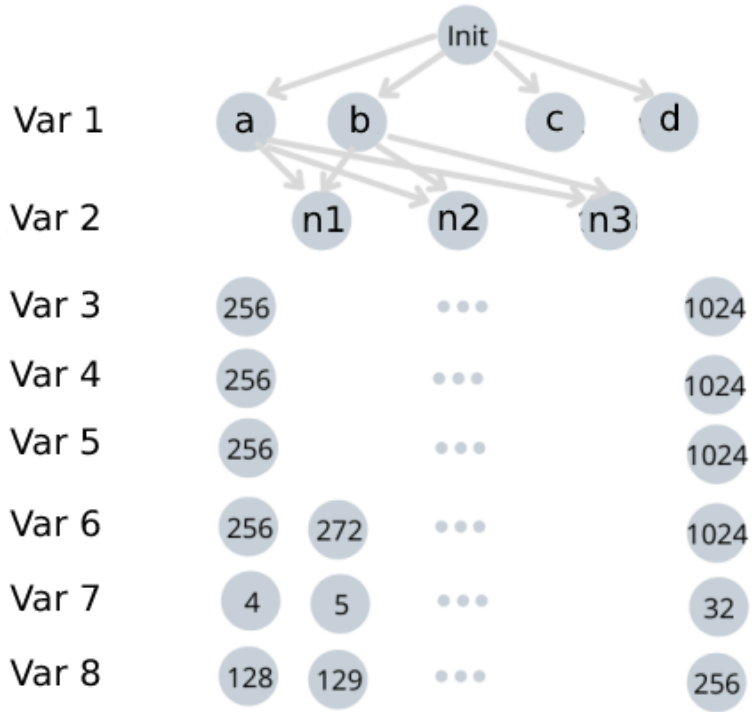
O processo descrito acima é repetido até que se atinja o critério de parada do algoritmo, como a taxa de variação entre as gerações, tempo de execução do código, uma solução considerada satisfatória, etc.

2.3.4 Ant-Colony

O algoritmo de otimização *Ant Colony* (*Ant Colony Optimization* - ACO) é uma metaheurística populacional inspirada na comunicação entre formigas com o uso de feromônios. Esse método de busca foi introduzido por Marco Dorigo em 1992 para resolução do *traveling salesman problem* (TSP), quando pôde mostrar seu valor na resolução de problemas combinatórios (CHENG; MAILUND, 2015).

A ideia por trás do algoritmo é de usar o princípio de rastro de feromônios deixados pelas formigas pelos caminhos percorridos por elas, que na natureza são um indicativo para que outras formigas possam seguir o mesmo caminho e cheguem até o alimento buscado. Naturalmente, as formigas optam por caminhos mais curtos (de menor custo) para que possam atingir o objetivo da colônia, como a busca por alimentos, e quanto mais formigas percorrem o mesmo percurso, uma quantidade maior de feromônios é deixada no trajeto, levando a uma eventual convergência na movimentação da colônia (CHENG; MAILUND, 2015).

Figura 11: Representação de um problema de otimização em grafo, onde cada camada representa os possíveis valores de uma certa variável.



Fonte: Elaborado pelo autor.

Esse mesmo princípio é usado pelo ACO, em que as formigas de cada geração percorrem um grafo no qual, para cada nível de camada, escolhem um valor para a variável do problema

representada pela camada, como ilustrado na Figura 11.

Semelhante ao Algoritmo Genético, cada iteração do ACO é formada por uma população de agentes de tamanho N_{pop} e, ao final de cada iteração, apenas as melhores formigas são recompensadas.

Inicialmente, todos os caminhos do grafo que representa o problema possuem uma quantidade de feromônio τ_0. A escolha de um caminho por uma formiga é probabilística e depende da quantidade de feromônios existente no percurso. Assim, seja i um nó de origem e j um nó de destino, então, a probabilidade de que a formiga escolha o nó j como seu próximo destino entre N possíveis caminhos é dada por:

$$p_{i,j} = \frac{\tau_{i,j}^{\alpha} \eta_{i,j}^{\beta}}{\sum_{k=1}^{N} \tau_{i,k}^{\alpha} \eta_{i,k}^{\beta}} \tag{2.15}$$

Onde:

$\tau_{i,j}$: Quantidade de feromônios entre os nós i e j

$\eta_{i,j}$: Visibilidade entre os nós i e j

α : Fator de influência da quantidade de feromônios para a escolha do caminho

β : Fator de influência da visibilidade para a escolha do caminho

Note que a ideia de visibilidade entre nós, representada por η, é um indicativo da "distância" existente entre os nós, o que pode ser um parâmetro útil para alguns problemas como o TSP. Ainda, pela equação acima, pode-se perceber que a probabilidade de escolha de um caminho é dependente da quantidade total de feromônios existentes na mesma camada.

Para uma dada geração, após todas as N_{pop} formigas escolherem seus caminhos, o custo de cada solução (i.e. cada caminho escolhido por cada formiga) é computado e as formigas são ranqueadas de acordo com esse custo. Em seguida, a quantidade de feromônios é atualizada e depende da taxa de evaporação de feromônios, ρ, ou seja, a quantidade de feromônios que deixa os caminhos a cada iteração; e da quantidade Q adicionada a cada caminho percorrido por cada formiga. Assim, para dado caminho entre os nós i e j, tem-se que:

$$\tau_{i,j}^{t+1} \leftarrow (1 - \rho) \cdot \tau_{i,j}^{t} + \sum_{k=1}^{N_{pop}} \Delta\tau_{i,j}^{k} \tag{2.16}$$

$$\Delta\tau_{i,j}^{k} = \begin{cases} Q, & \text{se a formiga } k \text{ passou pejo caminho } ij \\ 0, & \text{caso contrário} \end{cases}$$

Algumas variações do ACO são propostas na literatura e utilizam outros métodos para atualizar a quantidade de feromônios a cada iteração, como:

- Sistema elitista: Apenas as N_{elite} melhores formigas encontradas até o momento adicionam feromônios, garantindo que bons caminhos não sejam perdidos;
- Sistema Max-min: A quantidade de feromônios em cada caminho é limitada por um limite máximo e mínimo, o que limita a intensificação de caminhos pelas formigas e promove maior diversificação na escolha dos percursos;
- Sistema ASRank: Com base no ranqueamento de custos das soluções, as formigas que apresentam resultados melhores desempenhos adicionam mais feromônios quando comparado a formigas com performances piores.

O processo descrito acima é repetido até que se atinja o critério de parada do algoritmo.

2.4 Resolução do SELSP

2.4.1 Revisão da literatura

Ao analisar-se os materiais da literatura já publicados para o *Stochastic Economic Lot Scheduling Problem*, pode-se encontrar o trabalho de Wagner e Smits (2004) em que o controle de estoques foi analisado diante do cenário do SELSP considerando um sistema de produção com uma política de revisão de estoque periódica (R, S), focando na definição de sequenciamentos ótimos.

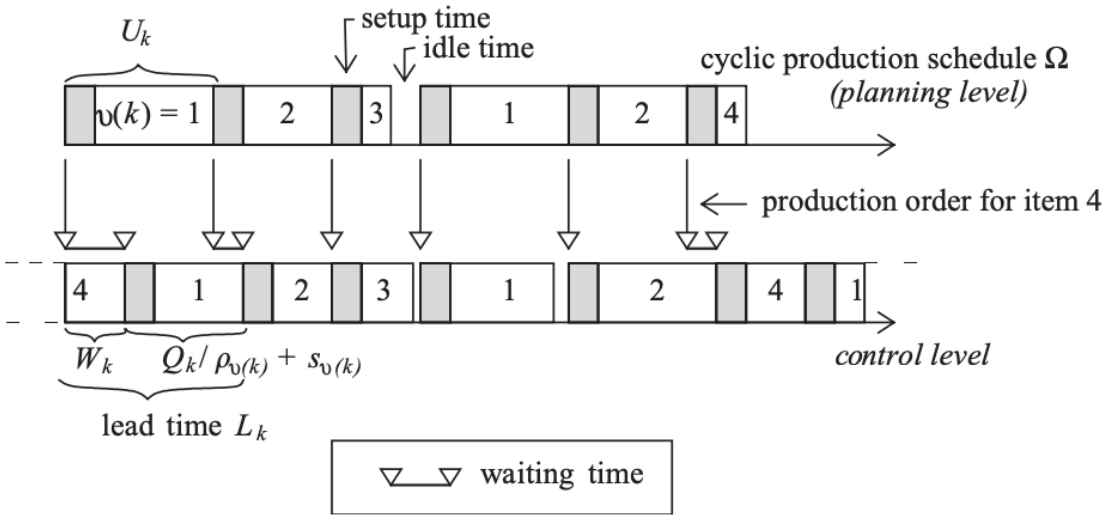
A cada vez que o estoque é revisado, uma ordem de produção é emitida para o produto revisado e a frequência de revisão é dada pelo tempo de ciclo de cada produto. O objetivo da otimização é determinar um sequenciamento de produção fixo que poderá ser repetido indefinidamente e que minimize os custos de *setup* e estocagem no longo prazo. Mesmo que os fatores estocásticos tragam incertezas ao sistema e gerem "atrasos" em relação ao que foi planejado, a sequência estabelecida é mantida e são ajustados os valores dos lotes de produção, como mostrado na Figura 12.

O método de busca proposto pelos autores é um método de busca local baseado no método de *Simulated Annealing*, tendo como palpite inicial uma solução baseada no período básico de

produção de cada produto. Assim, os ciclos de produção de cada produto são mantidos como múltiplos de seus períodos básicos. A solução proposta pelo método de busca local tem sua viabilidade verificada na sequência e o custo da solução é computado.

Mais uma vez, a metodologia de busca adotada é dependente da solução inicial proposta e a possibilidade de busca de soluções que fujam do padrão estabelecido pela solução inicial depende da configuração dos hiperparâmetros do algoritmo do *Simulated Annealing*.

Figura 12: Sequenciamento estático proposto por Wagner e Smits (2004).



Fonte: Wagner e Smits (2004).

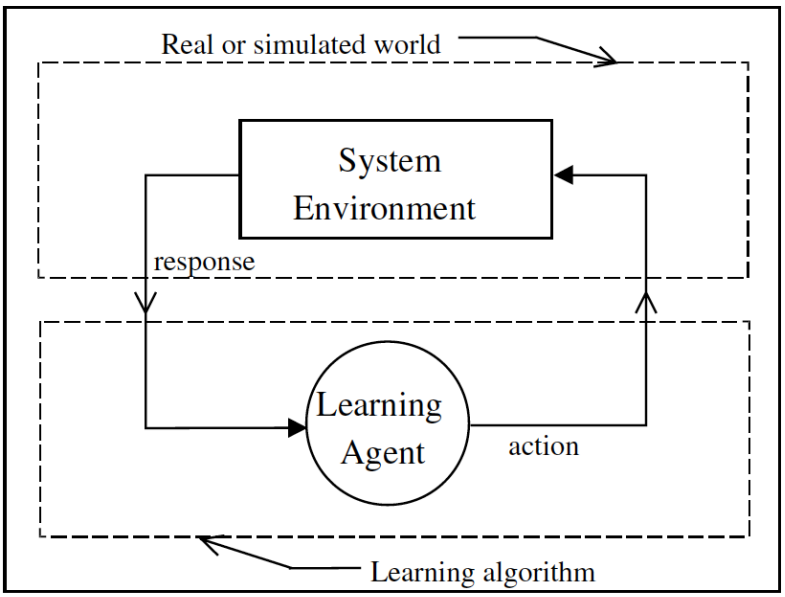
Paternina-Arboleda e Das (2005) adotam uma outra estratégia para a resolução do SELSP por meio de uma abordagem multi-agente de *reinforcement learning* (RL) para a obtenção de políticas de controle de estoque dinâmicas.

O modelo proposto pelos autores adota uma abordagem de simulação-otimização para aprimorar gradativamente a política de sequenciamento de produtos, no qual cada agente RL representa um tipo de produto (Figura 13). Caso um produto i esteja sendo produzido, assim que o sistema atinge seu nível de estoque básico R_i (nível de estoque máximo), o agente RL$_i$ pode tomar duas decisões: trocar a produção para um outro produto j ou manter a máquina (responsável pela produção) inativa. Caso haja troca do tipo do produto, o processo é repetido, agora para o produto j. Além disso, esse processo é repetido continuamente até que se obtenha uma política de controle de estoques clara.

Note que os agentes RL's não determinam os valores de nível de estoque básico R_k ($\forall\ k = 1, 2, ..., N$). Porém, os autores ainda aplicam uma busca de segundo nível para encontrar a combinação mais apropriada dos valores de R_k por meio do uso do *software* de otimização

OptQuest em conjunto com o *software* de simulação *ARENA*. A busca realizada é local e é aplicada sobre um espaço de ± 5 unidades.

Figura 13: Esquema do modelo de *reinforcement learning* de Paternina-Arboleda e Das (2005).



Fonte: Paternina-Arboleda e Das (2005).

Kämpf e Köchel (2006) exploram um problema similar ao SELSP, mas assumem a possibilidade de atender demandas pendentes. Os autores estudam a performance de políticas de sequenciamento simples e buscam valores ótimos para os parâmetros de estoque por meio de um modelo de simulação-otimização.

De forma a simplificar a questão do sequenciamento dos produtos a serem manufaturados, Kämpf e Köchel (2006) comparam três diferentes políticas: FCFS (*First Come First Served*), aleatória (cada item tem chance equiprovável de ser escolhido) e cíclica (ordem definida *a priori*, considerando apenas itens que precisam ser produzidos).

A estratégia adotada pelos autores para a busca de soluções para o problema foi a utilização de um algoritmo genético. Segundo eles, a escolha pelo algoritmo se deve à flexibilidade de aplicação do método para diferentes problemas de otimização, à sua robustez em relação à solução inicial e à necessidade de poucos *inputs*. Ainda, Kämpf e Köchel (2006) citam a possibilidade de criar "mutações" das soluções buscadas a cada geração do algoritmo, o que permite ampliar o espaço de busca das soluções do problema e minimizar a exposição a riscos de aprisionamento a mínimos locais.

Já Löhndorf, Riel e Minner (2014) abordam o mesmo problema de SELSP considerando tempos de *setup* dependentes da sequência de produtos e demandas estacionárias, no qual os

autores se dedicam à otimização da política de sequenciamento dos ciclos de produção.

Löhndorf, Riel e Minner (2014) comparam três políticas de produção. A primeira delas é a *Common cycle policy* (CCP), na qual uma sequência ótima é definida *a priori* de forma a minimizar o tempo total de *setup* considerando o sequenciamento de todos os produtos. Cada produto é produzido exatamente uma vez a cada ciclo.

A segunda política analisada é a *Fixed-cycle policy* (FCP), que calcula primeiramente a frequência ótima de produção e depois usa essa informação para a construção da sequência de produção. Essa política pode ser mais vantajosa para casos em que a demanda dos produtos é desbalanceada, uma vez que em políticas como a CCP, produtos de menor demanda são produzidos de forma frequente, mesmo sem necessidade.

Por fim, a terceira política analisada é a *Balanced cycle policy* (BCP), que busca evitar que sequenciamentos irregulares e desbalanceados sejam adotados pelo algoritmo de busca caso tenha uma grande disparidade de custos de *setup*. Dessa forma, produtos com baixo custo de *setup* são produzidos diversas vezes e outros com custo mais caro de *setup* são produzidos com menor frequência, levando a níveis maiores de estoque. Para essa política, além da minimização do tempo de *setup* total, existe outra otimização em paralelo, na qual a variabilidade *inter-setup* é minimizada. Assim, o desvio padrão dos índices de ocorrência de um mesmo produto em um mesmo sequenciamento de produção é minimizado, o que permite garantir padrões mais regulares de sequenciamentos de produção.

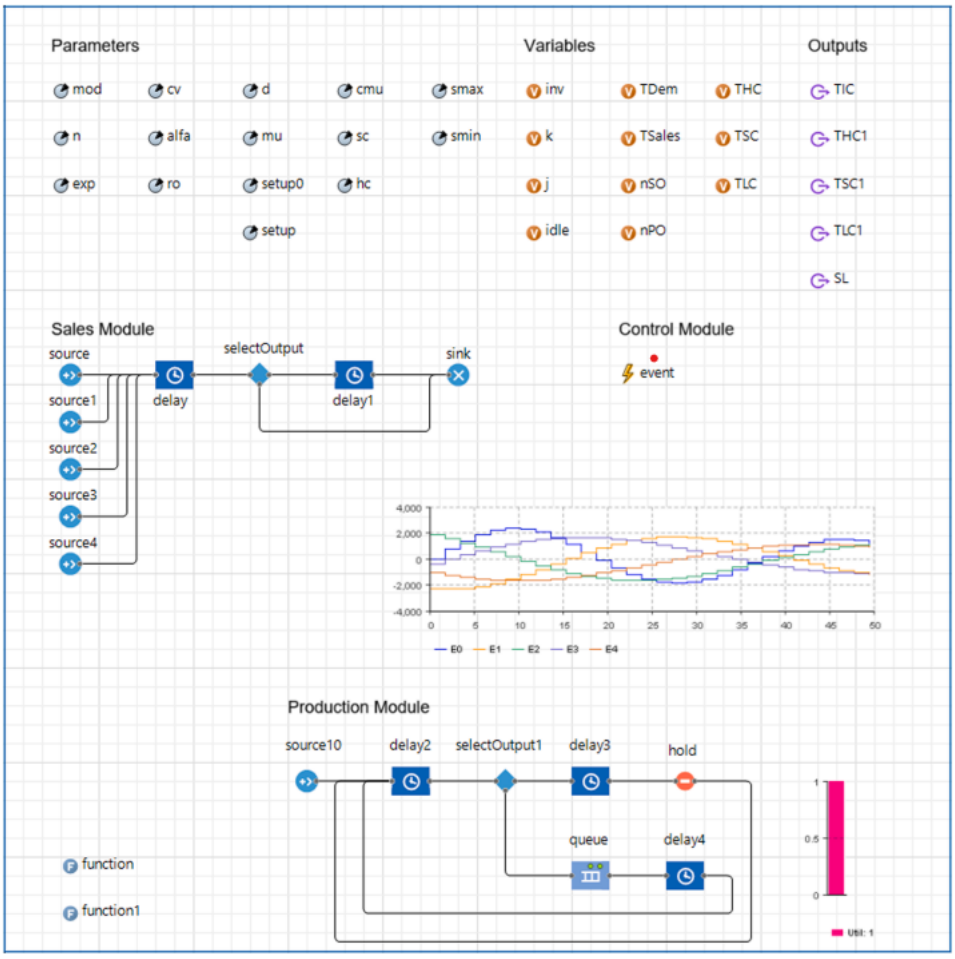
O método de busca adotado pelos autores se baseia em uma busca iterativa em duas etapas. A primeira delas é uma busca global usando o algoritmo de *Covariance matrix adaptation evolution strategy* (CMA-ES), o qual busca vetores que representem sequenciamentos de produção de forma a minimizar a esperança do custo médio da solução proposta. O algoritmo é iniciado com um palpite inicial da melhor solução e uma região de confiança onde se espera encontrar a melhor solução. Com o *output* do primeiro algoritmo de busca, um segundo algoritmo é aplicado, realizando uma busca local da melhor solução na "vizinhança" da sugestão do CMA-ES. Nessa segunda etapa, os autores analisam o desempenho de duas heurísticas, a *Lin–Kernighan* e a *2-opt*.

A utilização de um método de busca em duas etapas, com uma busca global e uma busca local, como proposto por Löhndorf, Riel e Minner (2014), é interessante, pois permite refinamentos das soluções de métodos de busca global. Todavia, o desempenho da metodologia proposta pelos autores ainda é dependente do palpite inicial estabelecido e do hiperparâmetro ligado à região de confiança da solução, exigindo certo conhecimento prévio da possível solução ótima.

Finalmente, cita-se o trabalho de Mesquita e Tomotani (2022), em que o controle de estoques foi analisado diante do cenário do SELSP considerando tempos de *setup* dependentes do sequenciamento dos produtos. Por meio de uma abordagem de simulação-otimização, os autores investigaram o problema analisando o impacto de diferentes parâmetros do sistema (ex: política de sequenciamento de produtos, número de produtos, flutuação da demanda, custos etc.) na variável resposta (custo total de estoque) e nas variáveis de decisão, (s, S) - que correspondem ao ponto de reabastecimento de estoques e ao nível de estoque máximo dos produtos (consideradas iguais para todos os produtos).

Além disso, foram consideradas duas políticas de programação da produção: FIS (*First in Sequence*), que define a ordem de verificação dos níveis de estoques dos itens, iniciando a produção para o primeiro que apresentar um nível abaixo do ponto de reabastecimento; e o LDS (*Lowest Days of Supply*), que, dentre os itens que estão com níveis abaixo do ponto de reabastecimento, prioriza aquele que tiver menor tempo de cobertura da demanda.

Figura 14: Exemplo de implementação do modelo de simulação no *AnyLogic* de Mesquita e Tomotani (2022).

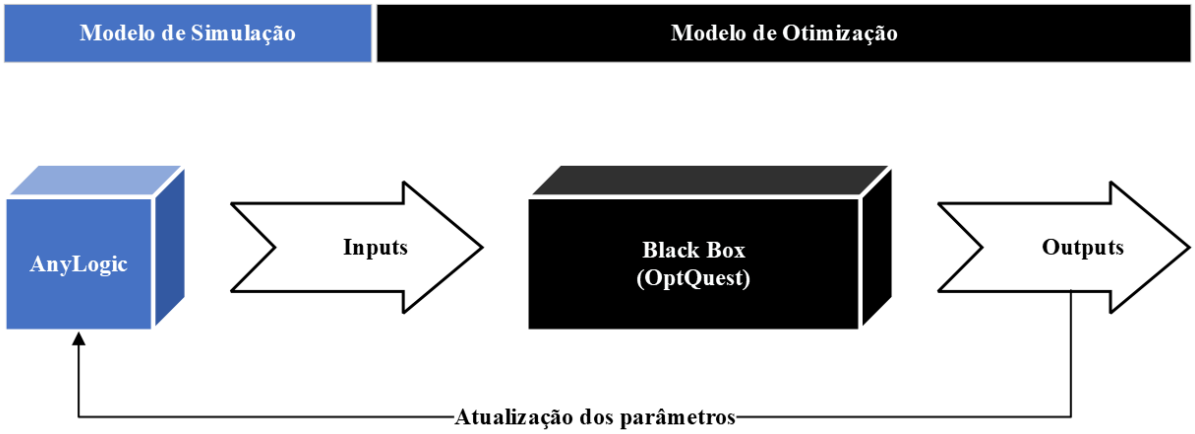


Fonte: Mesquita e Tomotani (2022).

A metologia adotada por Mesquita e Tomotani (2022) levou à implementação do modelo de simulação em um software dedicado, o *AnyLogic*, como visto na Figura 14. Além disso, a otimização dos parâmetros de estoque (s, S) de forma a minimizar o custo total de estoque foi feita com o auxílio de outro *software*, o *OptQuest*, um dos *softwares* líderes do mercado de soluções de otimização. Além disso, a utilização do método de simulação-otimização permitiu aos autores a possibilidade de estabelecer um DoE para testar diferentes condições operacionais (ex: taxa de produção, nível de utilização das máquinas, variabilidade da demanda etc.).

A utilização de *softwares* como o *OptQuest* impede que os autores do estudo tenham conhecimento a respeito do método de otimização utilizado e o critério de parada do algoritmo proposto, limitando possíveis otimizações para maior adequação da ferramenta ao problema explorado. Dessa forma, a etapa de otimização torna-se um processo caixa preta (Figura 15), na qual é possível ter conhecimento apenas dos *inputs* e dos *outputs* do modelo de otimização. Além disso, a utilização de um *software* com um método genérico de otimização pode mostrar menor eficiência em relação a uma ferramenta específica desenhada para o problema estudado.

Figura 15: Modelo de Otimização do *OptQuest* como um processo *Black box*.



Fonte: Elaborado pelo autor.

2.4.2 Síntese da literatura

Tendo em vista a análise dos estudos levantados na seção 2.4.1 e em linha com as revisões de literatura sobre o SELSP realizadas por Sox et al. (1999) e Winands, Adan e van Houtum (2011), pode-se notar que a literatura existente sobre o *Stochastic Economic Lot Scheduling Problem* difere-se não apenas na metodologia para resolução do problema, mas também na abordagem e foco dado ao SELSP.

Os diferentes estudos realizados até o momento tomam duas abordagens distintas: a

definição de tamanho de lote e o sequenciamento da produção. Em geral, os estudos analisados adotam simplificações do problema para que possam focar na resolução de um dos pontos mencionados acima. A título de exemplo, Kämpf e Köchel (2006) restringiram as possibilidades de sequenciamento de produtos e otimizaram os parâmetros de controle de estoque. Por outro lado, Paternina-Arboleda e Das (2005) restringiram o espaço amostral dos possíveis valores dos parâmetros de estoque e focalizaram na determinação de métodos de sequenciamento dos produtos.

Nota-se ainda que, para a determinação do sequenciamento de produtos, duas principais estratégias são adotadas: políticas de sequenciamento cíclicas ou dinâmicas. Para as políticas cíclicas, algoritmos são utilizados para definir uma sequência fixa de produção a ser repetida continuamente pela fábrica (WAGNER; SMITS, 2004). Já para sequenciamentos dinâmicos, a decisão passa a ser de definir políticas de priorização dos produtos (PATERNINA-ARBOLEDA; DAS, 2005).

A definição das políticas de controle do tamanho dos lotes de produção costuma ser mais simples (e possivelmente mais fácil de aplicar em situações reais) e, em geral, é baseada em políticas tais como os reabastecimentos (R, Q) ou (s, S), vistos na seção 2.1.3. A complicação para essa parte do problema está mais ligada à busca de métodos eficientes para a determinação dos parâmetros de controle de estoque.

Com relação aos métodos de busca observados, pode-se notar que as sugestões dadas pelos autores analisados também são distintas. Os algoritmos utilizados são variados, podendo tomar proveito de buscais globais (KÄMPF; KÖCHEL, 2006), locais (WAGNER; SMITS, 2004) ou uma combinação dos dois (LÖHNDORF; RIEL; MINNER, 2014).

Por fim, com relação à implementação dos modelos de simulação e otimização propostos pelos estudos analisados, notou-se que há certa diversidade, com autores fazendo o uso de *softwares* comerciais, tais como *AnyLogic* e *ARENA* para simulações e o *OptQuest* para otimização.

Dessa forma, no presente Trabalho, a abordagem para o *Stochastic Economic Lot Scheduling Problem* será a definição de tamanho de lotes para uma política de reabastecimento (s, S), definindo sequenciamentos por meio de políticas simples de priorização de produtos. Além disso, o Trabalho seguirá com a abordagem de simulação-otimização, implementando tanto o modelo de simulação como o de otimização em um *software* aberto, o *Python*, conforme detalhado no capítulo 3.

3 MÉTODO

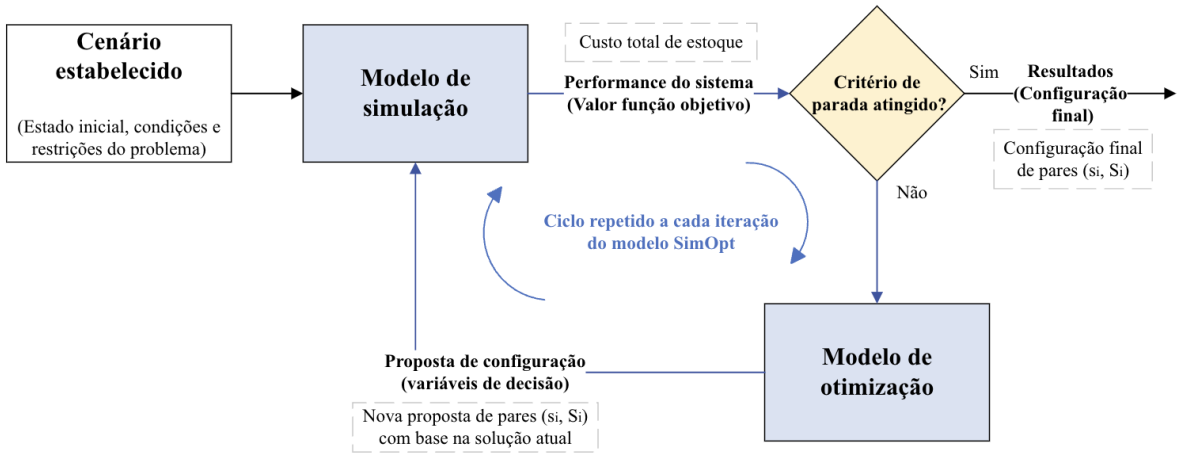
Conforme estabelecido na seção 1.3, este Trabalho de Formatura propõe-se a identificar e implementar métodos numéricos eficientes para a calibração dos parâmetros de controle de estoque (s_i, S_i) para o *Stochastic Economic Lot Scheduling Problem*.

Para tal, adota-se a estratégia de simulação-otimização implementada em *Python* por meio da qual são usados dois modelos distintos, um de simulação, que simula uma máquina com produção em lotes e N estoques, com base nos parâmetros do problema; e outro de otimização, que aplica métodos numéricos, tais como os apresentados na seção 2.3, para buscar otimizar os pares (s_i, S_i) e, consequentemente, os custos e nível de serviço.

Devido às características do problema, que impedem a definição da forma analítica da função objetivo, é possível perceber a importância da ação combinada dos dois modelos, uma vez que a avaliação das soluções propostas pelo modelo de otimização só pode ser feita com o uso do modelo de simulação. Assim, o algoritmo de busca é alimentado com a saída do modelo de simulação em cada iteração, até que o critério de parada do modelo simulação-otimização (definido *a priori*) seja atingido, conforme apresentado na Figura 16.

Note que essa avaliação das soluções é feita com base no custo total de estoque computado ao final da simulação para a configuração analisada, sendo o objetivo do otimizador encontrar a melhor configuração de N pares (s_i, S_i) que minimizem esse custo.

Figura 16: Diagrama do princípio de funcionamento do modelo de simulação-otimização.



Fonte: Elaborado pelo autor.

Além disso, para identificar os métodos de busca implementados mais eficientes, neste Trabalho, é adotada a metodologia de delineamento de experimentos (DoE). Contudo, para que os algoritmos possam ter seus desempenhos comparados, é preciso ajustar os parâmetros de cada método para que eles estejam em suas melhores condições de execução.

Dessa forma, os experimentos realizados são divididos em duas etapas:

(I) Experimentos de calibração;

(II) Experimentos de comparação.

Experimentos de calibração

Primeiramente, realizam-se os experimentos de calibração, nos quais os métodos de busca são testados em instâncias do problema similares às propostas nos experimentos de comparação. O objetivo dessa etapa é de verificar qual a melhor configuração dos parâmetros particulares de cada método de busca para a resolução do problema estudado.

Assim, um plano de experimentos de calibração é desenhado para cada algoritmo, estabelecendo diferentes níveis para cada parâmetro com base em referências da literatura, conforme detalhado no capítulo 5.1. Define-se como a melhor configuração de hiperparâmetros aquela que obtiver o menor custo de estoque total obtido durante os experimentos.

Experimentos de comparação

Em seguida, com os algoritmos calibrados, realizam-se os experimentos de comparação. Nessa etapa, os métodos de busca implementados e calibrados são testados sob diversas "condições de operação" estabelecidas por meio de diferentes instâncias do problema que se distinguem graças a mudanças nos parâmetros gerais, tais como número de produtos, variação da demanda, demanda média, custos unitários etc.

Sendo assim, por meio de um novo conjunto de experimentos, conforme detalhado no capítulo 6, é possível verificar a robustez das soluções encontradas pelos algoritmos e dos métodos de busca de solução para condições operacionais mais exigentes e problemas de maior complexidade, respectivamente.

Nos capítulos a seguir, apresentam-se em maior detalhe os modelos de simulação e de otimização, assim como os experimentos desenhados e seus respectivos resultados.

4 MODELO DE SIMULAÇÃO

Neste capítulo, apresentam-se o modelo conceitual da linha de produção simulada e sua implementação no modelo de simulação proposto.

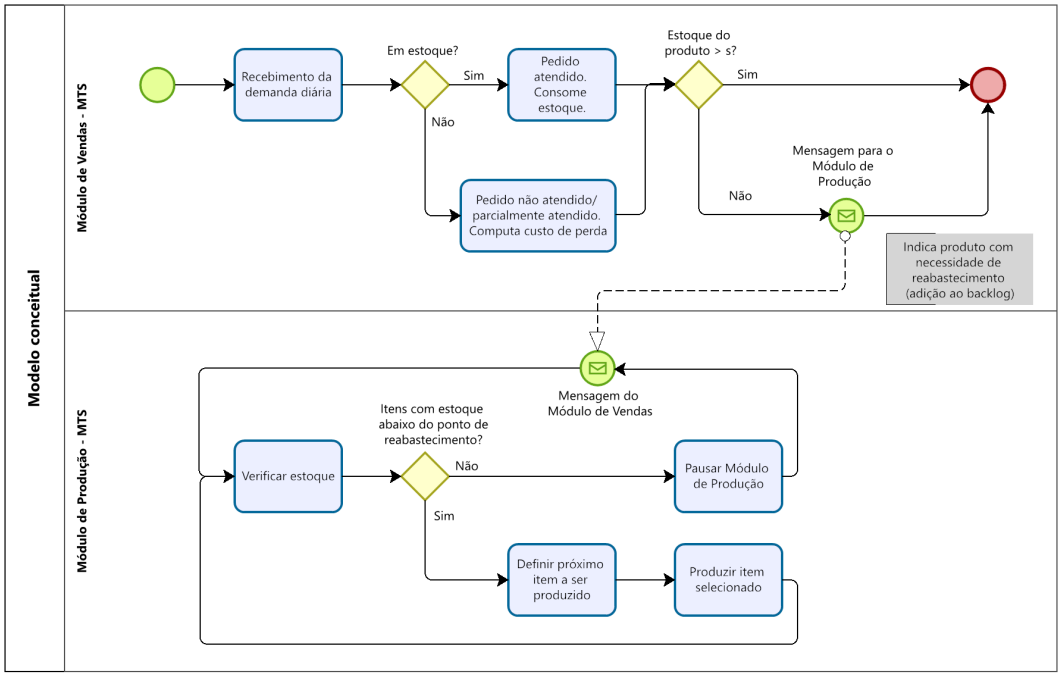
4.1 Modelo conceitual

O presente modelo conceitual representa a organização de uma fábrica. O modelo é estruturado em dois módulos: de Vendas e de Produção, conforme sugerido por Altiok e Melamed (2010).

Neste Trabalho, considera-se uma linha de produção capaz de produzir N produtos, produzidos em lotes, com tempos de *setup* para cada troca de produto na linha.

Todos os produtos manufaturados na linha de produção seguem uma estratégia de *Make-to-Stock* (MTS) para atender uma demanda diária variável que consome os produtos do estoque de produtos acabados da fábrica ao final de cada dia.

Figura 17: Modelo conceitual de vendas e produção para produtos MTS.



Fonte: Adaptado de Mesquita e Tomotani (2022).

A Figura 17 representa a lógica por trás dos módulos de vendas e produção para a estratégia MTS adotada.

O recebimento das demandas diárias é realizado pelo Módulo de Vendas, que recebe e consolida um único pedido por dia para cada produto, verifica se existe estoque suficiente para os itens solicitados e atende a demanda. Caso o estoque seja insuficiente para atender a demanda por completo, a demanda é satisfeita de forma parcial e os itens faltantes são considerados como vendas perdidas (sem *backlog)*. Assim, tem-se que:

$$Y_{i,t} = min(X_{i,t}, I_{i,t}) \tag{4.1}$$

Onde:

$Y_{i,t}$: Vendas do produto i no dia t

$X_{i,t}$: Demanda do produto i no dia t

$I_{i,t}$: Quantidade em estoque para o produto i no dia t

Após a verificação do estoque e do atendimento das demandas do dia, compara-se o estoque remanescente dos produtos com seus respectivos pontos de reabastecimento s_i. Caso a quantidade em estoque seja maior do que o estoque mínimo, nada é feito. Porém, caso contrário (i.e. $I_{i,t} \leq s_i$), se o Módulo de Produção estiver ocioso, ele é ativado e uma mensagem é enviada a ele contendo informações sobre os itens que devem ser produzidos, adicionando-os em um *backlog* de produção. Note que, caso haja necessidade de produzir um item e o Módulo de Produção já esteja ativo, é verificado se o item a ser produzido já está no *backlog* (i.e. sua produção já foi solicitada) e, caso não esteja, o mesmo é adicionado nessa lista.

A decisão do próximo produto a ser produzido e da quantidade a ser produzida é feita no Módulo de Produção. A priorização para o sequenciamento dos produtos segue uma estratégia LDS (*Lowest Days of Supply*), ou seja, os itens pendentes (i.e. que estão no *backlog* de produção) são ordenados conforme o tempo de cobertura de demanda e o produto com menor cobertura é priorizado.

Esse processo de verificação da prioridade dos produtos é repetido toda vez que há troca de um SKU na produção para que sejam atualizados os tempos de cobertura dos itens e o produto com menor tempo de cobertura no momento do *setup* seja escolhido.

A quantidade a ser produzida de cada produto é definida por uma política do tipo *order-up-to*, na qual se produz visando a atingir o estoque máximo, S_i. Assim, a quantidade a ser produzida é dada pela diferença entre S_i e o estoque atual do produto i no momento em que

ele é selecionado para iniciar sua produção. Observe que, para um dado produto, caso ocorram pedidos enquanto o lote é produzido, o nível S_i não é atingido.

O módulo de produção mantém-se ativo até que não haja mais itens a serem produzidos, quando entra em repouso e aguarda novo sinal do módulo de vendas.

A fábrica funciona de forma contínua, operando durante 7 dias por semana e 24 horas por dia e a máquina tem uma taxa de produção média dada por μ, que é igual para todos os produtos.

Porém, considera-se que a taxa de produção efetiva da máquina é influenciada pela sua taxa de eficácia (OEE - *Overall Equipment Effectiveness*), representada pelo símbolo ρ. Esse indicador pode ser influenciada por fatores como tempo de *setup*, manutenção, quebras, qualidade dos produtos, entre outros.

$$\mu_{Efetiva} = \mu \cdot \rho \tag{4.2}$$

Note que, embora variável, a demanda $X_{i,t}$ para dado produto i no dia t é estacionária, ou seja, pode-se representar sua distribuição a partir de uma distribuição *log-normal* com média μ_i e desvio padrão σ_i, que são dependentes dos parâmetros d_i (demanda média do produto) e de um coeficiente de variação, cv, comum a todos os produtos:

$$X_{i,t} \sim Lognormal(\mu_i^{log-normal}, (\sigma_i^{log-normal})^2) \tag{4.3}$$

$$\mu_i^{log-normal} = ln\left(\frac{d_i}{\sqrt{1+cv^2}}\right) \tag{4.4}$$

$$\sigma_i^{log-normal} = \sqrt{ln(1+cv^2)} \tag{4.5}$$

$$\forall i \in i = 1, 2, ..., N$$

O tempo de processamento de um lote de produção na máquina segue uma distribuição *log-normal*, com os parâmetros da distribuição calculados de forma similar ao da distribuição das demandas dos produtos:

$$t^{prod} \sim Lognormal(\mu_{prod}^{log-normal}(\mu, cv), (\sigma_{prod}^{log-normal}(\mu, cv))^2) \tag{4.6}$$

Para iniciar a produção, considera-se que seja necessário realizar um *setup* na máquina com valor médio de t_0^{setup}. Assim como para o tempo de produção dos lotes, o tempo de *setup* segue uma distribuição *log-normal*:

$$t^{setup} \sim Lognormal(\mu_{setup}^{log-normal}(t_0^{setup}, cv), (\sigma_{setup}^{log-normal}(t_0^{setup}, cv))^2) \tag{4.7}$$

Por fim, ressalta-se que o desempenho do controle de estoque é medido pelo seu nível de serviço e pelo seu custo de estoque total anual, que considera custos de manutenção de estoque, custos de *setup* e custo de ruptura de estoque, os quais podem ser computados da seguinte forma:

$$THC = \sum_{t=1}^{T^{sim}} \left(\sum_{i=1}^{N} E_{i,t} \cdot hc \right) \cdot \frac{T^{ano}}{T^{sim}} \tag{4.8}$$

$$TLC = \sum_{t=1}^{T^{sim}} \left(\sum_{i=1}^{N} (D_{i,t} - D_{i,t}) \cdot lc \right) \cdot \frac{T^{ano}}{T^{sim}} \tag{4.9}$$

$$TSC = N_{setup} \cdot sc \cdot \frac{T^{ano}}{T^{sim}} \tag{4.10}$$

$$TIC = THC + TSC + TLC \tag{4.11}$$

$$SL = \frac{T_{vendas}}{T_{demanda}} \tag{4.12}$$

Onde:

T^{sim} : Total de dias simulados

T^{ano} : Total de dias de operação da empresa no ano

THC : Custo de manutenção de estoque anual (*Total Holding Cost*)

TLC : Custo de ruptura de estoque anual (*Total Lost Sales Cost*)

TSC : Custo de *setup* anual (*Total Setup Cost*)

N_{setup} : Número total de *setups* realizados no ano

TIC : Custo total de estoque anual (*Total Inventory Cost*)

X_{total} : Demanda total anual

Y_{total} : Número total de vendas anual

SL : Nível de serviço (*Service Level*)

Na Tabela 1, apresentam-se os valores para os parâmetros fixos do problema. Já na Tabela 2, reúnem-se a notação e a descrição das principais variáveis do problema.

Tabela 1: Parâmetros fixos do problema.

Variável	Descrição	Valor	Unidade
T^{ano}	Total de dias de operação da empresa no ano	250	Dias
t_0^{setup}	Tempo de *setup* base médio	1	Horas
hc	Custo diário unitário de manutenção do estoque	$0,1$	R$/dia/unid.
lc	Custo unitário de ruptura de estoque	40	R$/unid.
sc	Custo fixo de *setup*	250	R$/*setup*

Tabela 2: Variáveis e parâmetros do problema.

Variável	Descrição	Unidade
N	Número de itens produzidos	-
T^{sim}	Total de dias simulados	Dias
X_{total}	Demanda total anual	Unid.
Y_{total}	Número total de unidades vendidas no ano	Unid.
d_i	Demanda média diário do produto i	Unid./dia
cv	Coeficiente de variação	-
ρ	Taxa de eficácia da máquina	-
μ	Taxa de produção da máquina	Unid./dia
s_i	Estoque mínimo/Ponto de reabastecimento para o item i	Unid.
S_i	Estoque máximo o item i	Unid.
THC	Custo de manutenção de estoque anual	R$
TLC	Custo de ruptura de estoque anual	R$
TSC	Custo de *setup* anual	R$
TIC	Custo total de estoque anual	R$
SL	Nível de serviço	-

4.2 Modelo computacional

O modelo computacional foi implementado em dois blocos, o de vendas e de produção, conforme os respectivos módulos apresentados anteriormente na seção 4.1. Esses módulos foram implementados por meio de um modelo de simulação de eventos discretos com o uso da biblioteca *Simpy* do *Python*.

No Algoritmo 2 apresenta-se o pseudocódigo para o módulo de vendas de cada produto, que recebe os pedidos diariamente e verifica se é possível atender a demanda com completude, bem como se é necessário iniciar a produção de algum item.

De forma similar, no Algoritmo 3 apresenta-se o pseudocódigo para o módulo de produção implementado que verifica no *log* quais os produtos a serem fabricados, seleciona o SKU mais prioritário e simula o tempo de *setup* e de manufatura para o item.

Ao final da simulação, os resultados são apresentados ao usuário por meio de um gráfico de monitoramento do nível de estoque para os itens. Além disso, são exibidos os indicadores de desempenho computados durante a simulação, tais como os custos (TIC, THC, TLC e TSC) e o nível de serviço (SL), como mostrado na Figura 18. O exemplo da Figura 18 representa o caso particular em que $(s_i, S_i) = (s, S)\ \forall\ i = 0, 2, ..., 9$.

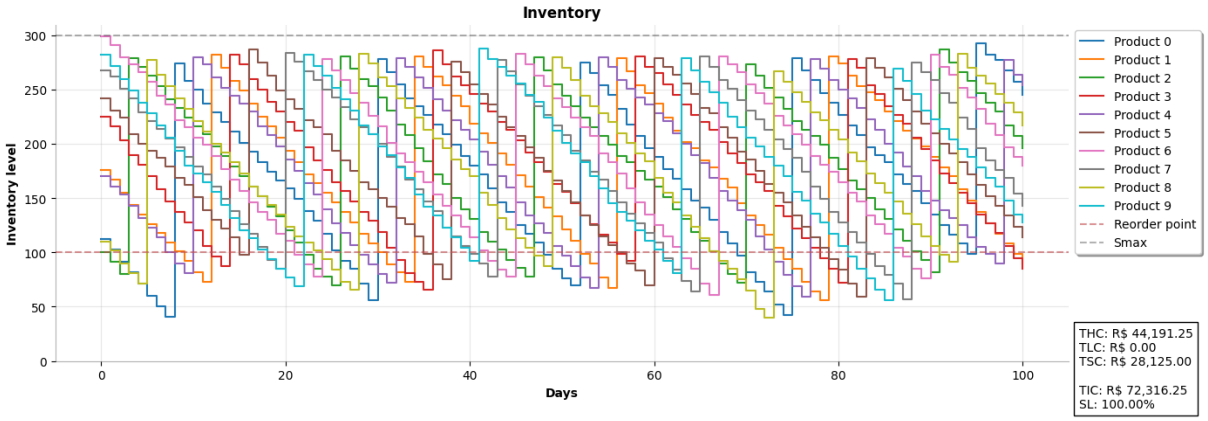
Algoritmo 2 Módulo de vendas

função MÓDULO DE VENDAS(env, id) ▷ Onde env é ambiente de simulação e id é o identificador do produto
  Enquanto VERDADEIRO **faça**
    $D_{id,t} \leftarrow$ DEMANDA(id) ▷ Computa a demanda estacionária diária
    $Q_{vendas} \leftarrow min(E_{id}, D_{id,t})$ ▷ Computa a quantidade vendida
    $E_{id} \leftarrow$ ATUALIZAR(Q_{vendas}) ▷ Atualiza o estoque
    Se $E_{id} \leq s_{id}$ **Então** ▷ Verifica se atingiu ponto de reabastecimento
      Se id **não está** em $Log_{\text{produção}}$ **Então** ▷ Verifica se produção já foi solicitada
        $Log_{\text{produção}} \leftarrow$ ADICIONAR(id)
      Fim Se
    Fim Se
    $T_{vendas} \leftarrow T_{vendas} + Q_{vendas}$ ▷ Atualiza o total de vendas
    $T_{demanda} \leftarrow T_{demanda} + D_{id,t}$ ▷ Atualiza o total de demanda
    $TLC \leftarrow T_{lc} + (D_{id,t} - Q_{vendas}) \cdot lc$ ▷ Atualiza o custo total de vendas perdidas
    Yield SIMULAR(24 horas) ▷ Executa todos os eventos das próximas 24 horas antes de reiniciar o *loop*
  Fim Enquanto
Fim função

Algoritmo 3 Módulo de produção

função MÓDULO DE PRODUÇÃO(env, $Log_{\text{produção}}$) ▷ Onde env é ambiente de simulação e $Log_{\text{produção}}$ é a lista de itens a produzir
    Enquanto VERDADEIRO **faça**
        Yield ATIVAÇÃO() ▷ Aguarda ativação do módulo de produção
        $id \leftarrow$ PRIORIZAR($Log_{\text{produção}}$) ▷ Seleciona o item a produzir
        $Q_{prod} \leftarrow S_{id} - E_{id}$
        Yield SETUP MÁQUINA(env, id) ▷ Simula o tempo de *setup*
        Yield PRODUZIR(env, id, Q_{prod}) ▷ Simula o tempo de produção do item
        Se TAMANHO($Log_{\text{produção}}$) **igual a** 0 **Então** ▷ Verifica se há mais itens a produzir
            $env \leftarrow$ REPOUSO PRODUÇÃO(env)
        Fim Se
    Fim Enquanto
Fim função

Figura 18: Exemplo de simulação para $N = 10$.



Fonte: Elaborado pelo autor.

4.3 Verificação e Validação

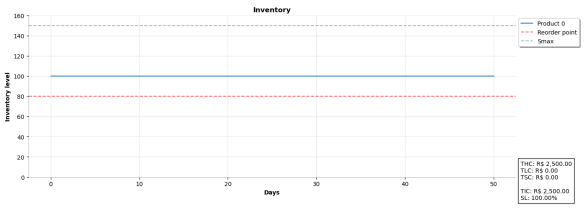
Com o modelo computacional implementado, é preciso fazer a verificação do modelo para atestar sua coerência e aderência ao problema a ser resolvido e, para tal, pode-se simular o funcionamento da fábrica com parâmetros determinísticos em um ambiente com baixa complexidade.

Para verificar a coerência dos resultados do modelo, foram realizados os seguintes testes:

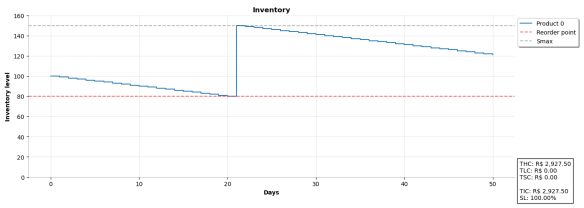
1. *Teste 1*: Um produto com demanda nula;

2. *Teste 2*: Um produto com demanda unitária determinística e taxa de produção muito maior do que a demanda diária;
3. *Teste 3*: Um produto com demanda determinística e taxa de produção igual à demanda diária;
4. *Teste 4*: Um produto com demanda determinística e ponto de reabastecimento nulo;
5. *Teste 5*: Dois produtos com demandas determinísticas distintas;
6. *Teste 6*: Dois produtos com demandas determinísticas iguais e pontos de reabastecimento distintos;
7. *Teste 7*: Três produtos com demandas determinísticas iguais e pontos de reabastecimento iguais.

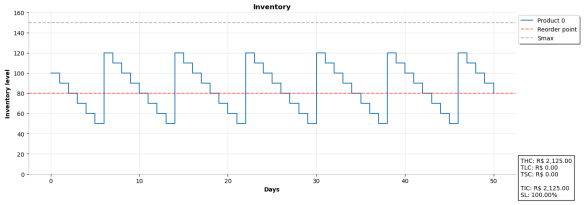
Figura 19: Testes 1 a 6 de verificação e validação do modelo de simulação.



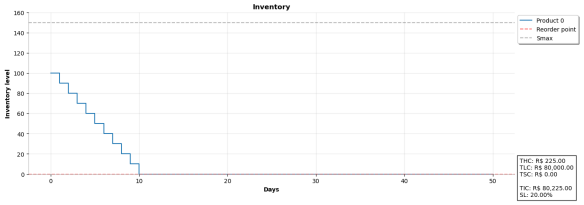
(a) Teste 1.



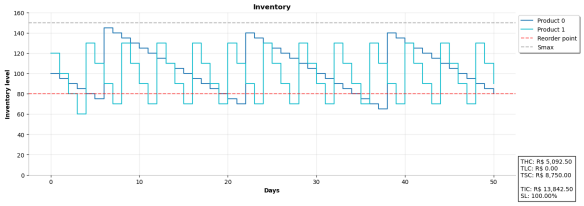
(b) Teste 2.



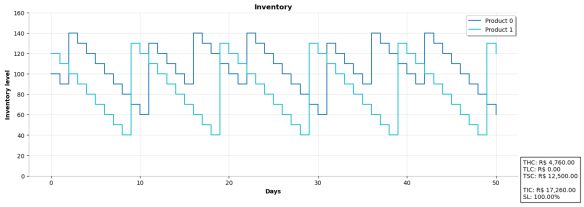
(c) Teste 3.



(d) Teste 4.



(e) Teste 5.



(f) Teste 6.

A seguir, apresenta-se em mais detalhe o último teste listado (Teste 7), que ilustra a validade do modelo no auxílio à resolução do problema explorado neste Trabalho.

Tabela 3: Parâmetros comuns aos cenários de exemplo do Teste 7.

Variável	**Valor**	**Unidade**
N	3	-
T^{sim}	50	Dias
d_i	10	Unid./dia
cv	0	-
ρ	100%	-
μ	30	Unid./dia
S_{max}	150	Unid.

Assim, para uma fábrica que produz 3 tipos de itens, define-se uma demanda diária comum a todos de 10 unid./dia/produto, conforme mostrado na Tabela 3, e parâmetros de estoque também comuns tais que $(s_i, S_i) = (s_{min}, S_{max}), \forall\, i = 0, 1, 2$.

Ao definir-se dois cenários, A e B, tais que os respectivos pontos de reabastecimento são $s^A_{min} = 100$ e $s^B_{min} = 50$, e assumindo que os estoques iniciais são iguais a S_{max}, é possível simular a performance do sistema e obter os resultados apresentados na Tabela 4 e nas Figuras 20 e 21.

Tabela 4: Comparação dos cenários A e B ($N = 3$) para o Teste 7.

	Cenário A	Cenário B
s_{min}	100	50
THC	R$ 6.420	R$ 4.900
TLC	R$ 0	R$ 38.000
TSC	R$ 18.750	R$ 10.000
TIC	R$ 25.170	R$ 52.900
SL	100%	87,33%

Ao analisar os resultados obtidos, percebe-se que eles são coerentes com o comportamento esperado. O cenário A obteve um custo de manutenção de estoque maior do que o cenário B, já que seu ponto de reabastecimento mais conservador leva a um nível de estoque médio maior.

Essa abordagem mais conservadora também foi importante para o nível de serviço apresentado pelos cenários, que foi de 100% para o cenário A (com $TLC_A =$ R$\$\ 0$) e 87,33% para o cenário B, que apresentou um custo de ruptura de estoque bem maior ($TLC_B =$ R$\$\ 38.000$).

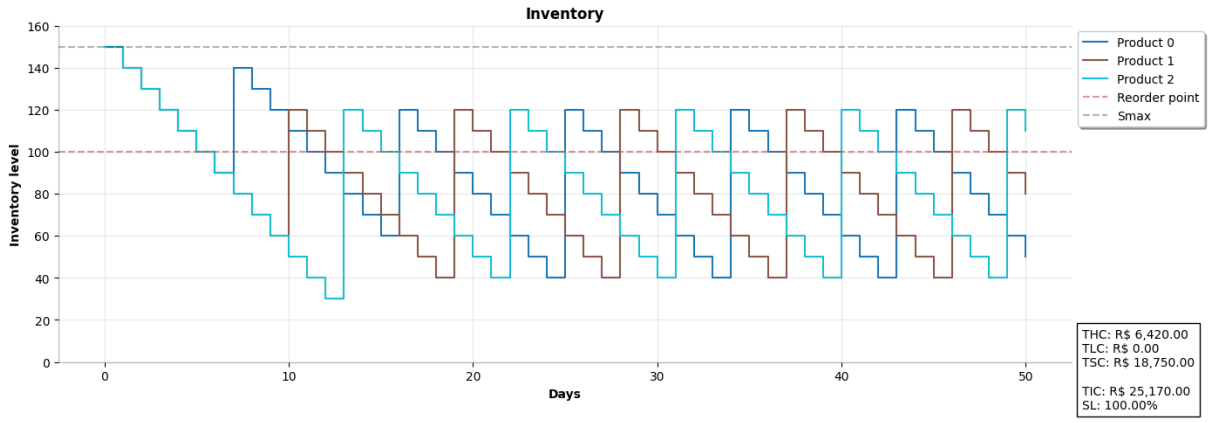
Com uma menor diferença entre os valores de S_{max} e s_{min}, mais *setups* são exigidos para o cenário A quando comparado ao cenário B, o que foi verificado pelo custo de *setup* computado

para as respectivas instâncias ($TLC_A =$ R\$ 18.750 e $TLC_B =$ R\$ 10.000).

Por fim, embora para certos empreendimentos o nível de serviço apresentado pelo cenário B seja satisfatório, observa-se que o *trade-off* entre o tamanho dos lotes e a performance da fábrica do cenário B não é atrativo, pois seu custo total final é maior do que para o cenário A ($TLC_A =$ R\$ 25.170 $<$ $TLC_B =$ R\$ 52.900).

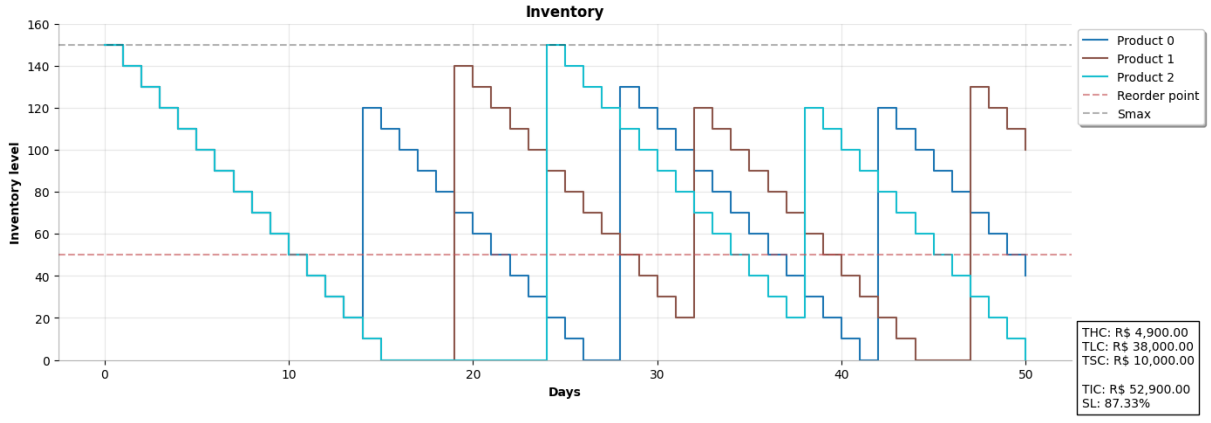
Portanto, por meio do exemplo apresentado, pode-se validar a utilidade do modelo na comparação de diferentes configurações de sistemas produtivos, o que auxiliará na identificação das melhores configurações de parâmetros de controle de estoque para uma fábrica, que é um dos objetivos deste Trabalho de Formatura.

Figura 20: Resultado obtido para o Cenário A.



Fonte: Elaborado pelo autor.

Figura 21: Resultado obtido para o Cenário B.



Fonte: Elaborado pelo autor.

5 MODELO DE SIMULAÇÃO-OTIMIZAÇÃO

Com o modelo de simulação verificado e validado, segue-se com a implementação dos métodos de busca em duas etapas: a implementação da estrutura global do método e a calibração dos hiperparâmetros dos algoritmos.

5.1 Experimentos de calibração

Na primeira fase da implementação dos métodos de busca, o foco principal estava na criação da estrutura fundamental dos métodos, incluindo a definição das principais funções conforme o modo de funcionamento dos métodos, como apresentado na seção 2.3.

Após a implementação inicial do método de busca, passou-se para a segunda etapa: a de calibração. Nessa fase, o objetivo era ajustar os valores dos hiperparâmetros de cada algoritmo para otimizar seus desempenhos em termos de eficiência e precisão.

A calibração é uma etapa crítica, pois é por meio dela que serão buscados os potenciais máximos de performance dos métodos de busca para o problema estudado. É importante ressaltar que a calibração de algoritmos não é uma tarefa trivial e pode ser tratada como um problema à parte por si só, devido à complexidade do processo e ao tempo que pode tomar (HAMADI; MONFROY; SAUBION, 2012).

A necessidade de adaptar os valores dos parâmetros às particularidades do problema e às restrições de execução dos experimentos impede a simples utilização de parâmetros indicados na literatura. Todavia, os valores da literatura ainda se mostram úteis para indicar ordens de grandeza e intervalos, como em Jin et al. (2019) para o algoritmo *Nelder-Mead*; ou relações de dependência entre os parâmetros, como indicado em Wong e Komarudin (2008) e Angelova e Pencheva (2011) para os algoritmos *Ant Colony* e *Genetic Algorithm*, respectivamente.

Além disso, é preciso aliar a intuição e o conhecimento prévio do problema para equilibrar as estratégias de "intensificação" e "diversificação" durante a busca por soluções.

A intensificação diz respeito à exploração com foco em regiões do espaço de busca que já foram observados. Assim, essa é uma estratégia que visa a aperfeiçoar as soluções atuais, buscando outros candidatos próximos àqueles que já foram identificadas como promissores.

Isso significa que o algoritmo prioriza a exploração de áreas onde as melhores soluções foram encontradas anteriormente.

A diversificação, por outro lado, visa a explorar áreas do espaço de busca que ainda não foram bem exploradas, mesmo que isso signifique sair das regiões onde boas soluções já foram encontradas. É uma estratégia que ajuda a evitar a convergência prematura e a garantir que o algoritmo continue procurando por soluções potencialmente melhores em diferentes partes do espaço de busca.

As principais técnicas de otimização de métodos de busca metaheurísticos encontradas na literatura podem ser classificadas em otimizações *online* e *offline*. As técnicas *online* são mais complexas e buscam adaptar os valores dos hiperparâmetros durante a execução do algoritmo para resolução de instâncias do problema. Já as técnicas *offline* são mais simples e buscam a configuração apropriada dos algoritmos antes da execução dos algoritmos e, tradicionalmente, esses métodos de otimização dos hiperparâmetros ocorrem via tentativa e erro (HAMADI; MONFROY; SAUBION, 2012).

Para este Trabalho, optou-se pela adoção da calibração *offline* por meio de um experimento fatorial completo para cada um dos três métodos de busca que possuem hiperparâmetros. Para cada algoritmo, buscou-se na literatura intervalos de valores esperados para os parâmetros e, a partir deles, definiram-se diferentes níveis para cada hiperparâmetro. Devido à diferença entre o número de parâmetros a serem calibrados em cada método de busca, buscou-se equilibrar o número de experimentos realizados para cada algoritmo, conforme ilustrado na Tabela 5.

Tabela 5: Tabela dos experimentos fatoriais de calibração.

Algoritmo	**Parâmetros**	**Tipo de experimento**	**Combinações**	**Minutos**
Nelder-Mead	4	4^k	256	2048
Genetic Algorithm	5	3^k	243	1944
Ant-Colony	8	2^k	256	2048

Note que o teste das configurações/combinações de parâmetros foi feito com base em 8 instâncias do problema estudado, para que se pudesse testar a performance dos parâmetros em diferentes circunstâncias. Assim, para cada conjunto de hiperparâmetros, os algoritmos buscaram soluções para os cenários apresentados na Tabela 6:

Tabela 6: Tabela dos cenários a serem testados por cada configuração de hiperparâmetros dos métodos de busca.

Cenário	N	p	cv	ρ	Tempo de execução
1	5	0	0,1	80%	60 s
2	5	0	0,5	80%	60 s
3	5	0,15	0,1	80%	60 s
4	5	0,15	0,5	80%	60 s
5	20	0	0,1	80%	60 s
6	20	0	0,5	80%	60 s
7	20	0,15	0,1	80%	60 s
8	20	0,15	0,5	80%	60 s

Na Tabela 6, N é o número de produtos, p é um fator de distribuição de demanda, cv é o coeficiente de variação e ρ é a taxa de eficácia da máquina, conforme descrito com mais detalhes no capítulo 6.

A cada algoritmo teve 60 segundos para buscar o melhor custo possível para cada cenário. Dessa forma, cada algoritmo foi calibrado durante aproximadamente 2.000 minutos ou 33 horas.

Ao final, para avaliar o desempenho da configuração de parâmetros, foram comparadas a soma dos custos obtidos nos 8 cenários, sendo considerada como a melhor configuração aquela com menor custo total.

A seguir, apresentam-se em mais detalhes os diferentes métodos de busca implementados para a resolução do problema estudado, assim como a calibração executada para cada um deles.

5.2 Busca aleatória

O método de busca aleatória é uma abordagem que se baseia na aleatoriedade para explorar e buscar soluções em um dado espaço de busca. Nesse método, o processo de busca não segue uma ordem determinística, mas sim uma estratégia guiada pelo acaso.

Ao utilizar o método de busca aleatória, o algoritmo seleciona uma solução candidata de forma aleatória dentro do espaço de busca e avalia seu desempenho. Em seguida, na próxima iteração, uma nova solução candidata é sorteada e avaliada, sem qualquer influência/dependência do resultado obtido na iteração anterior.

Se a nova solução candidata obtida for considerada melhor do que a solução anterior, ela é

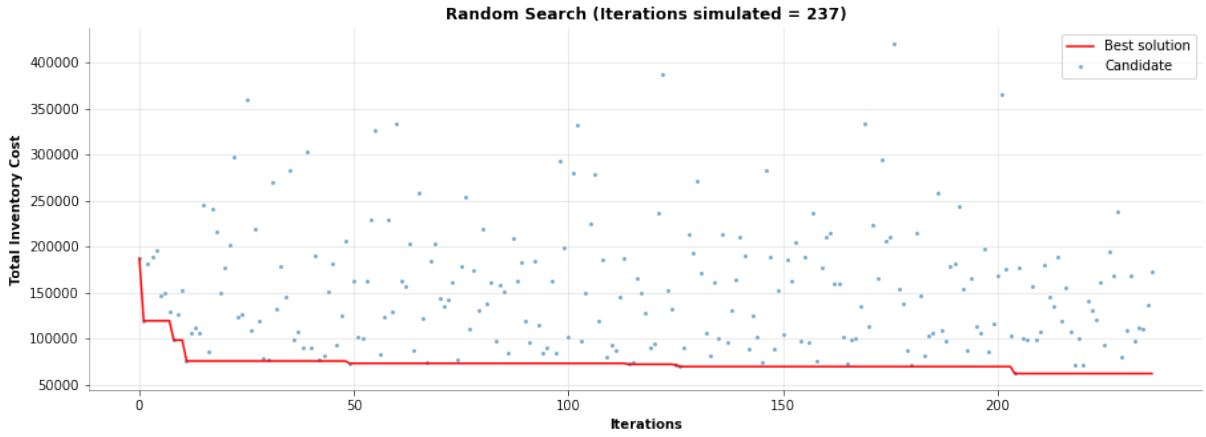
aceita e se torna a nova solução atual. Caso contrário, a solução atual permanece inalterada.

Uma das principais vantagens do método de busca aleatória é a sua capacidade de evitar a estagnação em ótimos locais, pois a aleatoriedade na escolha das soluções permite que sejam exploradas áreas diferentes do espaço de busca, aumentando a probabilidade de encontrar soluções melhores. Além disso, de forma prática, o método é simples de ser implementado.

No entanto, é importante ressaltar que o método de busca aleatória não trabalha com a noção de convergência para soluções ótimas, o que resulta em uma exploração incompleta de subespaços com melhor potencial. Devido a esse fato, o método torna-se sensível à dimensionalidade do problema, pois grandes problemas possuem espaços de busca mais complexos.

Na Figura 22, pode-se observar o comportamento do algoritmo descrito acima. Para um problema com 5 produtos, o algoritmo foi capaz de encontrar soluções melhores com o avanço das iterações. Porém, pela Figura 22, fica evidente que não há uma convergência nas soluções candidatas analisadas uma vez que há grande variação entre os custos computados.

Figura 22: Custo dos candidatos analisados em cada iteração da Busca aleatória (*Random Search)*.



Fonte: Elaborado pelo autor.

Em resumo, o método de busca aleatória é uma abordagem interessante para explorar espaços de busca em problemas em que a estrutura do espaço não é conhecida ou não pode ser facilmente explorada. Sua capacidade de evitar ótimos locais e sua simplicidade de implementação são características atrativas. No entanto, é necessário avaliar cuidadosamente a adequação desse método em relação às características do problema em questão.

No Algoritmo 4, representa-se o pseudocódigo usado para implementação da busca aleatória.

Algoritmo 4 Busca aleatória

```
1: função BUSCA ALEATÓRIA
2:     s_best ← None                                  ▷ Melhor solução
3:     f_best ← ∞                                     ▷ Custo da melhor solução
4:     continuar ← VERDADEIRO
5:     Enquanto continuar faça
6:         s ← SORTEAR()                              ▷ Gera um candidato aleatório
7:         f_s ← AVALIAR(s)                           ▷ Calcula o custo para o candidato s
8:         Se f_s < f_best Então
9:             f_best ← f_s
10:            s_best ← s
11:        Fim Se
12:        Se Critério de parada atendido Então
13:            continuar ← FALSO
14:        Fim Se
15:    Fim Enquanto
16:    Retorne s_best, f_best
17: Fim função
```

5.3 Nelder-Mead

Como visto na seção 2.3, o algoritmo de busca *Nelder-Mead* é um método de busca com grande potencial para auxiliar na resolução do problema estudado neste Trabalho de Formatura. Esse método analisa múltiplos pontos durante uma mesma iteração e busca a convergência para os candidatos com maior potencial (i.e. menor custo).

Sendo assim, seguiu-se com a implementação desse método de busca, adaptando-o para o contexto deste estudo. No Algoritmo 5, descreve-se o pseudocódigo para o código implementado:

Algoritmo 5 *Nelder-Mead*

função NELDER-MEAD
    $\alpha \leftarrow \alpha(constante)$ ▷ Fator de Reflexão
    $\beta \leftarrow \beta(constante)$ ▷ Fator de Expansão
    $\gamma \leftarrow \gamma(constante)$ ▷ Fator de Contração
    $\rho \leftarrow \rho(constante)$ ▷ Fator de Encolhimento
    $Continuar \leftarrow TRUE$ ▷ Critério de parada
    $S_0 \leftarrow SIMPLEXINICIAL()$ ▷ Onde S_0 é o *Simplex* inicial com $N + 1$ vértices criado aleatoriamente
    $S \leftarrow S_0$ ▷ *Simplex* atual
    Enquanto $Continuar$ **faça**
        $L_{cost} \leftarrow$ ORDENAR(S) ▷ Calcula os custos de cada ponto de S e ordena em ordem crescente em uma lista
        $f_l, P_l \leftarrow L_{cost}[0], P(L_{cost}[0])$ ▷ Determina o ponto de melhor custo
        $f_h, P_h \leftarrow L_{cost}[-1], P(L_{cost}[-1])$ ▷ Determina o ponto de pior custo
        $f_s, P_s \leftarrow L_{cost}[-2], P(L_{cost}[-2])$ ▷ Determina o ponto de segundo pior custo
        $P_c \leftarrow$ CENTROIDE(S)
        $P_r \leftarrow$ RESTRINGIR(REFLETIR(P_c, P_h, α))
        Se $f_l \leq f(P_r)$ **E** $f(P_r) < f_s$ **Então** ▷ Caso #1
            $S \leftarrow$ NOVO SIMPLEX(S, P_h, P_r)
        Senão Se $f(P_r) < f_l$ **Então** ▷ Caso #2
            $P_e \leftarrow$ RESTRINGIR(EXPANDIR(P_c, P_r, β))
            Se $f(P_e) < f(P_r)$ **Então** ▷ Caso #2.1
                $S \leftarrow$ NOVO SIMPLEX(S, P_h, P_e)
            Senão ▷ Caso #2.2
                $S \leftarrow$ NOVO SIMPLEX(S, P_h, P_r)
            Fim Se
        Senão Se $f(P_r) \geq f_s$ **Então** ▷ Caso #3
            $P_{Cont,1}, P_{Cont,2} \leftarrow$ RESTRINGIR(CONTRAIR(P_c, P_h, P_r, γ))
            $P_{Cont,final} \leftarrow$ MENOR CUSTO($P_{Cont,1}, P_{Cont,2}$)
            Se $f(P_{Cont,final}) < f_h$ **Então** ▷ Caso #3.1
                $S \leftarrow$ NOVO SIMPLEX(S, P_h, $P_{Cont,final}$)
            Senão ▷ Caso #3.2
                $S \leftarrow$ ENCOLHER(S, P_l)
            Fim Se

33: **Senão** ▷ Caso #4
34: $S \leftarrow$ RESTRINGIR(ENCOLHER(S, P_l))
35: **Fim Se**
36: **Se** PARAR(S) **Então**: ▷ Verifica se critérios de parada são respeitados
37: $Continuar \leftarrow FALSE$
38: **Fim Se**
39: **Fim Enquanto**
40: $L_{cost} \leftarrow$ ORDENAR(S)
41: $f_{best}, P_{best} \leftarrow L_{cost}[0], P(L_{cost}[0])$
42: **Imprima** f_{best}, P_{best}
43: **Fim função**

O código implementado segue a estrutura indicada na literatura. Porém, para garantir que os novos candidatos gerados nos processos de reflexão, expansão, encolhimento e contração sejam candidatos viáveis, uma adaptação foi feita ao código adicionando a função `RESTRINGIR`, que garante que, para cada candidato, todos os pares (s_i, S_i) sigam os seguintes critérios:

$$\begin{cases} s_i, S_i \geq 0 & \forall i = 1, 2, ..., N \\ s_i \leq S_i & \forall i = 1, 2, ..., N \end{cases}$$

Assim, adotou-se como premissa que caso s_i ou S_i fossem negativos, seus valores seriam substituídos por 0 e caso $s_i > S_i$, o valor de s_i seria substituído de forma que $s_i = S_i$.

De forma a obter melhores desempenhos com o algoritmo, é essencial ajustar seus hiperparâmetros para se adequarem ao problema específico em estudo. Neste Trabalho, aborda-se o processo de calibração dos parâmetros do *Nelder-Mead* focando na adaptação dos 4 fatores $\alpha,\ \beta,\ \gamma$ e ρ, que dizem respeito aos fatores de reflexão, expansão, contração e encolhimento.

A calibração dos hiperparâmetros foi realizada por meio de experimento fatorial 4^k completo, conforme explicado anteriormente. Para cada parâmetro, foram estabelecidos diferentes níveis, tendo como referência os valores apresentados na literatura por Jin et al. (2019). Um resumo dos valores testados no experimento de calibração é apresentado na Tabela 7.

Tabela 7: Plano de Experimento fatorial 4^k de calibração do Algoritmo de *Nelder-Mead*.

Parâmetro	**Descrição**	**Nível 1**	**Nível 2**	**Nível 3**	**Nível 4**
α	Fator de Reflexão	0,5	1	1,5	2
β	Fator de Expansão	0,5	1	1,5	2
γ	Fator de Contração	0,5	1	1,5	2
ρ	Fator de Encolhimento	0,5	1	1,5	2

Com base no experimento fatorial definido, 256 configurações de parâmetros foram testadas, conforme apresentado na Tabela 8:

Tabela 8: Resultado dos Experimentos de calibração para o *Nelder-Mead*.

Rank	α	β	γ	ρ	**Custo**
1	1,0	0,5	0,5	1,5	896.373,7
2	1,0	2,0	0,5	0,5	899.543,8
3	1,5	2,0	0,5	1,0	903.970,1
4	2,0	0,5	0,5	1,5	908.802,7
5	1,5	2,0	2,0	0,5	913.935,5
...					
252	2,0	0,5	1,5	2,0	1.424.901,6
253	2,0	0,5	2,0	1,5	1.454.313,3
254	2,0	2,0	1,0	2,0	1.455.548,4
255	1,5	0,5	1,5	1,5	1.458.122,7
256	1,5	1,0	1,0	1,5	1.465.053,6

Dessa forma, os melhores valores para os hiperparâmetros obtidos após a calibração são apresentados na Tabela 9:

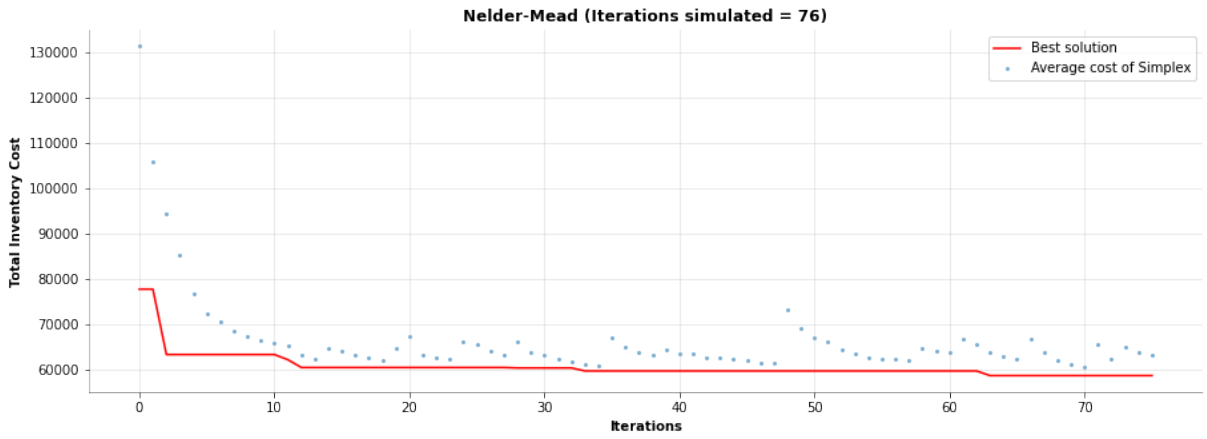
Tabela 9: Parâmetros calibrados do Algoritmo de *Nelder-Mead*.

Parâmetro	**Descrição**	**Valor**
α	Fator de Reflexão	1,0
β	Fator de Expansão	0,5
γ	Fator de Contração	0,5
ρ	Fator de Encolhimento	1,5

Finalmente, na Figura 23 é possível ver um exemplo de resultado obtido com este método

de busca para o mesmo caso de 5 produtos testado na busca aleatória. É possível notar que a média do custo dos candidatos analisados tende a convergir para valores próximos ao da melhor solução encontrada, o que não era observado com o método de busca aleatória. Assim, percebe-se que, no algoritmo de *Nelder-Mead*, prioriza-se a intensificação em relação a diversificação.

Figura 23: Custo médio do Simplex em cada iteração do *Nelder-Mead*.



Fonte: Elaborado pelo autor.

5.4 Genetic Algorithm

O terceiro método de busca implementado foi o *Genetic Algorithm* (Algoritmo Genético), uma abordagem do tipo populacional que examina múltiplas soluções em cada geração (iteração) e procura convergir para os indivíduos com maior aptidão (ou seja, menor custo), inspirando-se nos mecanismos de seleção natural e troca genética. No Algoritmo 6, descreve-se o pseudocódigo para o código implementado.

Da mesma forma que o Algoritmo *Nelder-Mead*, o Algoritmo Genético foi calibrado de forma a obter melhor performance no problema tratado neste Trabalho.

Para adaptar o Algoritmo Genético ao problema em questão, foram ajustados 5 parâmetros: o tamanho da população (N_{pop}); o número de indivíduos sobreviventes (N_{keep}) ou a taxa de sobrevivência (P_{keep}); o número de pontos de cruzamento ($N_{crossover}$) ou a taxa de *crossover* ($P_{crossover}$); a taxa de mutação (τ) e a taxa de contribuição dos pais (β) na prole. Esses parâmetros desempenham um papel crucial no equilíbrio entre diversidade de exploração da busca, bem como na convergência do algoritmo.

Novamente, o processo de calibração dos parâmetros ocorreu por meio de um experimento fatorial completo. Dessa vez, um experimento fatorial 3^k foi realizado e os possíveis níveis para cada um dos hiperparâmetros foram definidos com base nos valores apresentados por Angelova

e Pencheva (2011). O plano dos experimentos de calibração é apresentado na Tabela 10.

Tabela 10: Plano do Experimento fatorial 3^k de calibração do Algoritmo Genético.

Parâmetro	**Descrição**	**Nível 1**	**Nível 2**	**Nível 3**
N_{pop}	População total	10	15	20
P_{keep}	Taxa de sobrevivência	30%	50%	70%
$P_{crossover}$	Taxa de *crossover*	50%	70%	100%
τ	Taxa de mutação	1%	5%	10%
β	Taxa de contribuição dos pais	50%	70%	100%

Em seguida, os resultados das 243 configurações de parâmetros testadas são apresentados na Tabela 11.

Tabela 11: Resultados dos Experimentos de calibração para o Algoritmo Genético.

Rank	N_{pop}	τ	P_{keep}	β	$P_{crossover}$	Custo
1	20	0,01	0,7	0,5	1,0	1.053.974,2
2	10	0,10	0,5	0,7	0,7	1.055.028,9
3	10	0,05	0,5	0,5	0,7	1.055.926,4
4	15	0,01	0,5	0,7	1,0	1.059.304,3
5	15	0,05	0,7	0,5	0,7	1.076.928,7
...						
239	10	0,01	0,5	1,0	0,5	1.322.820,2
240	10	0,05	0,3	1,0	0,5	1.336.739,9
241	10	0,05	0,3	1,0	0,7	1.346.779,8
242	15	0,10	0,3	1,0	1,0	1.354.014,7
243	10	0,01	0,3	1,0	0,7	1.362.867,0

Os valores para os hiperparâmetros calibrados do Algoritmo Genético podem ser encontrados na Tabela 12, em que N é o número de variáveis do problema.

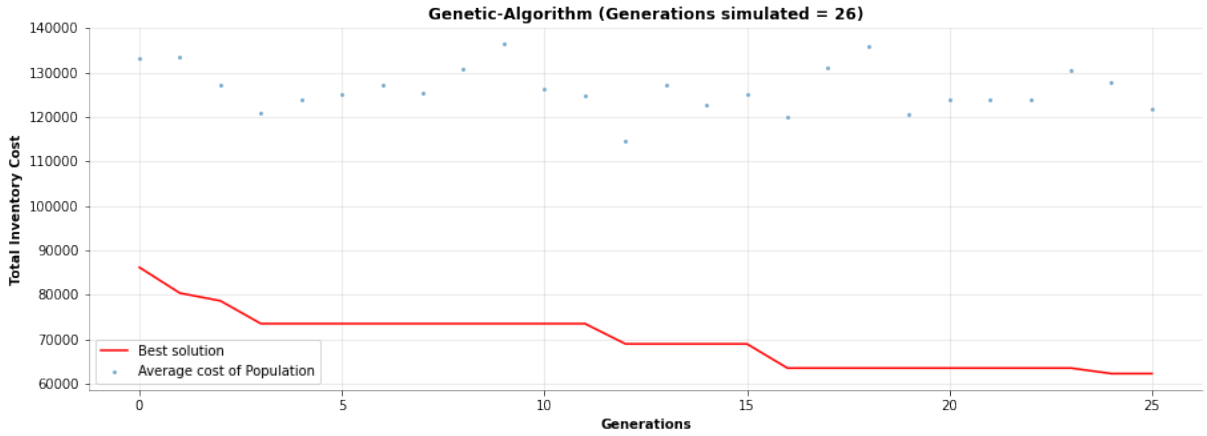
Tabela 12: Parâmetros calibrados do Algoritmo Genético.

Parâmetro	Descrição	Valor
N_{pop}	População total	20
N_{keep}	População sobrevivente	14
$N_{crossover}$	Pontos de cruzamento	N
τ	Taxa de mutação	1%
β	Taxa de contribuição dos pais	50%

Algoritmo 6 Algoritmo Genético

função ALGORITMO GENÉTICO(N_{var}) ▷ Onde N_{var} é o número de variáveis do problema
    $N_{pop} \leftarrow N_{pop}(constante)$ ▷ Tamanho da população
    $N_{keep} \leftarrow N_{keep}(constante)$ ▷ # Indivíduos que sobrevivem por geração
    $\beta \leftarrow \beta(constante)$ ▷ Fator de contribuição
    $\tau \leftarrow \tau(constante)$ ▷ Taxa de mutação
    $N_{crossover} \leftarrow N_{crossover}(constante)$ ▷ Pontos de cruzamento
    $S_0 \leftarrow$ POPULAÇÃO INICIAL(N_{pop}, N_{var}) ▷ Cria a população inicial
    $Continuar \leftarrow TRUE$ ▷ Critério de parada
    $S \leftarrow S_0$
    Enquanto $Continuar$ **faça**
        $L_{cost} \leftarrow$ ORDENAR(S)
        $S_{Sobreviventes} \leftarrow$ SELEÇÃO NATURAL(N_{keep}, L_{cost}, S)
        $S_{Descendentes} \leftarrow$ ACASALAMENTO($N_{pop}, N_{keep}, S_{Sobreviventes}, \beta$)
        $S \leftarrow S_{Sobreviventes} + S_{Descendentes}$
        $S \leftarrow$ MUTAÇÃO(S, τ)
        Se PARAR(S) **Então** ▷ Verifica se critérios de parada são respeitados
            $Continuar \leftarrow FALSE$
        Fim Se
    Fim Enquanto
    $L_{cost} \leftarrow$ ORDENAR(S)
    $f_{best}, Cromossomo_{best} \leftarrow L_{cost}[0], S[L_{cost}[0]]$
    Imprima $f_{best}, Cromossomo_{best}$
Fim função

Figura 24: Custo médio da população em cada iteração do Algoritmo Genético.



Fonte: Elaborado pelo autor.

Finalmente, na Figura 24 é possível ver o resultado obtido com este método de busca para o exemplo de 5 produtos testado com os outros métodos de busca. É possível notar que a média do custo da população, neste exemplo, apresentou oscilação e uma queda lenta em seu valor, mas a melhor solução encontrada pelo algoritmo apresentou evolução com o avanço das iterações.

5.5 Ant Colony

O quarto e último algoritmo de busca implementado foi o *Ant Colony*. Assim como o *Genetic Algorithm*, esse método adota uma abordagem populacional, usando populações de formigas para analisar múltiplos caminhos (soluções candidatas) durante uma iteração, e busca a convergência para as soluções com maior potencial (ou menor custo).

Assim, prosseguiu-se com a implementação deste método de busca, adaptando-o para se adequar ao contexto deste estudo conforme apresentado no Algoritmo 7, que descreve o pseudocódigo para o *Ant Colony Optimization* implementado.

Assim como nos outros algoritmos, uma calibração dos hiperparâmetros foi realizada. Porém, além do parâmetros próprios do algoritmo, foram testados diferentes valores para a densidade da malha para discretização do problema. Como explicado anteriormente, o algoritmo *Ant Colony* utiliza a estrutura de um grafo durante a otimização. Assim, para problemas como o estudado, no qual o espaço de soluções é contínuo, é preciso estabelecer uma discretização do espaço.

Malhas mais densas permitem uma exploração mais detalhada do espaço de soluções, porém, com o aumento da complexidade do problema (i.e. número de variáveis e tamanho

do intervalo de possíveis valores), a densidade do grafo que representa o problema aumenta exponencialmente, o que prejudica a performance do algoritmo. Assim, para este problema, adotou-se como densidade da malha o valor de 5 unidades, logo, para dado intervalo de valores discretizado, dois possíveis valores para a solução estarão separados por 5 unidades.

Algoritmo 7 *Ant Colony Optimization*

função ACO(G) ▷ Onde G é o grafo do problema
  $N_{pop} \leftarrow N_{pop}(constante)$ ▷ Tamanho da população
  $\alpha \leftarrow \alpha(constante)$ ▷ Influência do feromônio na escolha
  $\beta \leftarrow \beta(constante)$ ▷ Influência da visibilidade na escolha
  $\rho \leftarrow \rho(constante)$ ▷ Taxa de evaporação
  $Q \leftarrow Q(constante)$ ▷ Quantidade de feromônios adicionada por formiga
  $\tau_0 \leftarrow \tau_0(constante)$ ▷ Quantidade inicial de feromônios
  $\tau_{min}, \tau_{max} \leftarrow \tau_{min}, \tau_{max}(constantes)$ ▷ Quantidades mínima e máxima de feromônios
  $N_{update} \leftarrow N_{update}(constante)$ ▷ N melhores formigas a adicionarem feromônios
  $f_{best} \leftarrow \infty$
  $Caminho_{best} \leftarrow None$
  $Continuar \leftarrow TRUE$ ▷ Critério de parada
  Enquanto $Continuar$ **faça**
    $L_{caminhos} \leftarrow list()$
    Para Formiga em N_{pop} **faça**
      $Caminho_{formiga} \leftarrow$ ESCOLHER CAMINHO(G, α, β)
      $L_{caminhos} \leftarrow$ ARMAZENAR($Caminho_{formiga}$)
    Fim Para
    $L_{cost} \leftarrow$ ORDENAR($L_{caminhos}$)
    Se $L_{cost}[0] < f_{best}$ **Então** ▷ Atualiza o melhor caminho
      $f_{best} = L_{cost}[0]$
      $Caminho_{best} \leftarrow L_{caminhos}[L_{cost}[0]]$
    Fim Se
    $G \leftarrow$ ATUALIZAR FEROMÔNIOS($G, L_{caminhos}, \rho, Q, N_{update}, \tau_{min}, \tau_{max}$)
    Se PARAR($Caminho_{best}, f_{best}$) **Então** ▷ Verifica critérios de parada
      $Continuar \leftarrow FALSE$
    Fim Se
  Fim Enquanto
  Imprima $f_{best}, Caminho_{best}$
Fim função

Assim como realizado com os outros algoritmos, uma calibração do algoritmo *Ant Colony* foi executada, dessa vez por meio de um experimento fatorial completo 2^k. Wong e Komarudin (2008) apresentam algumas referências de valores para os principais parâmetros do algoritmo, o que permitiu a definição dos valores mais apropriados para os níveis de possíveis valores para os hiperparâmetros do método de busca, conforme ilustrado na Tabela 13:

Tabela 13: Plano do Experimento fatorial 2^k de calibração do Algoritmo *Ant Colony*.

Parâmetro	**Descrição**	**Nível 1**	**Nível 2**
N_{pop}	População total	10	20
α	Influência do feromônio	1	2
β	Influência da visibilidade	0	1
ρ	Taxa de evaporação	0,25	0,5
Q	Quantidade de feromônio adicionada por formiga	1	2
N_{update}	Melhores formigas que poderão adicionar feromônios	25%	50%
τ_{min}	Quantidade mínima de feromônio por aresta	0,05	1
τ_{max}	Quantidade máxima de feromônio por aresta	20	30

Os resultados dos 256 experimentos realizados com diferentes configurações de parâmetros são apresentados na Tabela 14:

Tabela 14: Resultados dos Experimentos de calibração para o Algoritmo *Ant Colony*.

Rank	N_{pop}	α	β	ρ	Q	τ_{min}	τ_{max}	N_{update}	Custo
1	10	2	0	0,5	2	0,05	20	0,25	1.011.512,8
2	10	2	0	0,5	2	0,05	30	0,25	1.019.816,2
3	20	2	1	0,25	1	1,0	20	0,25	1.035.834,0
4	20	1	1	0,25	2	0,05	30	0,50	1.050.036,6
5	20	2	0	0,25	2	0,05	20	0,25	1.053.231,6
...									
252	10	1	1	0,25	2	1,0	30	0,50	1.180.912,6
253	10	2	1	0,5	1	1,0	30	0,50	1.181.282,8
254	10	1	1	0,5	2	0,05	20	0,25	1.196.696,0
255	20	2	1	0,2	2	0,05	20	0,50	1.199.525,0
256	10	2	0	0,25	1	1,0	20	0,25	1.202.177,0

Sendo assim, os valores encontrados após o fim da calibração podem ser observados na Tabela 15:

Tabela 15: Parâmetros calibrados do Algoritmo *Ant Colony*.

Parâmetro	Descrição	Valor
N_{pop}	População total	10
α	Influência do feromônio	2
β	Influência da visibilidade	0
ρ	Taxa de evaporação	0,5
Q	Quantidade de feromônio adicionada por formiga	2
N_{update}	N melhores formigas que poderão adicionar feromônios	2
τ_{min}	Quantidade mínima de feromônio por aresta	0,05
τ_{max}	Quantidade máxima de feromônio por aresta	20

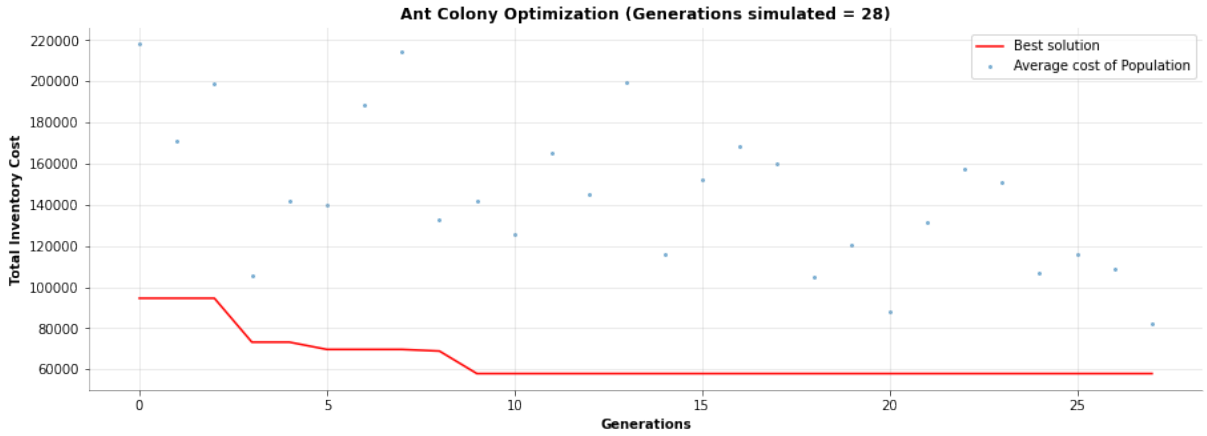
Note ainda que, para a implementação e calibração do ACO, foi adotada uma técnica alternativa de atualização da quantidade de feromônios dos caminhos percorridos pelas formigas, conhecida na literatura como MMAS (*MAX-MIN Ant System*).

Uma das principais vantagens do MMAS é a sua capacidade de encontrar soluções de alta qualidade de forma consistente. Isso se deve ao fato de que o MMAS utiliza uma estratégia de atualização de feromônio que limita a quantidade de feromônio depositada pelas formigas no percurso (τ_{min} e τ_{max}). Esse controle rigoroso ajuda a evitar a convergência prematura para soluções subótimas, permitindo que o algoritmo explore continuamente o espaço de busca em busca de soluções melhores.

Além disso, determinou-se uma quantidade máxima de formigas que podem adicionar feromônios a cada geração (N_{update}). Dessa forma, são priorizados apenas os melhores trechos para acelerar a convergência da população de formigas para espaços com maior potencial de sucesso.

Na Figura 25, verifica-se o resultado obtido com o método de busca *Ant Colony Optimization* para o exemplo de 5 produtos testado anteriormente. É possível notar que a média do custo da população apresentou oscilações durante sua queda gradativa que acompanhou a queda do custo da melhor solução encontrada pelo algoritmo.

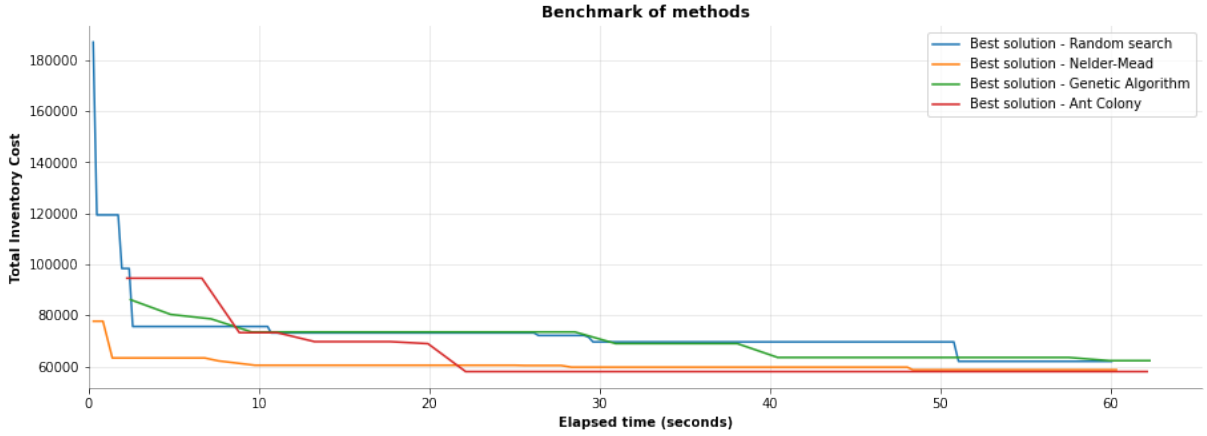
Figura 25: Custo médio da população em cada iteração do *Ant Colony*.



Fonte: Elaborado pelo autor.

Por fim, na Figura 26, apresenta-se de forma consolidada os resultados de todos os algoritmos implementados considerando uma mesma instância de problema. Nota-se que os algoritmos apresentam o comportamento esperado, com diminuição do valor da melhor solução encontrada com o avanço do tempo. Além disso, a performance dos algoritmos atinge níveis semelhantes após 60 segundos de execução.

Figura 26: Exemplo comparativo dos métodos de busca implementados.



Fonte: Elaborado pelo autor.

Portanto, com os algoritmos de busca implementados e calibrados, é possível seguir para a etapa de Experimentos de Comparação, que será apresentada no capítulo 6, no qual será descrita a metodologia utilizada para avaliar e comparar o desempenho dos algoritmos.

6 PLANEJAMENTO DOS EXPERIMENTOS DE COMPARAÇÃO

Com o modelo de simulação-otimização implementado e calibrado, é possível iniciar a etapa dos experimentos de comparação. Nessa parte, será estudado o desempenho dos 4 métodos de busca implementados sob diferentes condições de estresse da linha de produção e de complexidade do problema.

Para identificar o método numérico mais eficiente para a otimização dos parâmetros de controle de estoque para o *SELSP*, serão realizados diferentes experimentos seguindo a metodologia do *Design of Experiments* (DoE). Essa abordagem permitirá a realização de uma avaliação comparativa entre os algoritmos de otimização de forma sistemática e estatística.

A metodologia do DoE começa pela identificação das variáveis de interesse que podem afetar o desempenho dos algoritmos. No contexto em questão, essas variáveis podem incluir o número de produtos (N), a variabilidade da sistema (cv), a taxa de eficácia da máquina (ρ) e a distribuição da demanda (p).

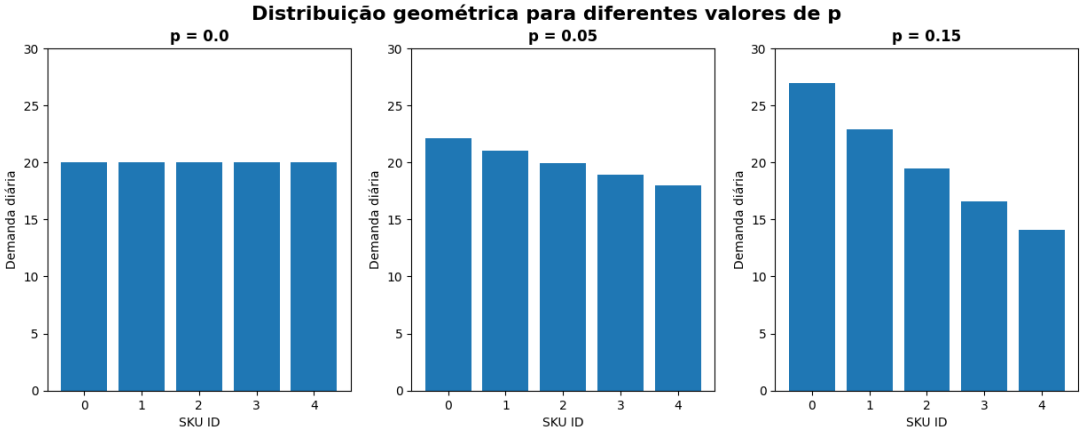
Uma vez identificados os fatores, é necessário definir os níveis em que cada variável será testada. Na Tabela 16, são apresentados os fatores que serão variados durante os experimentos e seus respectivos valores/níveis:

Tabela 16: Tabela resumo dos fatores do DoE e seus níveis.

Fator	**Descrição**	**Níveis**
N	Número de produtos	5; 10; 20
cv	Coeficiente de variação	10%; 25%; 50%
ρ	Taxa de eficácia da máquina	70%; 80%; 90%
p	Distribuição da demanda (geométrica)	0; 0,05; 0,15

Note que, para a distribuição da demanda, foi adotada uma distribuição geométrica. Essa abordagem permite o controle dos diferentes níveis de distribuição com apenas um único fator, que representa a probabilidade de sucesso (p) característica dessa distribuição. Assim, quanto maior o valor de p, menos uniforme é a distribuição da demanda entre os N produtos do problema, conforme ilustrado na Figura 27:

Figura 27: Distribuição geométrica para diferentes valores de p.



Fonte: Elaborado pelo autor.

Além das variáveis apresentadas na Tabela 16, que adotarão diferentes valores durante os experimentos, são estabelecidos outros parâmetros, constantes ou semi-variáveis, que também são importantes para a definição do problema, conforme apresentado na Tabela 17:

Tabela 17: Tabela resumo das variáveis usadas nos Experimentos de Comparação.

Parâmetro	**Descrição**	**Tipo**	**Valor**
Mod	Estratégia de sequenciamento	Constante	*LDS*
CMU	Margem de contribuição unitária	Constante	R$ 40/produto
TCM	Margem de contribuição total anual	Constante	R$ 1.000.000/ano
t^{setup}	Tempo de *setup*	Constante	1 h/*setup*
sc	Custo de *setup*	Constante	R$ 250/*setup*
lc	Custo de vendas perdidas	Constante	R$ 40/*unid.*
hc	Custo de armazenagem unitário	Constante	R$ 0,1/*unid.*/dia
T^{sim}	Número de dias simulados	Constante	100 dias
$T_{demanda}$	Demanda total anual	Semi-variável	25.000 unid./ano
t^{prod}	Tempo de processamento médio	Semi-variável	X h/unid.

O próximo passo do DoE é a geração de experimentos, em que cada experimento representa uma combinação específica de níveis das variáveis. Tendo em vista o baixo custo para a realização de cada um dos experimentos, optou-se pela realização de um experimento fatorial completo com 3 repetições, totalizando 972 ensaios, sendo 324 ensaios únicos (81 para cada um dos 4 métodos de busca), conforme apresentado na Tabela 18.

Além disso, para a realização dos experimentos, determinou-se que o espaço de solução

explorado seria limitado a valores entre 0 e 1.000 para quaisquer variáveis. Todos os algoritmos tiveram um tempo máximo de execução de 60 segundos para cada um dos ensaios.

Portanto, no contexto deste estudo, o DoE será uma ferramenta essencial para avaliar o desempenho dos algoritmos de otimização (*Random Search* - RD, *Nelder-Mead* - NM, *Genetic Algorithm* - GA e *Ant Colony* - ACO) diante das variações nos parâmetros do problema, contribuindo para uma escolha mais informada e eficaz do algoritmo a ser utilizado em diferentes situações de otimização.

Tabela 18: Tabela resumo do plano de Experimentos de Comparação.

Ordem Padrão	**Ordem dos Ensaios**	**N**	**cv**	ρ	**p**	**Algoritmo**
470	1	5	0,2	0,7	0,0	NM
395	2	20	0,2	0,9	0,1	GA
376	3	20	0,2	0,8	0,0	ACO
312	4	10	0,5	0,8	0,1	ACO
968	5	10	0,5	0,9	0,1	ACO
64	6	20	0,2	0,9	0,0	ACO
465	7	5	0,1	0,9	0,1	RD
292	8	10	0,5	0,7	0,0	ACO
708	9	20	0,2	0,8	0,1	ACO
231	10	10	0,1	0,8	0,0	GA
...						
725	963	20	0,5	0,7	0,1	RD
212	964	5	0,5	0,9	0,1	ACO
460	965	5	0,1	0,9	0,0	ACO
886	966	10	0,1	0,8	0,1	NM
150	967	5	0,2	0,7	0,1	NM
10	968	20	0,1	0,7	0,1	NM
524	969	5	0,5	0,8	0,1	ACO
732	970	20	0,5	0,7	0,1	ACO
533	971	5	0,5	0,9	0,1	RD
972	972	10	0,5	0,9	0,1	ACO

Os resultados dos experimentos de comparação serão analisados estatisticamente no capítulo 7 para determinar como as diferentes variáveis afetam o desempenho dos algoritmos e identificar qual algoritmo apresenta melhor desempenho na resolução do problema de otimização dos parâmetros de estoque no contexto do *Stochastic Economic Lot Scheduling Pro-*

blem (SELSP).

7 DISCUSSÃO DOS RESULTADOS

Neste capítulo, apresentam-se os resultados dos 972 experimentos do DoE de comparação proposto no capítulo 6. Os experimentos foram executados em um computador pessoal com processador M1 com arquitetura ARM de 8 núcleos e 16 GB de RAM com frequência de 4.266 MHz.

A Tabela 20 exibe uma amostra dos resultados obtidos para cada ensaio do DoE, onde são apresentados os parâmetros que definem a instância do problema; as variáveis resposta - Custo de estoque total (TIC) e Nível de serviço (SL); e as variáveis de decisão (s_i, S_i), que representam os parâmetros de estoque para cada produto.

Ao analisar-se os efeitos do número de produtos no custo de estoque (Figura 28 e Tabela 19), é possível notar que há uma correlação positiva entre o custo e o número de produtos, possivelmente pela maior necessidade de estocagem de itens.

É interessante notar também que há uma maior dispersão entre os resultados para o caso de $N = 5$ e esse comportamento é ocasionado, possivelmente, pela maior intensidade de demanda dos produtos (i.e. rateio da demanda total em poucos produtos). Essa característica pode tornar mais complexa a busca por níveis ótimos de estoque que equilibrem custos de armazenamento baixos e minimização da penalização por vendas perdidas.

Essa hipótese é reforçada quando se observa a Tabela 19, na qual o nível de serviço médio para soluções com 5 produtos é o menor, indicando maiores custos com vendas perdidas.

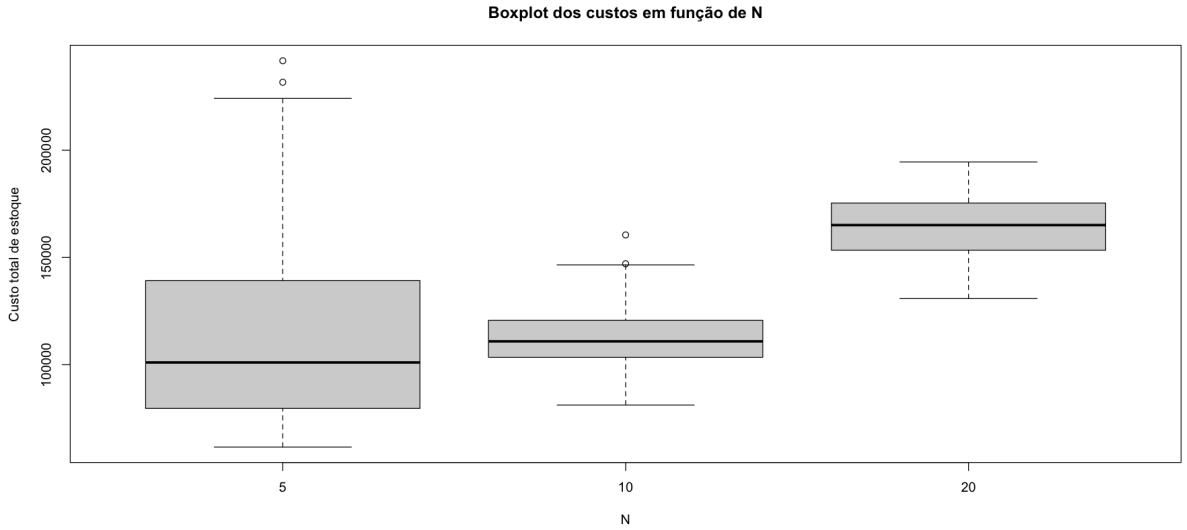
Tabela 19: Média e desvio padrão dos resultados para diferentes níveis de N.

	TIC		**SL**	
N	Média	Desv. pad.	Média	Desv. pad.
5	113.443,56	41.486,46	95,27	4,74
10	112.458,95	13.878,13	98,54	1,25
20	164.414,18	14.511,26	98,83	1,03

Tabela 20: Resultado dos experimentos do DoE.

OrdemPad	OrdemEns	N	cv	ρ	p	Algoritmo	TIC	SL	(s, S)
470	1	5	0,2	0,7	0,0	NM	103.489,4	99,1	(1.201,4; 1.399,2), (914,7; 1.658,8), (639,6; 1...
395	2	20	0,2	0,9	0,1	GA	169.395,4	99,5	(10,6; 19,1), (44,2; 647,5), (136,9; 517,6), (1...
376	3	20	0,2	0,8	0,0	ACO	160.969,9	98,7	(60,0; 270,0), (155,0; 215,0), (560,0; 635,0), ...
312	4	10	0,5	0,8	0,1	ACO	100.176,4	99,3	(175,0; 365,0), (260,0; 455,0), (585,0; 715,0),...
968	5	10	0,5	0,9	0,1	ACO	105.675,8	99,7	(435,0; 825,0), (120,0; 490,0), (655,0; 935,0),...
64	6	20	0,2	0,9	0,0	ACO	178.758,1	99,6	(535,0; 700,0), (15,0; 105,0), (35,0; 155,0), (...
465	7	5	0,1	0,9	0,1	RD	71.023,1	99,6	(376,4; 781,6), (849,3; 928,3), (603,9; 630,8),...
292	8	10	0,5	0,7	0,0	ACO	134.373,5	96,9	(95,0; 195,0), (810,0; 835,0), (895,0; 915,0), ...
708	9	20	0,2	0,8	0,1	ACO	162.261,0	96,5	(330,0; 480,0), (165,0; 340,0), (295,0; 365,0),...
231	10	10	0,1	0,8	0,0	GA	109.521,2	98,1	(355,6; 421,0), (67,3; 115,5), (202,3; 381,7), ...
417	11	20	0,5	0,8	0,1	RD	175.688,9	97,2	(229,6; 745,5), (133,9; 304,8), (68,6; 999,9), ...
624	12	10	0,5	0,7	0,1	ACO	134.581,1	95,3	(395,0; 560,0), (430,0; 640,0), (50,0; 280,0), ...
414	13	20	0,5	0,8	0,1	NM	139.206,4	100,0	(268,7; 489,6), (214,7; 380,5), (324,9; 505,6),...
861	14	5	0,5	0,9	0,1	RD	82.533,4	99,2	(237,9; 430,9), (156,1; 348,4), (904,4; 958,2),...
5	15	20	0,1	0,7	0,1	RD	153.345,0	99,6	(675,3; 953,4), (211,1; 394,6), (267,6; 505,8),...
...									
610	958	10	0,2	0,9	0,1	NM	85.176,4	99,8	(183,3; 483,9), (314,1; 657,7), (277,5; 549,5),...
542	959	10	0,1	0,7	0,0	NM	113.270,7	98,8	(136,2; 489,8), (223,2; 422,4), (553,1; 656,7),...
253	960	10	0,2	0,7	0,0	RD	126.196,0	97,4	(518,2; 749,4), (668,8; 914,1), (78,4; 279,2), ...
914	961	10	0,2	0,8	0,0	NM	107.816,9	98,0	(174,6; 520,1), (445,4; 688,6), (498,8; 649,1),...
161	962	5	0,2	0,8	0,1	RD	93.021,3	97,2	(585,6; 802,3), (223,8; 539,3), (784,3; 797,7),...
725	963	20	0,5	0,7	0,1	RD	174.823,8	97,5	(11,7; 164,4), (160,1; 268,0), (312,6; 579,1), ...
212	964	5	0,5	0,9	0,1	ACO	84.695,7	98,8	(90,0; 290,0), (355,0; 595,0), (250,0; 255,0), ...
460	965	5	0,1	0,9	0,0	ACO	66.688,9	99,9	(305,0; 795,0), (615,0; 670,0), (220,0; 705,0),...
886	966	10	0,1	0,8	0,1	NM	97.171,1	99,7	(481,3; 904,1), (283,1; 283,1), (223,4; 636,2),...
150	967	5	0,2	0,7	0,1	NM	146.411,3	94,8	(1.985,1; 2.081,7), (692,2; 692,2), (883,6; 883...
10	968	20	0,1	0,7	0,1	NM	147.620,4	99,5	(300,4; 522,2), (388,1; 626,4), (380,9; 602,8),...
524	969	5	0,5	0,8	0,1	ACO	87.810,4	97,6	(320,0; 760,0), (480,0; 555,0), (795,0; 930,0),...
732	970	20	0,5	0,7	0,1	ACO	189.504,2	99,1	(730,0; 850,0), (440,0; 980,0), (245,0; 395,0),...
533	971	5	0,5	0,9	0,1	RD	72.704,1	98,6	(375,3; 626,8), (499,9; 650,0), (145,8; 296,0),...
972	972	10	0,5	0,9	0,1	ACO	115.019,3	99,5	(415,0; 890,0), (490,0; 610,0), (390,0; 595,0),...

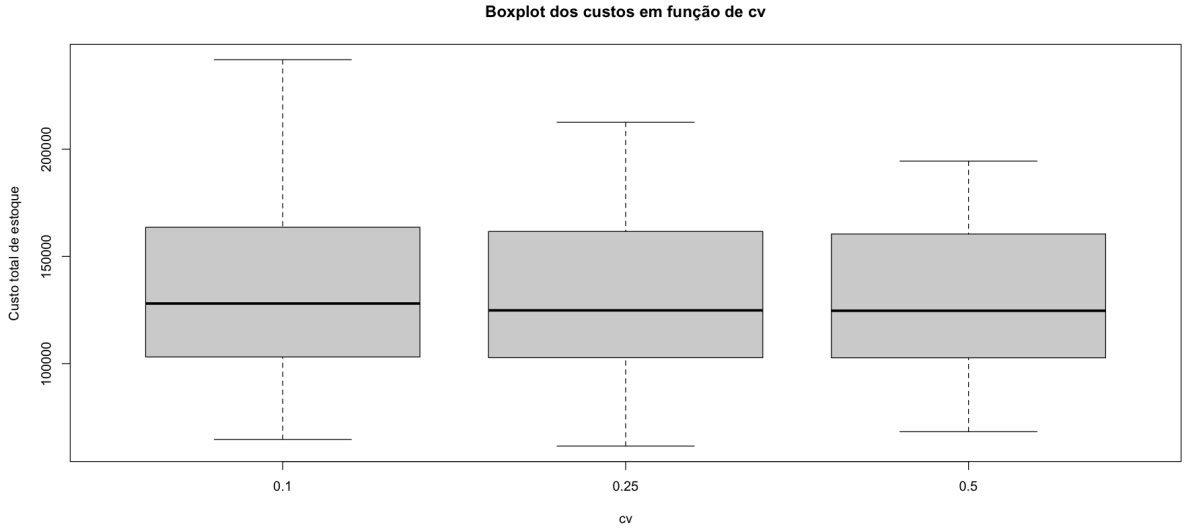
Figura 28: Boxplots com dispersão das amostras para os diferentes níveis de N.



Fonte: Elaborado pelo autor.

Em seguida, analisa-se o efeito de cv (coeficiente de variação) no custo. Por meio da Figura 29 e da Tabela 21, percebe-se que as soluções encontradas tendem a ter comportamentos semelhantes, indicando uma baixa influência desse fator no custo das soluções. Ao mesmo tempo, isso revela uma alta robustez do modelo de simulação-otimização para encontrar soluções mesmo em cenários de maior incerteza operacional.

Figura 29: Boxplots com dispersão das amostras para os diferentes níveis de cv.



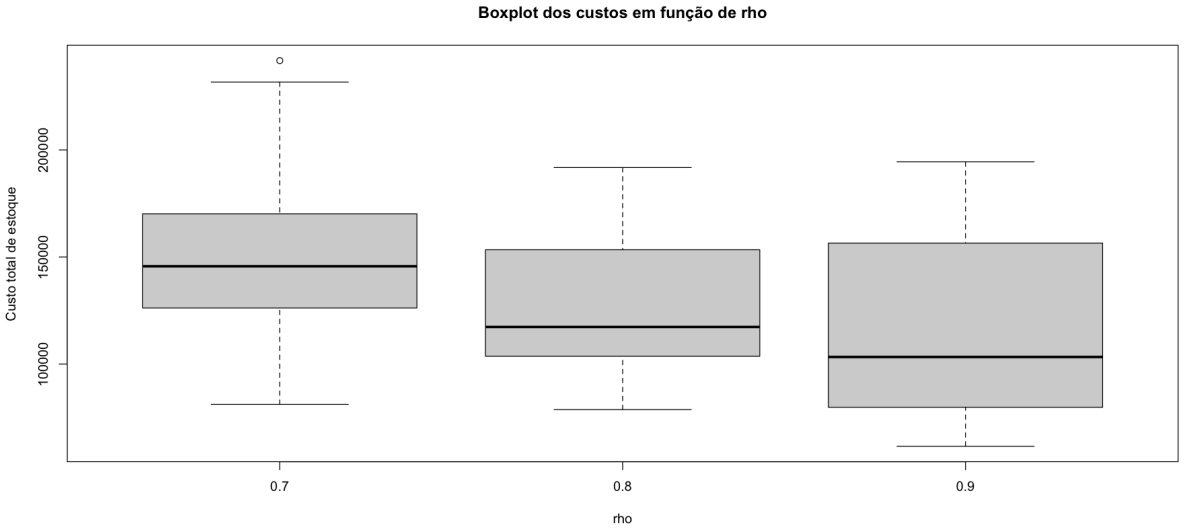
Fonte: Elaborado pelo autor.

Tabela 21: Média e desvio padrão dos resultados para diferentes níveis de cv.

	TIC		SL	
cv	Média	Desv. pad.	Média	Desv. pad.
0,10	131.018,56	38.131,75	97,41	3,83
0,25	129.191,77	35.346,70	97,53	3,33
0,50	130.106,36	34.507,78	97,70	2,67

De forma similar, analisa-se o impacto de ρ no custo das soluções. Sendo esse fator equivalente ao nível de eficácia da máquina, é natural esperar uma correlação negativa entre a taxa de eficácia da máquina e o custo, uma vez que uma menor eficácia implica, em geral, em maiores níveis de estoque devido à menor confiabilidade do sistema. Esse comportamento é confirmado pelos resultados obtidos, conforme apresentado pela Figura 30 e pela Tabela 22.

Figura 30: Boxplots com dispersão das amostras para os diferentes níveis de ρ.



Fonte: Elaborado pelo autor.

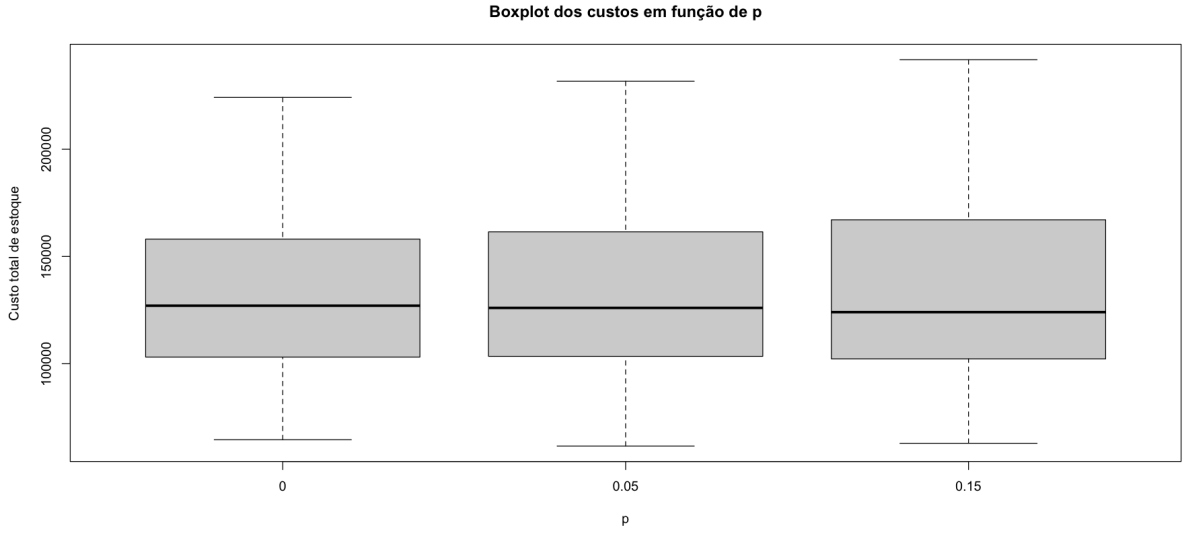
Tabela 22: Média e desvio padrão dos resultados para diferentes níveis de ρ.

	TIC		SL	
ρ	Média	Desv. pad.	Média	Desv. pad.
0,7	148.600,31	28.440,41	95,73	4,62
0,8	126.623,52	29.031,93	97,84	2,26
0,9	115.092,85	40.777,61	99,07	0,86

Já o fator p apresenta um comportamento similar ao do fator cv, com uma baixa influência

aparente da heterogeneidade das demandas dos SKUs no custo de estoque total, conforme ilustrado na Figura 31 e na Tabela 23. Esse fato reforça a capacidade do modelo construído de obter soluções para diferentes condições operacionais de demanda.

Figura 31: Boxplots com dispersão das amostras para os diferentes níveis de p.



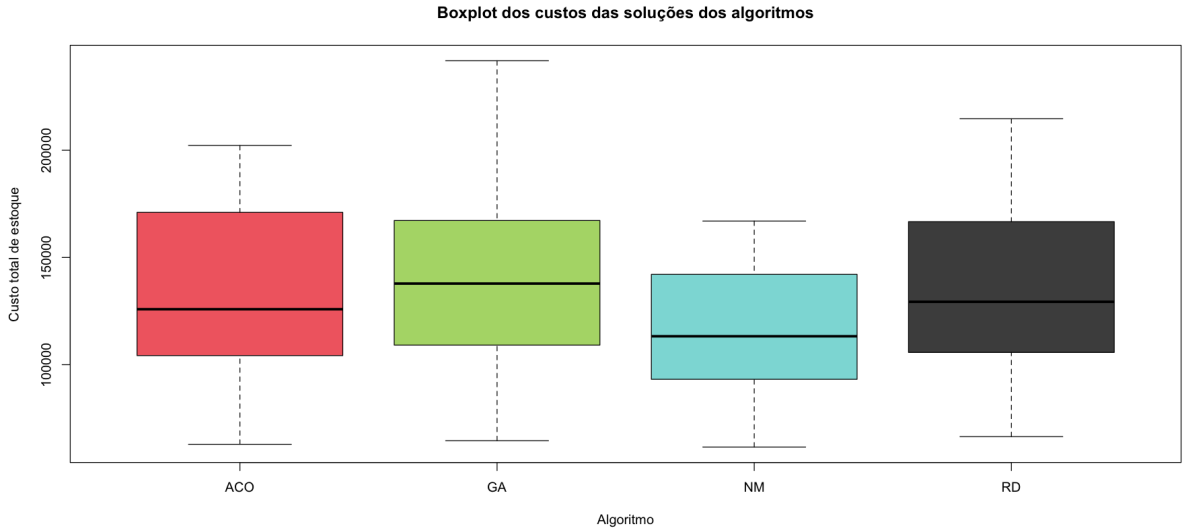
Fonte: Elaborado pelo autor.

Tabela 23: Média e desvio padrão dos resultados para diferentes níveis de p.

	TIC		**SL**	
p	Média	Desv. pad.	Média	Desv. pad.
0,00	128.612,01	34.563,72	97,65	3,24
0,05	129.988,41	35.364,88	97,50	3,43
0,15	131.716,26	38.022,37	97,49	3,27

Finalmente, pela Figura 32 e pela Tabela 24, analisam-se os impactos dos métodos de busca implementados nos custos das soluções. É possível perceber que o método de busca *Nelder-Mead* encontrou soluções melhores do que os outros métodos, além de apresentar uma menor dispersão entre as amostras. Já os outros três métodos (Busca aleatória, *Genetic Algorithm* e *Ant-Colony*) apresentaram performances semelhantes, sendo o *Genetic Algorithm* o método de busca com a performance geral menos interessante.

Figura 32: Boxplots com dispersão das amostras para os métodos de busca.



Fonte: Elaborado pelo autor.

Tabela 24: Média e desvio padrão dos resultados para diferentes métodos de busca.

	TIC		**SL**	
Método	Média	Desv. pad.	Média	Desv. pad.
ACO	133.173,14	37.525,93	97,48	3,07
GA	138.236,79	37.815,27	96,56	4,10
NM	115.424,01	27.708,92	98,91	1,62
RD	133.588,31	35.908,79	97,24	3,49

De forma a verificar e mensurar a influência dos diferentes fatores testados no custo total de estoque para o problema estudado, uma Análise de Variância foi realizada e é apresentada na Tabela 25. Ainda, na Equação 7.1, é representado o modelo linear proposto para realizar a Análise de Variância citada:

$$\begin{aligned} TIC = {} & n + cv + \rho + p + \text{alg} + n:cv + n:\rho + n:p + n:\text{alg} \\ & + cv:\rho + cv:p + cv:\text{alg} + \rho:p + \rho:\text{alg} + p:\text{alg} \end{aligned} \tag{7.1}$$

Note que para um nível de significância de $\alpha = 5\%$, todos os efeitos principais testados são significativos para o custo, com exceção do fator cv. Ainda, considerando esses fatores, os principais efeitos são os dos fatores N, ρ e dos algoritmos de busca, sendo a distribuição da demanda um fator com efeito menos intenso, conforme havia sido observado anteriormente.

É importante ressaltar que a Tabela 25 indica que o emprego de diferentes métodos de busca traz diferenças estatisticamente relevantes (mesmo para níveis de significância extremamente baixos: $\alpha < 2 \times 10^{-16}$) para o custo total de estoque. Sendo assim, analisa-se a seguir o impacto dos algoritmos para diferentes instâncias do SELSP.

Tabela 25: Análise de Variância para o Custo Total de estoque.

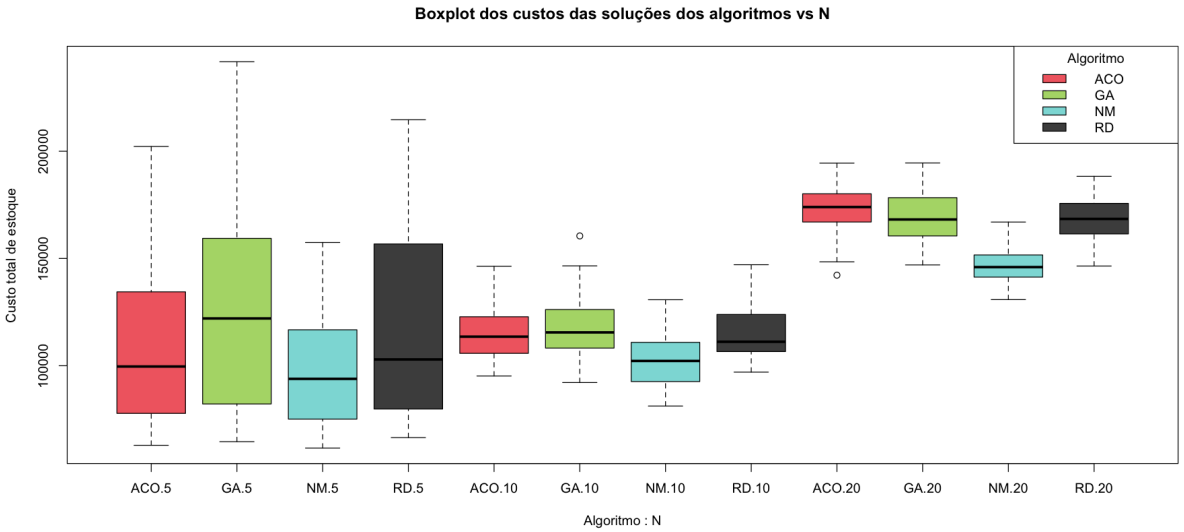
Fator	**Grau de liberdade**	**Soma de quadrados**	**Quadrados médios**	**F *value***	**Pr(>F)**
n	2	$5,722 \times 10^{11}$	$2,861 \times 10^{11}$	$1697,925$	$< 2 \times 10^{-16}$ ***
cv	2	$5,406 \times 10^{8}$	$2,703 \times 10^{8}$	1,604	0,201623
rho	2	$1,878 \times 10^{11}$	$9,389 \times 10^{10}$	$557,188$	$< 2 \times 10^{-16}$ ***
p	2	$1,568 \times 10^{9}$	$7,839 \times 10^{8}$	4,652	0,009770 **
alg	3	$7,368 \times 10^{10}$	$2,456 \times 10^{10}$	$145,750$	$< 2 \times 10^{-16}$ ***
$n : cv$	4	$1,308 \times 10^{10}$	$3,270 \times 10^{9}$	$19,406$	$2,45 \times 10^{-15}$ ***
$n : rho$	4	$2,148 \times 10^{11}$	$5,370 \times 10^{10}$	$318,681$	$< 2 \times 10^{-16}$ ***
$n : p$	4	$3,781 \times 10^{9}$	$9,453 \times 10^{8}$	5,610	0,000184 ***
$n : alg$	6	$1,231 \times 10^{10}$	$2,051 \times 10^{9}$	$12,171$	$3,38 \times 10^{-13}$ ***
$cv : rho$	4	$4,769 \times 10^{9}$	$1,192 \times 10^{9}$	$7,076$	$1,31 \times 10^{-05}$ ***
$cv : p$	4	$2,496 \times 10^{8}$	$6,240 \times 10^{7}$	0,370	0,829880
$cv : alg$	6	$1,583 \times 10^{9}$	$2,639 \times 10^{8}$	1,566	0,153916
$rho : p$	4	$1,177 \times 10^{9}$	$2,942 \times 10^{8}$	1,746	0,137820
$rho : alg$	6	$1,595 \times 10^{10}$	$2,658 \times 10^{9}$	$15,773$	$< 2 \times 10^{-16}$ ***
$p : alg$	6	$1,219 \times 10^{9}$	$2,032 \times 10^{8}$	1,206	0,300744
Resíduos	912	$1,537 \times 10^{11}$	$1,685 \times 10^{8}$		

Notas: Nível de signifância: 0 '***' 0,001 '**' 0,01 '*' 0,05 '.' 0,1 ' ' 1

Na Figura 33, comparam-se os resultados obtidos pelos diferentes métodos de busca para as interações de segundo grau com os demais fatores testados no DoE (N, cv, ρ e p). Ao observar as Figuras 33b, 33c e 33d, de forma geral, a diferença relativa de performances dos algoritmos é a mesma para as diferentes instâncias do problema, sendo o algoritmo *Nelder-Mead* consistentemente o método que atinge menores custos entre os quatro.

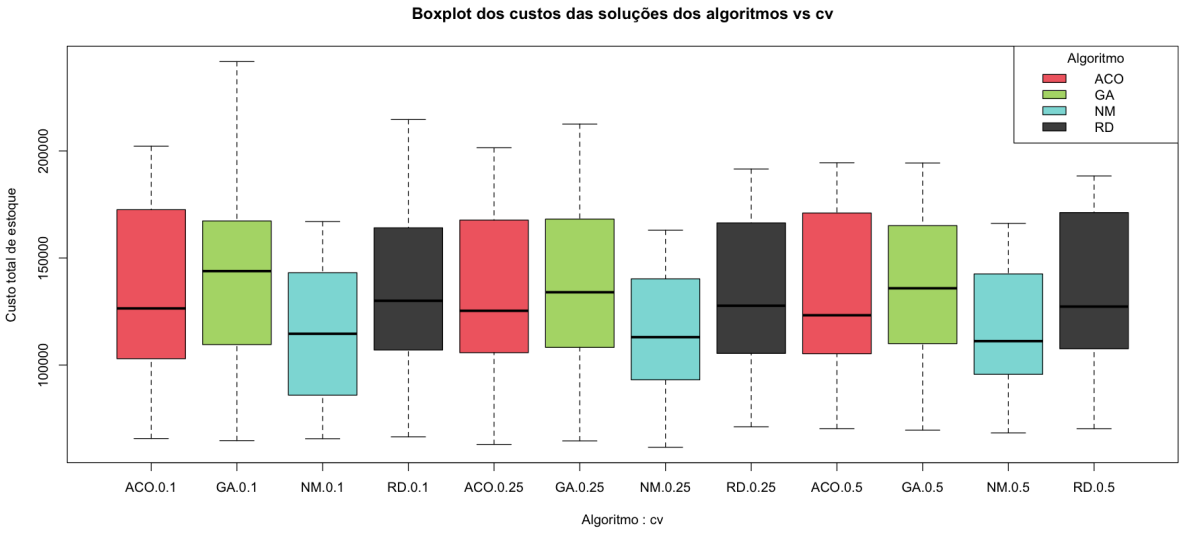
Figura 33: Boxplots com dispersão das amostras para os métodos de busca para diferentes configurações de (a) N, (b) cv, (c) ρ e (d) p.

(a) Boxplots com dispersão das amostras para os métodos de busca *versus* N.



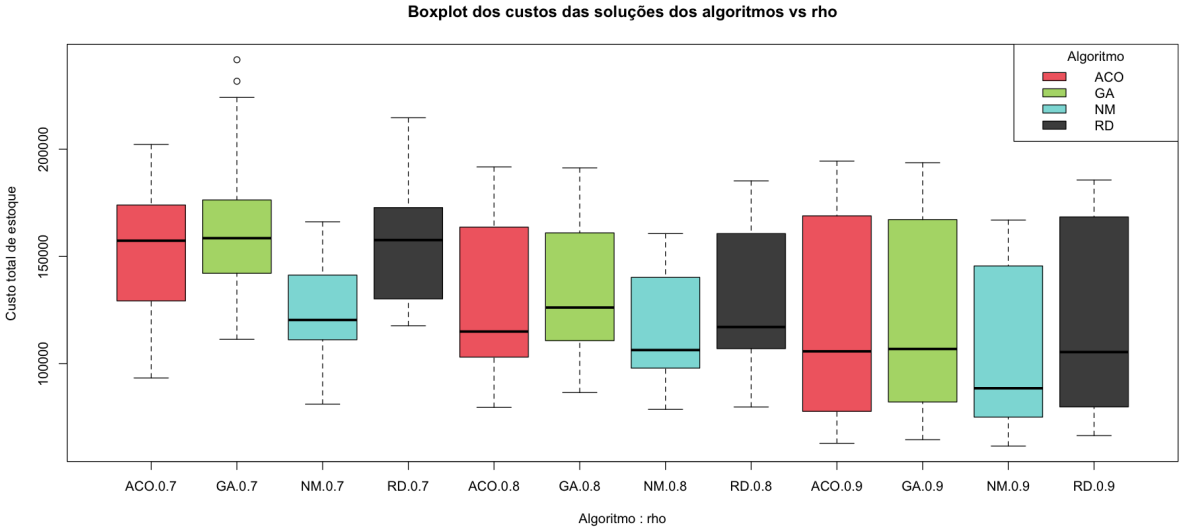
Fonte: Elaborado pelo autor.

(b) Boxplots com dispersão das amostras para os métodos de busca *versus* cv.



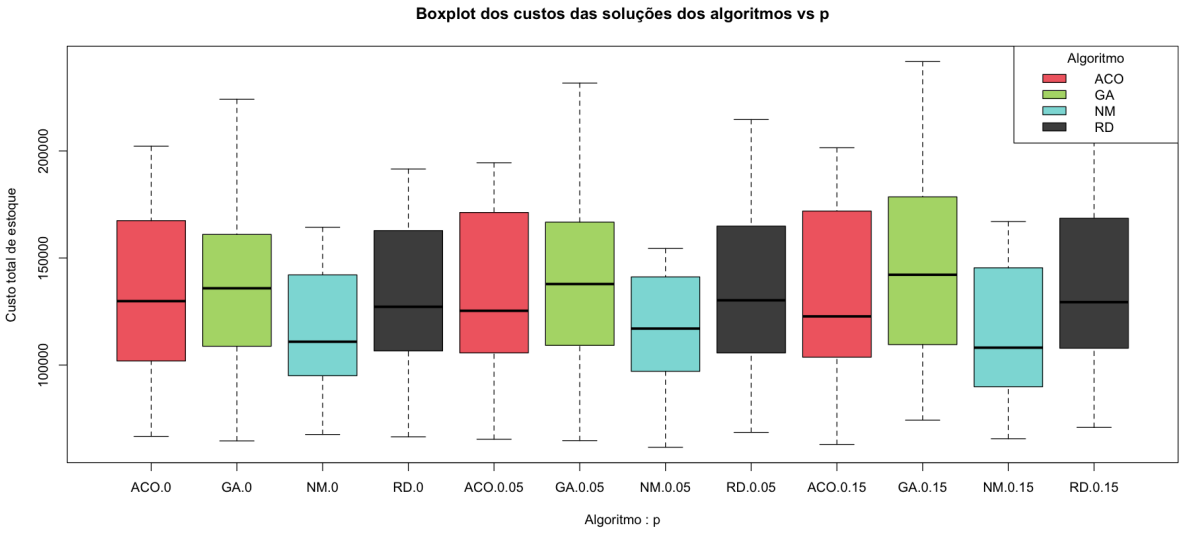
Fonte: Elaborado pelo autor.

(c) Boxplots com dispersão das amostras para os métodos de busca *versus* ρ.



Fonte: Elaborado pelo autor.

(d) Boxplots com dispersão das amostras para os métodos de busca *versus* p.

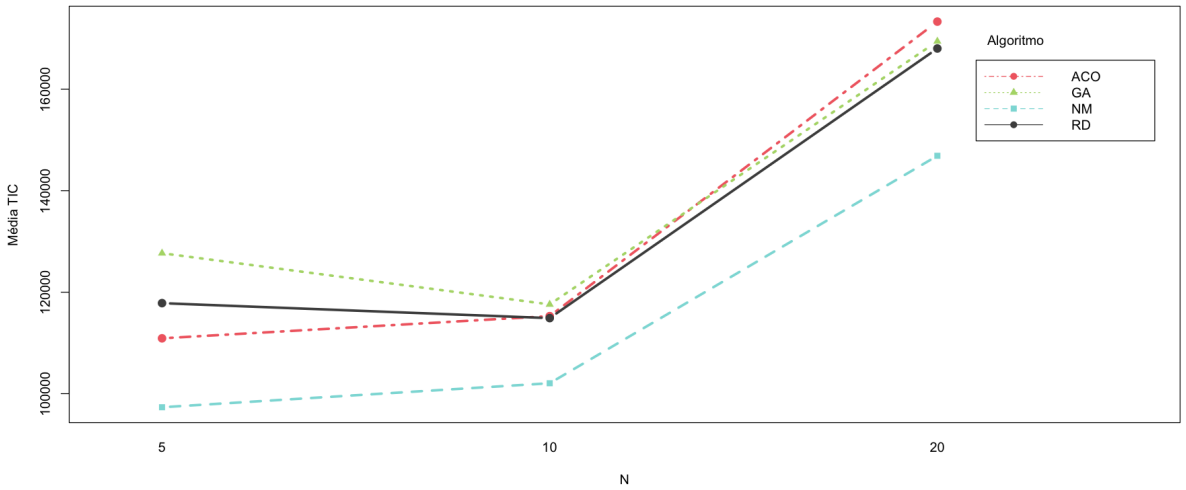


Fonte: Elaborado pelo autor.

Porém, ao analisar-se a Figura 33a, nota-se um aumento no distanciamento entre os desempenhos dos algoritmos com o aumento do número de produtos. Se, por um lado, para poucos produtos ($N = 5$) há uma maior proximidade entre os desempenhos entre os algoritmos *Nelder-Mead* e *Ant-Colony* (sendo estes os dois melhores métodos nessas condições), quando se aumenta o número de produtos ($N = 20$) o método de *Nelder-Mead* passa a ser indiscutivelmente o melhor método, enquanto o algoritmo *Ant-Colony* passa a ser o método com a pior performance, conforme ilustrado na Figura 34. Além disso, por meio da Tabela 26, nota-se que

o algoritmo *Nelder-Mead* apresenta custos de 11 a 24% inferiores do que os outros métodos.

Figura 34: Gráfico de interação entre os fatores "métodos de busca" e "número de produtos N".



Fonte: Elaborado pelo autor.

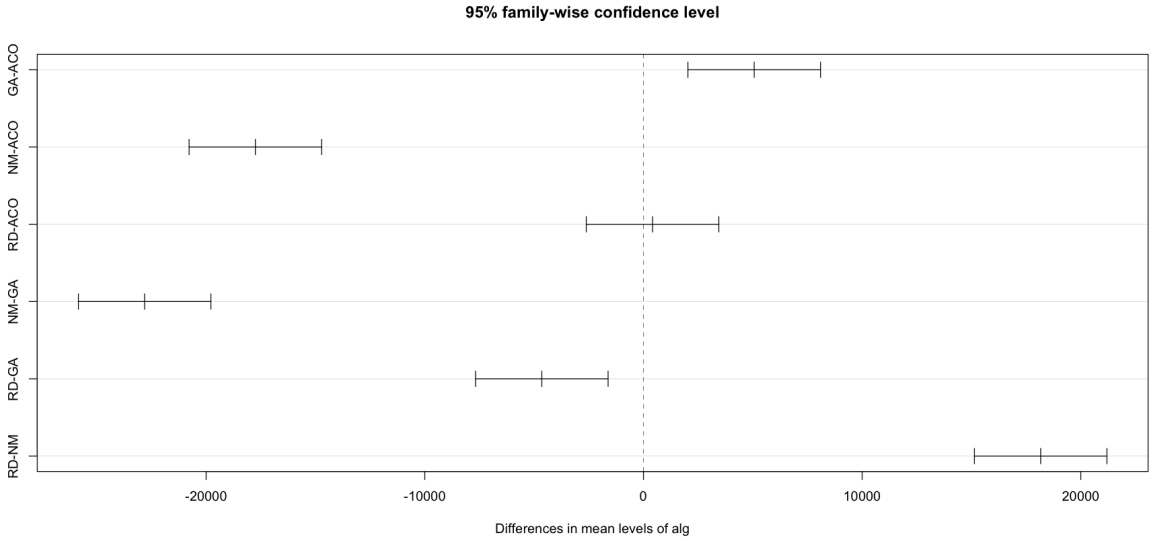
Tabela 26: Diferença percentual entre o custo médio do *Nelder-Mead versus* os demais algoritmos para diferentes níveis de N.

N	NM vs. ACO	NM vs. GA	NM vs. RD
5	-12,2%	-23,8%	-17,4%
10	-11,5%	-13,2%	-11,2%
20	-15,3%	-13,3%	-12,6%

O fato acima pode ser explicado pela diferença de requisitos exigida pelos dois métodos. Enquanto o algoritmo de *Nelder-Mead* pode ser executado em espaços contínuos de soluções, o algoritmo *Ant-Colony* exige uma discretização do espaço para que o problema possa ser representado como um grafo. Dessa forma, com o aumento do número de produtos, a dimensão do problema a ser estudado aumenta e o tamanho e densidade do grafo que representa o problema crescem de forma exponencial, o que provavelmente influencia negativamente na performance do algoritmo e dificulta sua aplicação em cenários com grande quantidade de produtos.

Para verificar a relevância estatística dessa diferença entre o desempenho médio dos algoritmos, realiza-se um teste HSD de Tukey (*Honest Significant Differences*). Esse teste é uma ferramenta estatística de comparação múltipla utilizada para identificar diferenças significativas entre as médias de vários grupos em um conjunto de dados.

Figura 35: Teste de Tukey para os métodos de busca.



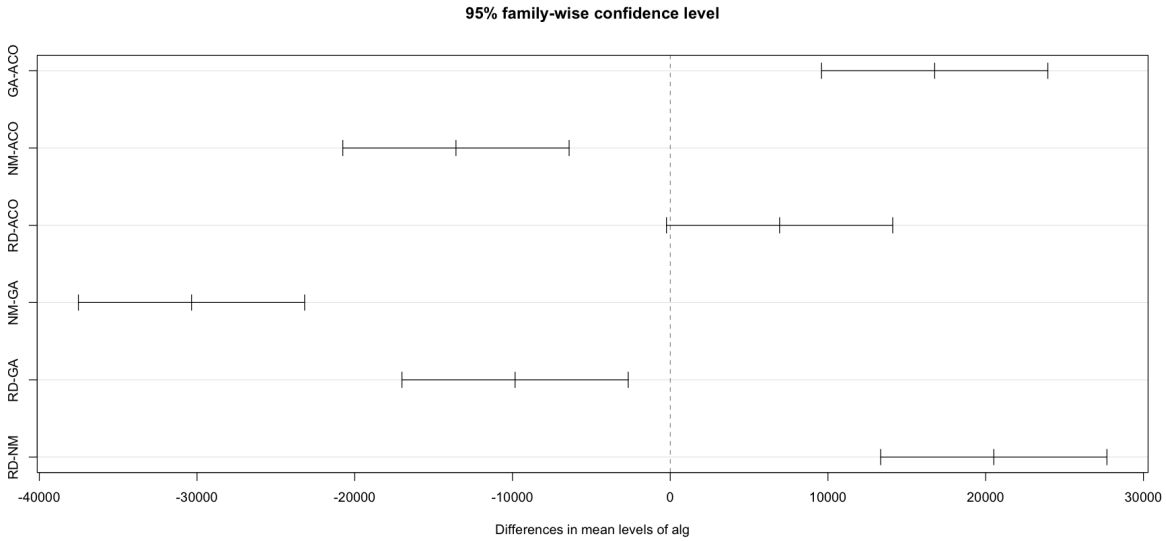
Fonte: Elaborado pelo autor.

A Figura 35 indica que, para um nível de significância de $\alpha = 5\%$, o algoritmo de *Nelder-Mead* apresenta, de fato, um desempenho superior aos demais métodos de busca, já que é possível afirmar que há uma diferença não nula entre as médias de desempenho dos algoritmos. De forma, geral, a maior diferença de desempenho ocorre quando compara-se os algoritmos *Nelder-Mead* e *Genetic Algorithm*, dado que o primeiro apresentou soluções com custos médios cerca de 20.000 unidades monetárias mais baratos, em média, do que o último.

Conforme observado anteriormente, por meio Figura da 36, confirma-se o destaque de desempenho do *Nelder-Mead* perante os outros métodos de busca para todos os níveis de N testados. Ainda, é possível confirmar que, conforme o problema torna-se mais complexo (i.e. maiores valores de N), a diferença de desempenho entre os métodos de busca aleatória, Algoritmo Genético e *Ant-Colony* torna-se quase irrelevante para um nível de significância de $\alpha = 5\%$, conforme apresentado nas Figuras 36b e 36c.

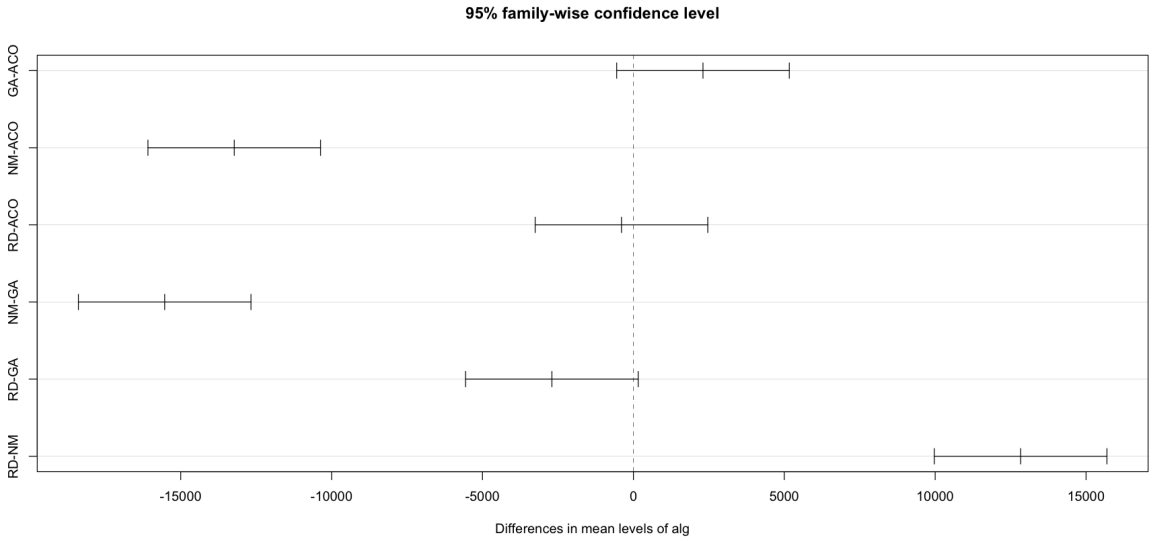
Figura 36: Testes de Tukey para os métodos de busca para (a) $N = 5$, (b) $N = 10$ e (c) $N = 20$.

(a) Teste de Tukey para os métodos de busca com $N = 5$.



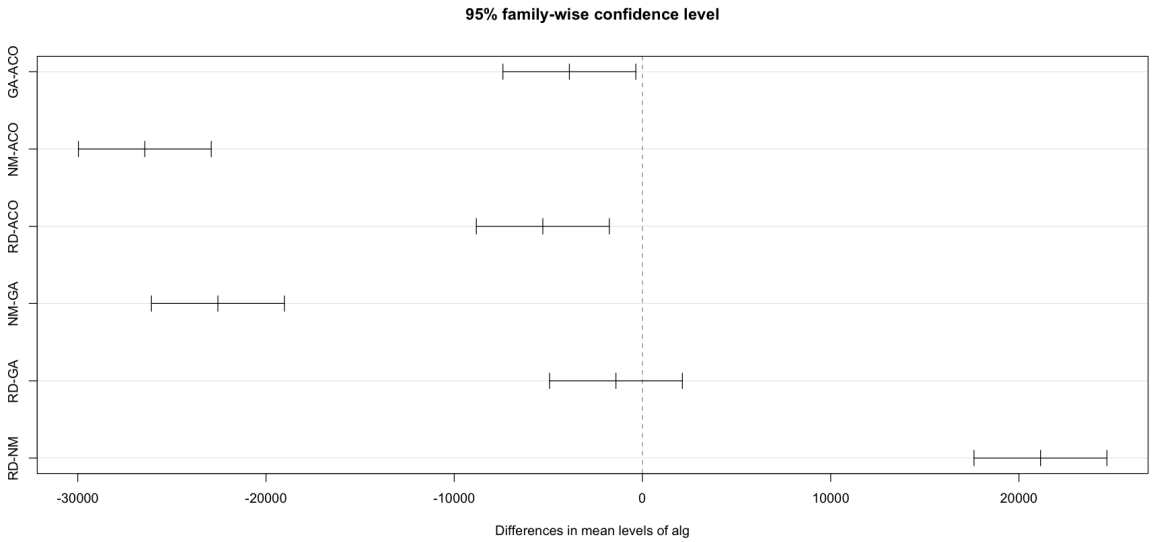
Fonte: Elaborado pelo autor.

(b) Teste de Tukey para os métodos de busca com $N = 10$.



Fonte: Elaborado pelo autor.

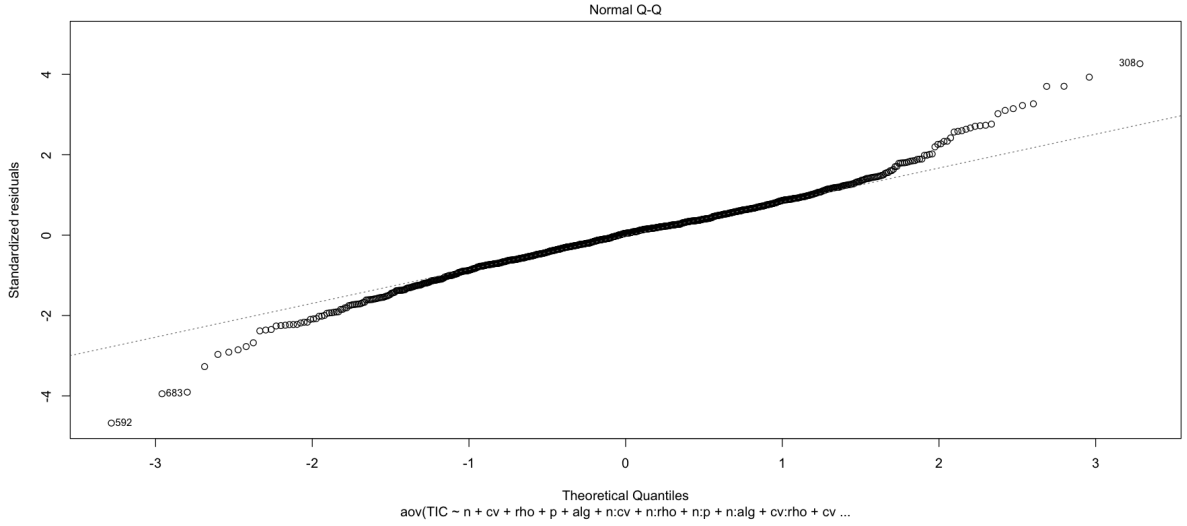
(c) Teste de Tukey para os métodos de busca com $N = 20$.



Fonte: Elaborado pelo autor.

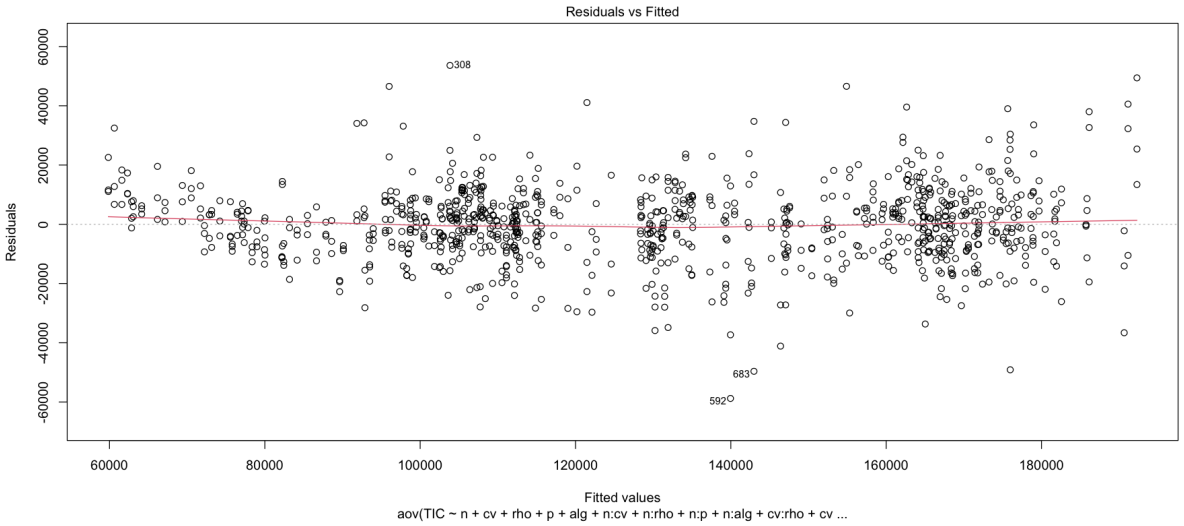
Por fim, nas Figuras 37 e 38, apresentam-se duas análise de resíduos para verificar a validade de algumas das hipóteses adotadas para a execução da Análise de Variância realizada nesta seção. Na Figura 37, nota-se que a grande maioria dos pontos acompanha a linha da reta normal, tendo maior dispersão nos valores extremos em direções opostas, o que indica a validade da hipótese de normalidade na distribuição dos resíduos. Em seguida, pela Figura 38, observa-se que a linha vermelha tende a manter-se próxima da linha tracejada, o que fornece um forte indício de aderência dos dados ao modelo linear proposto (Equação 7.1).

Figura 37: Gráfico Q-Q de resíduos padronizados.



Fonte: Elaborado pelo autor.

Figura 38: Gráfico Resíduos *vs* Fitted.



Fonte: Elaborado pelo autor.

Portanto, a análise dos resíduos permite validar o modelo proposto para a análise do desempenho dos diferentes métodos de busca implementados para a resolução do problema de otimização dos parâmetros de controle de estoque no contexto do SELSP.

Por fim, além dos resultados obtidos pelos métodos de busca, propõe-se a discussão dos efeitos da calibração nos resultados observados. Como explicado na seção 5, cada método de busca exigiu uma etapa de calibração dos seus hiperparâmetros. Porém, devido à diferença entre as quantidades de parâmetros a serem calibrados em cada método (vide Tabela 5) e o tempo necessário para a execução dos experimentos de calibração, a busca de valores ótimos para os hiperparâmetros de métodos mais complexos (ex: *Genetic Algorithm* e *Ant-Colony*) torna-se mais trabalhosa.

Diante de cenários com restrições de tempo e recursos, esse fato pode limitar a performance dos algoritmos, já que os esforços para a exploração dos experimentos de calibração podem ser reduzidos. Nesse sentido, algoritmos com menos parâmetros, tais como o *Nelder-Mead*, podem tornar-se mais atrativos, pois permitem uma melhor exploração do espaço de possíveis valores dos parâmetros, o que pode aumentar as chances de ganho de performance do método de busca.

Dessa forma, diante dos resultados e discussões apresentados anteriormente, no capítulo 8 a seguir, apresentam-se as principais conclusões obtidas por meio deste Trabalho de Formatura, assim como ressalvas e possíveis desdobramentos futuros para o Trabalho.

8 CONCLUSÕES

8.1 Síntese

O controle de estoques é foco de atenção para muitas empresas devido a sua importância financeira e operacional. Porém, ainda que largamente estudado, a calibração de parâmetros de estoque ainda levanta desafios às empresas, que acabam por recorrer a diferentes soluções para otimizar sua operação.

Nesse sentido, o presente Trabalho de Formatura se propôs a estudar o problema de calibração de parâmetros de estoque no contexto do *Stochastic Economic Scheduling Problem* (SELSP), que consiste na programação de uma única máquina capaz de produzir múltiplos produtos, mas apenas um tipo de cada vez. Embora teórico, o SELSP ilustra a realidade de muitas indústrias, principalmente do setor químico, cosmético e têxtil, evidenciando a contribuição prática do estudo para a indústria.

Para tratar do problema de calibração dos parâmetros de estoque, foi proposta uma abordagem por meio de um modelo de simulação-otimização estruturado em uma plataforma aberta, o *Python*. Essa abordagem trouxe diversas vantagens para resolução do problema.

A primeira vantagem foi a integração entre o modelo de simulação e de otimização dos parâmetros de estoque, o que permitiu que os valores sugeridos pelo modelo de otimização pudessem ser facilmente avaliados pelo modelo de simulação. Além disso, os resultados da simulação puderam ser usados para alimentar o modelo de otimização novamente, estabelecendo um ciclo de retroalimentação entre os modelos de forma sinérgica.

Ainda, a abordagem adotada permitiu ter maior transparência e controle sobre o método de otimização. Em geral, as empresas buscam *softwares* licenciados/pagos para poder otimizar seus indicadores operacionais. Porém, há pouca visibilidade sobre a metodologia de otimização adotada por essas ferramentas, fazendo com que o processo de otimização se torne uma "caixa preta". Em geral, essas soluções de mercado não adotarão o método de otimização mais adequado para o problema da empresa, já que elas são oferecidas para diferentes problemas, sendo então necessário fazer uso de métodos mais genéricos.

Todavia, a implementação de um modelo próprio em *Python* permitiu que fossem testados

diferentes métodos de busca de forma a verificar qual seria o mais adequado para o problema estudado. Além disso, essa abordagem permitiu o ajuste fino dos métodos por meio da calibração de seus hiperparâmetros para melhorar ainda mais o desempenho do método de otimização para a calibração dos parâmetros de estoque.

Ademais, como mencionado acima, a implementação do modelo de simulação-otimização em uma plataforma *open source* permite oferecer uma ferramenta para a resolução do problema estudado de forma gratuita, sem a necessidade de adquirir licenças para *softwares*, além de poder receber contribuição de outros pesquisadores que tenham interesse em dar continuidade ao estudo do problema.

Uma das realizações fundamentais deste trabalho foi a avaliação de quatro métodos de busca diferentes, a saber: Busca Aleatória, *Nelder-Mead*, *Genetic Algorithm* e *Ant-Colony Optimization*. Notavelmente, o *Nelder-Mead* destacou-se tanto em desempenho (soluções até 24% melhores) como em praticidade, demonstrando maior eficácia na busca por soluções ótimas e maior facilidade na calibração de hiperparâmetros em relação aos outros métodos testados.

O algoritmo se mostrou mais eficiente em diferentes condições, como ambientes de maior incerteza/variabilidade, mas também em cenários de maior complexidade. O ganho de desempenho se tornou mais notável em problemas de maior dimensão, muito por causa da flexibilidade do método em buscar soluções em espaços contínuos. Esse destaque reforça a importância crítica da escolha desse método de busca na otimização do problema SELSP, devido à maior eficácia em cenários diversos.

Em resumo, a implementação bem-sucedida de um modelo de simulação-otimização em *Python*, com o método de busca *Nelder-Mead*, apresentou uma contribuição significativa para o campo da otimização de parâmetros de estoque em ambientes de produção complexos, como o SELSP. Essa abordagem flexível e acessível não apenas trouxe uma ferramenta de solução para o problema, mas também refletiu um compromisso com a acessibilidade e a redução da dependência de soluções comerciais. Dessa forma, o Trabalho proporcionou uma valiosa contribuição para a comunidade científica e industrial, oferecendo uma alternativa de código aberto para abordar desafios de otimização de estoque.

8.2 Limitações e desdobramentos futuros

Reconhece-se que o presente Trabalho apresenta certas limitações, que podem ser usadas para orientar futuros trabalhos sobre o tema. A calibração dos métodos de busca, embora essencial, permanece um desafio. Neste trabalho, optou-se pelo uso de um experimento fatorial

completo para a realização dos experimentos de calibração dos métodos de busca, sendo testadas aproximadamente 256 combinações de hiperparâmetros para cada método. A abordagem adotada pode ser mais vulnerável a níveis de calibração subótimos dos métodos de busca, o que pode afetar diretamente o desempenho dos métodos testados.

Além disso, neste Trabalho foram avaliados 4 métodos de busca, embora na literatura existam diversos outros métodos que poderiam ser adaptados ao problema estudado. Sendo assim, a avaliação de métodos não foi exaustiva e abre oportunidades para que sejam explorados outros algoritmos de otimização.

Nesse sentido, melhorias na calibração de hiperparâmetros e a exploração de outros métodos de otimização devem ser áreas de foco contínuo para futuros desdobramentos deste trabalho. Além disso, a incorporação de métodos de busca local pode permitir uma exploração mais profunda do espaço de soluções, aprimorando ainda mais os resultados obtidos.

Outra direção futura sugerida é a implementação da calibração *online* de hiperparâmetros. Essa abordagem pode tornar o modelo mais adaptável às estratégias de diversificação e intensificação na busca de soluções, permitindo uma otimização contínua e flexível para o problema estudado. Além disso, a busca por métodos de calibração mais avançados para os hiperparâmetros dos métodos de busca pode aumentar a robustez do processo de otimização.

8.3 Disponibilidade de dados

Os dados e modelos que embasam os resultados obtidos neste Trabalho podem ser encontrados no repositório online a seguir: Repositório do GitHub.

REFERÊNCIAS

ALTIOK, T.; MELAMED, B. *Simulation Modeling and Analysis with ARENA*. Elsevier Science, 2010. ISBN 9780080548951. Disponível em: ⟨https://books.google.com.br/books?id=5SezxR5q4mYC⟩.

ANGELOVA, M.; PENCHEVA, T. Tuning genetic algorithm parameters to improve convergence time. *International Journal of Chemical Engineering*, v. 2011, 01 2011.

AXSÄTER, S. *Inventory Control*. [S.l.]: Springer International Publishing, 2015. (International Series in Operations Research & Management Science). ISBN 978-3-319-15729-0.

BARRY, R.; JAY, H.; CHUCK, M. *Operation Management: Sustainability and Supply Chain Management 12th. Ed.* [S.l.]: Pearson Education Limited, 2017. ISBN 978-0-13-413042-2.

CHENG, J. Y.; MAILUND, T. Ancestral population genomics using coalescence hidden markov models and heuristic optimisation algorithms. *Computational Biology and Chemistry*, v. 57, p. 80–92, 2015. ISSN 1476-9271. 13th Asia Pacifi c Bioinformatics Conference, HsinChu, Taiwan, 21-23 January 2015. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S1476927115000183⟩.

CHUNG, S. H.; CHAN, H. K. A two-level genetic algorithm to determine production frequencies for economic lot scheduling problem. *IEEE Transactions on Industrial Electronics*, v. 59, n. 1, p. 611–619, 2012.

HAMADI, Y.; MONFROY, E.; SAUBION, F. *Autonomous Search*. Springer Berlin Heidelberg, 2012. (SpringerLink : Bücher). ISBN 9783642214349. Disponível em: ⟨https://books.google.com.br/books?id=_s9L8YBgVmEC⟩.

HAUPT, R.; HAUPT, S. *Practical Genetic Algorithms*. Wiley, 2004. (Wiley InterScience electronic collection). ISBN 9780471671756. Disponível em: ⟨https://books.google.com.br/books?id=k0jFfsmbtZIC⟩.

HOPP, W.; SPEARMAN, M. *Factory Physics: Third Edition*. [S.l.]: Waveland Press, 2011. ISBN 9781478609049.

JIN, Y. et al. *Data-Driven Evolutionary Optimization: An Overview and Case Studies*. [S.l.: s.n.], 2019. v. 23. 442-458 p.

KÄMPF, M.; KÖCHEL, P. Simulation-based sequencing and lot size optimisation for a production-and-inventory system with multiple items. *International Journal of Production Economics*, v. 104, n. 1, p. 191–200, 2006. ISSN 0925-5273. Strategic Issues and Innovation in Production Economics. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0925527306000661⟩.

LARRAÑETA, J.; ONIEVA, L. The economic lot-scheduling problem: A simple approach. *The Journal of the Operational Research Society*, Palgrave Macmillan Journals, v. 39, n. 4, p. 373–379, 1988. ISSN 01605682, 14769360. Disponível em: ⟨http://www.jstor.org/stable/2582117⟩.

LÖHNDORF, N.; RIEL, M.; MINNER, S. Simulation optimization for the stochastic economic lot scheduling problem with sequence-dependent setup times. *International Journal of Production Economics*, v. 157, p. 170–176, 2014. ISSN 0925-5273. The International Society for Inventory Research, 2012. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0925527314001625⟩.

MESQUITA, M. A.; TOMOTANI, J. V. Simulation-optimization of inventory control of multiple products on a single machine with sequence-dependent setup times. *Computers Industrial Engineering*, v. 174, p. 108793, 2022. ISSN 0360-8352. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0360835222007811⟩.

NAHMIAS, S.; OLSEN, T. *Production and Operations Analysis: Seventh Edition*. Waveland Press, 2015. ISBN 9781478628248. Disponível em: ⟨https://books.google.com.br/books?id=SIsoBgAAQBAJ⟩.

PATERNINA-ARBOLEDA, C. D.; DAS, T. K. A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problem. *Simulation Modelling Practice and Theory*, v. 13, n. 5, p. 389–406, 2005. ISSN 1569-190X. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S1569190X04001406⟩.

SOX, C. R. et al. A review of the stochastic lot scheduling problem1this paper is based upon work supported in part by the national science foundation under grant number dmi-9409344.1. *International Journal of Production Economics*, v. 62, n. 3, p. 181–200, 1999. ISSN 0925-5273. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0925527398002473⟩.

TOMOTANI, J.; MESQUITA, M. Lot sizing and scheduling: a survey of practices in brazilian companies. *Production Planning Control*, v. 29, p. 1–11, 12 2017.

WAGNER, M.; SMITS, S. R. A local search algorithm for the optimization of the stochastic economic lot scheduling problem. *International Journal of Production Economics*, v. 90, n. 3, p. 391–402, 2004. ISSN 0925-5273. Production Control and Scheduling. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0925527303002019⟩.

WINANDS, E.; ADAN, I.; van Houtum, G. The stochastic economic lot scheduling problem: A survey. *European Journal of Operational Research*, v. 210, n. 1, p. 1–9, 2011. ISSN 0377-2217. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0377221710004170⟩.

WONG, K. Y.; KOMARUDIN. Parameter tuning for ant colony optimization: A review. p. 542–545, 2008.

APÊNDICE A

A.1 Exemplo de funcionamento do Modelo SimOpt construído

Figura 39: Modelo SimOpt - Exemplo de configuração de uma instância de problema no modelo com $N = 10$, $cv = 0,1$, $\rho = 0,9$, $p = 0,05$ e tempo máximo de execução de 3 min.

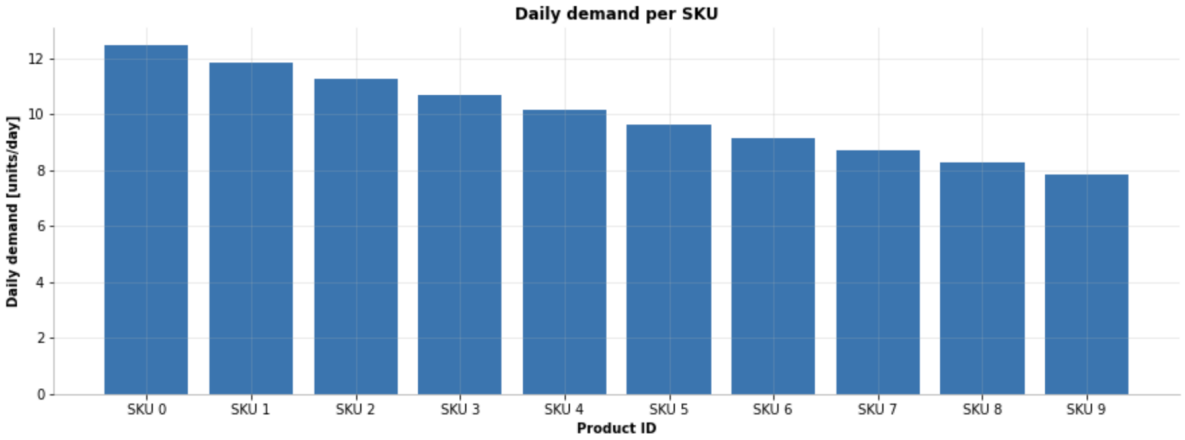
4) Setup flags

In [ ]:

```
## Defining parameters of the problem

flags_ = {

        ## 1) Problem parameters

        'N': 10,                          ## Number of items
        'cv': 0.1,                       ## Coefficient of variation
        'rho': 0.9,                      ## Availability rate of the machine
        'p_geo': 0.05,                    ## Demand distribution coefficient: {0 = Uniform; 0,05 = Slightly concentrated ; 0,15 = Highly concent

        'mod': 'LDS',                    ## Production prioritization method: {FIS, LDS}
        'days_year': 250,                ## Working days per year
        'setup0': 1,                     ## Base setup time (hours)

        ## 2) Cost parameters

        'gross_margin': 1000000,         ## Gross margin
        'cmu': 40,                       ## Contribution margin per unit
        'sc': 250,                       ## Setup cost
        'lc': 40,                        ## Setup cost
        'hc': 0.1,                       ## Daily holding cost


        ## 3) Simulation-Optimization parameters
        'max_time': 3*60 ,               ## Max time for the optimization model (seconds)
        'simulation_time': 24*100,       ## in hours
        'random_state': 1,               ## Randomness: {0, 1}
        'reps': 3,                       ## Repetitions for each solution test
        'tqdm': True,                    ## Progress bar: {True = ON; False = OFF}

        ## 4) Nelder-Mead hyperparameters
        'Reflection_rate': 1,
        'Expansion_rate': 0.5,
        'Contraction_rate': 0.5,
        'Shrinkage_rate': 1.5,
        'max_iter_NM': 1000,
        'max_same_best_NM': 500,
        'tol_NM': 1

        }


config_ = define_config(flags_)
```

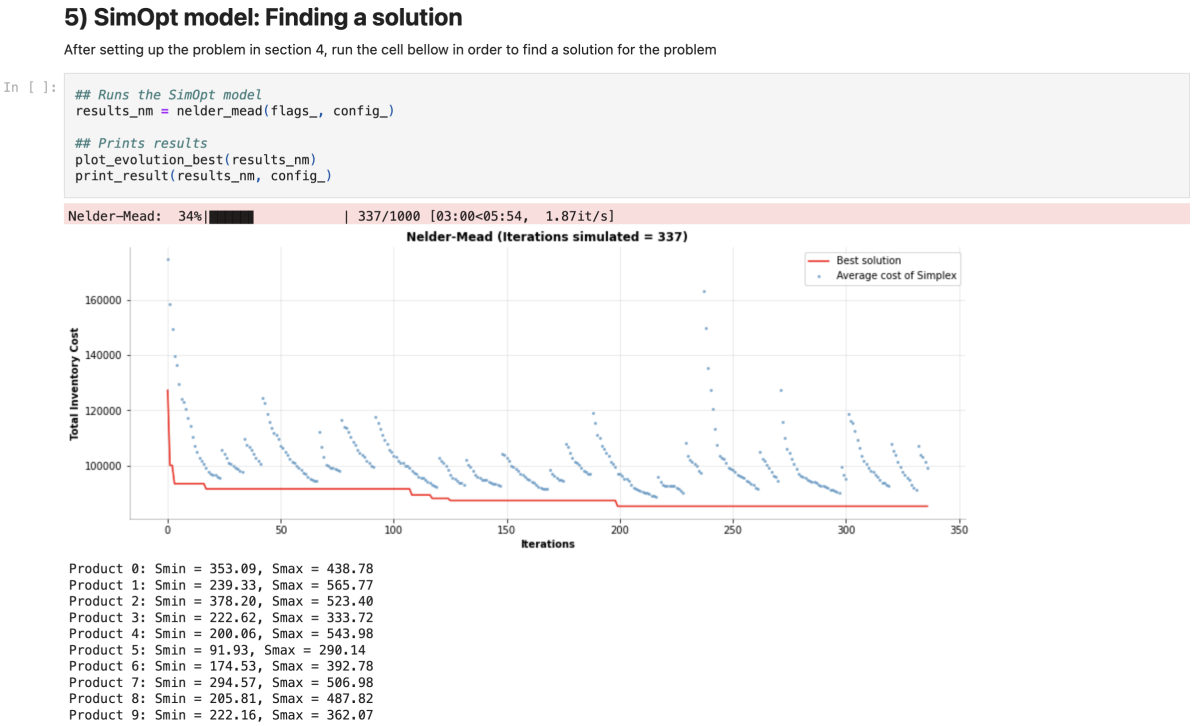
Fonte: Elaborado pelo autor.

Figura 40: Modelo SimOpt - Exemplo de simulação da distribuição da demanda entre os produtos a partir dos parâmetros indicados na configuração do problema.



Fonte: Elaborado pelo autor.

Figura 41: Modelo SimOpt - Exemplo de *output* do método de busca, com indicação dos mellhores pares (s_i, S_i) encontrados para cada produto.



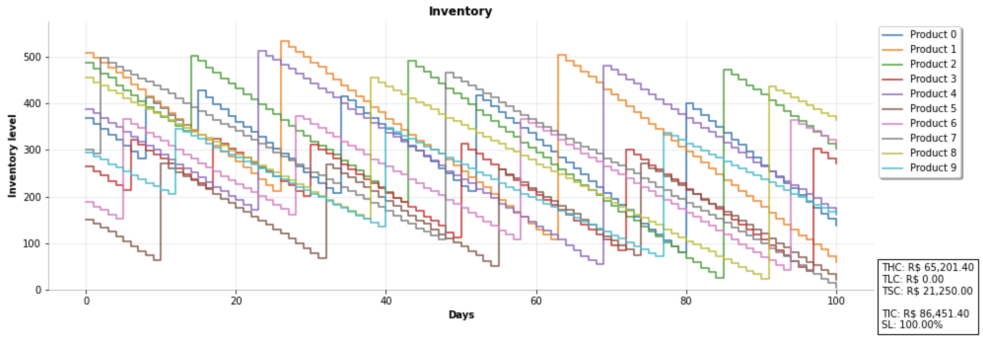
Fonte: Elaborado pelo autor.

Figura 42: Modelo SimOpt - Exemplo de simulação da evolução do estoque e dos indicadores de performance da fábrica utilizando os parâmetros de estoque sugeridos pelo método de busca.

6) Simulating best solution

In [ ]:
```
simulate_best(results_nm, config_)
```

```
==============
Simulation Started
Days simulated: 100
Simulation Ended
==============
Performance indicators:
TIC: 86,451.40
THC: 65,201.40  | TLC: 0.00  | TSC: 21,250.00
Total Contribution Margin: 998,900.00
SL: 100.00%
```



Fonte: Elaborado pelo autor.