INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA E GESTÃO DA CADEIA DE ABASTECIMENTO




# Gestão de inventários no retalho

**BRUNA RODRIGUES CRUZ**
outubro de 2024

Instituto Superior de Engenharia do Porto

# GESTÃO DE INVENTÁRIOS NO RETALHO

Bruna Rodrigues Cruz

**2024**

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

P.PORTO


P.PORTO

Instituto Superior de Engenharia do Porto

# GESTÃO DE INVENTÁRIOS NO RETALHO

Bruna Rodrigues Cruz

Estudante n.º 1210734

Dissertação apresentada ao Instituto Superior de Engenharia do Porto para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia e Gestão da Cadeia de Abastecimento, realizada sob a orientação do Doutor Manuel Pereira Lopes.

**2024**

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

P.PORTO


P.PORTO

# AGRADECIMENTOS

A fase final deste percurso acaba por ser um momento de introspeção e reflexão, sobretudo de quem esteve ao meu lado neste longo caminho e que de alguma forma contribuiu para o meu sucesso. Assim, não posso deixar de agradecer às entidades e pessoas por tudo o que fizeram por mim.

Ao Instituto Superior de Engenharia do Porto, reconheço o seu papel na minha formação. Agradeço aos professores e em particular ao meu orientador, Professor Manuel Pereira Lopes pelo seu apoio, disponibilidade, sugestões e todos os conhecimentos transmitidos que foram determinantes para a concretização desta dissertação.

Um obrigado à empresa pela disponibilização dos dados, e sobretudo ao Jorge e ao Bruno, pelo apoio que me transmitiram, pela preocupação, acompanhamento e todas as sugestões que foram imprescindíveis para a conclusão deste projeto.

Um agradecimento à minha família, em especial aos meus pais pelo amor e apoio incondicional, por todo o esforço e trabalho para me proporcionarem sempre o melhor, por toda a compreensão e paciência nos dias mais difíceis ao longo do mestrado. Obrigada por acreditarem em mim e no percurso académico que segui, mesmo quando eu duvidei.

À minha irmã, por estar sempre presente mesmo estando na ponta oposta do país, é um exemplo para mim e a pessoa que mais admiro. Agradeço por todas as conversas e conselhos, por todos os momentos em que me senti compreendida e por toda a preocupação.

Por último, agradeço aos amigos que Viana me deu, que tornaram todo o meu percurso académico muito mais leve. Obrigada por todos os momentos de pausas e distração, pelo incentivo, compreensão, suporte e amizade que fui sentindo ao longo deste tempo. Proporcionaram-me equilíbrio e motivação nos momentos mais desafiantes.

página propositadamente em branco

# RESUMO

Equilibrar a existência de *stock* em armazém com as necessidades das lojas é um desafio na gestão de inventários, devido ao elevado número de *stock keeping units* (SKU). O ideal é manter um nível de *stock* adequado e suficiente para corresponder à procura, pois ter *stock* em excesso implica custos adicionais, mas ter rutura de *stock* resulta na perda de vendas. Desta forma, além da gestão de inventários no retalho ser um requisito, é a temática sobre o qual este projeto se debruça, nomeadamente, no retalho de produtos cosméticos.

Dada a natureza real e complexa do problema, foi fundamental observar e refletir acerca do modelo de gestão de inventários utilizado pela empresa. Durante esta fase, observou-se que o processo de abastecimento das posições de *picking* oferecia grandes oportunidades de melhoria, quer em termos de localização dos artigos, quantidades a repor e frequência das reposições. Nesse sentido, a melhoria deste processo surgiu como uma oportunidade para estudar e desenvolver uma solução eficaz, que obtivesse um desempenho superior ao método atualmente utilizado pela empresa.

Para dar resposta ao problema, foi desenvolvido um sistema de apoio à decisão, que após recolher os dados, fez a preparação dos mesmos, de forma a aplicar o algoritmo desenvolvido. Assim, partindo das necessidades das lojas e do *stock* atual, o sistema de apoio criado determinou uma lista de necessidades de reposição, para posteriormente proceder à sua alocação nas posições de *picking*. É de salientar que a lista de necessidades de reposição elaborada segue uma ordenação pela prioridade de cada artigo, estimada pela sua relevância nas vendas e cuja presença em loja fosse indispensável.

O modelo foi testado ao longo de uma semana, apresentando resultados favoráveis quando comparado com o estado atual. A comparação entre a situação real apresentada pela empresa e o algoritmo proposto, revela que o novo modelo proporciona uma redução média de 74,7% na área ocupada por novas posições, evita em média a reposição de 78,7% em posições livres e reduz em cerca de 36,7% o número de reposições efetuadas. Além disso, o tempo computacional referente ao processamento do sistema de apoio à decisão foi sempre inferior a 1 minuto. Estes resultados demonstram o potencial da solução proposta, evidenciando um desempenho superior ao método atual para o período analisado.

PALAVRAS-CHAVE

Gestão de inventários, reposição das posições de *picking*, heurísticas

página propositadamente em branco

# ABSTRACT

Balancing the existence of stock in the warehouse with the needs of stores is a challenge in inventory management, due to the high number of stock keeping units (SKU). The ideal is to maintain adequate and sufficient stock levels to meet demand, as having excess stock implies additional costs, but stock shortages result in lost sales. Thus, in addition to inventory management in retail being a requirement, it is the topic that this project focuses on, particularly in cosmetics retail.

Given the real nature of the problem, it was essential to observe and reflect on the inventory management model used by the company. During this phase, it was observed that the replenishment process offered significant opportunities for improvement, either in terms of the location of the items, quantities to be replenished and the frequency of replenishments. In this sense, improving this process emerged as an opportunity to study and develop an effective solution that would achieve a better performance than the company's current method.

To address the problem, a decision support system was developed, which, after collecting the data, prepares it, to apply the developed algorithm. Thus, based on the stores' needs and the current stock, the created support system determines a list of replenishment needs to subsequently allocate them to the picking positions. It is worth noting that the list of replenishment needs follows a prioritization based on each item's relevance to sales and its indispensable presence in the store.

The model was tested over a week, presenting favorable results when compared to the current state of the company. The comparison between the real situation presented by the company and the proposed algorithm reveals that the new model provides an average reduction of 74,7% in the area occupied by new positions, avoids the average occupation of 78,7% of replacements in free positions and reduces by an average of 36,7% the number of replenishments carried out. Furthermore, the computational time for processing the decision support system was always less than 1 minute. These results demonstrate the potential of the proposed solution, showing a greater performance to the current method for the period analyzed.

KEYWORDS

Inventory management, replenishment of picking positions, heuristics

página propositadamente em branco

# ÍNDICE

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABELAS

página propositadamente em branco

# LISTAS DE SIGLAS

## Lista de Siglas

| | |
|---|---|
| 2AC | *Advantage Actor-Critic* |
| DL | *Deep Learning* |
| DRL | *Deep Reinforcement Learning* |
| DB | *Direct Backpropagation* |
| ISEP | Instituto Superior de Engenharia do Porto |
| IA | Inteligência Artificial |
| JM | Jogos de Markov |
| ML | *Machine Learning* |
| PHN | *Planning Horizon Normalized* |
| P. Porto | Politécnico do Porto |
| PDM | Processo de Decisão de Markov |
| PDI | Processo de Decisão Interativo |
| PPO | *Proximal Policy Optimization* |
| RNA | Redes Neuronais Artificiais |
| RL | *Reinforcement Learning* |
| SARSA | *State-Action-Reward-State-Action* |
| SKU | *Stock Keeping Unit* |

página propositadamente em branco

# 1. INTRODUÇÃO

Neste capítulo introdutório é abordado o contexto deste projeto, apoiado pela descrição do problema da gestão de inventários e pela sua relevância. De seguida, identifica-se a questão e os objetivos de investigação propostos, nomeadamente o objetivo geral e os objetivos específicos. No final do capítulo encontra-se a opção metodológica selecionada para alcançar os objetivos do projeto, assim como a apresentação da empresa que serviu de caso de estudo e a estrutura da dissertação.

## 1.1. Problema de investigação, enquadramento e pertinência

Equilibrar a oferta de *stock* com a procura é um desafio na gestão de inventários. A dificuldade em gerir inventários é descrita por Agù & Nnate (2016) como o processo de escolher a quantidade de produto que uma empresa fornece e quando esta deve solicitar tais produtos. Idealmente, uma empresa de retalho deve ter inventário suficiente para satisfazer a procura, sem perder vendas pela falta de produto. Por outro lado, ter *stock* em excesso implica custos associados ao mesmo. Singh & Kumar (2011) apontam como principais custos, o custo administrativo de realizar um pedido de encomenda, o custo de manutenção de *stock* e o custo de perda de lucro quando o produto se encontra em rutura. Considerando que estes tipos de custos podem absorver entre 25% a 40 % dos custos totais de uma empresa (Ballou & Srivastava, 2007) é crucial adotar técnicas de gestão de inventários eficazes para alcançar o sucesso no mercado competitivo atual.

A adoção de ferramentas de controlo de inventários é um desafio para muitas empresas de retalho (Lwiki et al., 2013), sendo o seu propósito reduzir custos de modo a melhorar a eficiência organizacional (Eveline et al., 2019). Um estudo realizado por Lieberman & Demeester (1999) estabeleceu uma relação positiva entre a redução de *stocks* e o aumento da produtividade. Na vertente financeira, o estudo de Claycomb et al. (1999) demonstrou que a diminuição do inventário proporciona uma melhoria de desempenho no retorno sobre o investimento, no retorno sobre as vendas e no lucro. Posteriormente, Chebet & Kitheka (2019) observaram a existência de uma relação positiva entre a viabilidade operacional, a gestão de inventários e o aumento do retorno do investimento da empresa.

Assim, considerando que a gestão de inventários deve ser vista como um requisito e não como uma tendência (Aro-Gordon & Gupte, 2016), este projeto irá debruçar-se sobre esta temática no contexto de uma empresa de retalho de produtos cosméticos, através da implementação de um algoritmo de gestão de reposição de inventários, pretendendo-se de seguida comparar o desempenho deste com o modelo em uso pela empresa, de forma a fortalecer a sua cadeia de abastecimento.

## 1.2. Questão e objetivos de investigação

Com base no problema abordado no ponto anterior, este projeto pretende responder à questão: é possível desenvolver e implementar um novo algoritmo de gestão de reposição de inventários com melhor desempenho que o atual método da empresa e que melhore a atuação da sua cadeia de abastecimento?

Desta forma, o objetivo principal passa por criar e implementar um novo algoritmo para a gestão de inventários de uma empresa de retalho de produtos cosméticos, sendo os objetivos específicos a cumprir os seguintes:

- Identificar as limitações do algoritmo atual;
- Conceber o algoritmo de gestão de inventários;
- Implementar o algoritmo criado;
- Avaliar os resultados obtidos e comparar com o desempenho do modelo atual.

## 1.3. Opções metodológicas

Considerando os objetivos previamente enunciados é fundamental selecionar a opção metodológica a seguir.

Na literatura, dependendo do problema em estudo, Creswell (2012) divide a pesquisa metodológica nas perspetivas qualitativa e quantitativa. O autor refere que na pesquisa qualitativa são desenvolvidos problemas com variáveis desconhecidas, sendo o objetivo principal explorá-las, enquanto na pesquisa quantitativa, o investigador identifica um problema de pesquisa, considerando factos observáveis com base na necessidade de explicar a razão de um dado acontecimento. Assim, sendo o foco deste trabalho a comparação de resultados do algoritmo desenvolvido com o algoritmo em uso pela empresa e posteriormente mensuração dos ganhos, a pesquisa enquadra-se na perspetiva quantitativa.

Os métodos de investigação apresentam designações diferentes entre autores, sendo que Pereira Coutinho (2011) propõe os métodos experimental puro, quase experimental, *survey*, etnográfico, investigação-ação, correlacional, entre outros. Neste trabalho será seguido o método de investigação-ação dada a natureza prática e real do problema que se pretende resolver, tendo em vista a melhoria do mesmo através de uma estratégia reflexiva onde "o investigador reflete sobre a ação antes e depois, numa visão integrada da teoria e prática" (Cardoso, 2014). A investigação-ação baseia-se em validar ou refutar a teoria com base nas ações implementadas para solucionar o problema. Trata-se de um processo cíclico e dinâmico, englobando as fases de planificação, ação, observação e reflexão (Creswell, 2012).

## 1.4. Apresentação da empresa

A empresa em estudo é uma empresa portuguesa de comércio de retalho de produtos cosméticos. A sua sede situa-se na Maia e tem mais de 100 lojas espalhadas por Portugal continental, Madeira e Espanha. O seu volume de negócios no ano transato foi de 70 milhões €, apresentando *stock* em loja de 7.5 milhões €.

As vendas são segmentadas em dois tipos, os clientes finais e os clientes profissionais, que têm de provar serem profissionais da área da cosmética para acederem a uma tarifa especial. Neste sentido, as vendas para profissionais situam-se nos 52,4%, sendo que os restantes 47,6% correspondem à percentagem das vendas realizadas pelos clientes finais. Ambos os clientes têm acesso a campanhas constantes, sendo que devido aos descontos, a quantidade de produtos comprados por profissionais é bastante superior.

## 1.5. Estrutura do trabalho

Este trabalho é composto por 5 capítulos. Neste primeiro capítulo é apresentada a contextualização do projeto, apoiada pela descrição do problema e sua importância, mas também pela questão de pesquisa, objetivos e metodologia que orientam o desenvolvimento desta dissertação. Além disso, é apresentada uma breve descrição da empresa em estudo.

O segundo capítulo aborda o estado da arte da gestão de inventários no retalho. Esta revisão bibliográfica envolve não só considerações da gestão de inventários, mas também acerca de *Machine Learning* e dos seus principais modelos. Inclui ainda estratégias de ML na gestão de inventários.

No capítulo 3, realiza-se a descrição do modelo de gestão de inventário seguido pela empresa nas fases de reabastecimento das lojas, *picking* e reposição das posições de *picking*, assim como as principais limitações apontadas. Além disso, inclui também a seleção dos dados, a sua análise e preparação e posteriormente, a nova proposta de solução para melhorar o processo de reposição das posições de *picking.*

O quarto capítulo é dedicado à apresentação e discussão dos resultados obtidos, comparando-os com os resultados reais do modelo atual da empresa.

O último capítulo apresenta as conclusões finais obtidas com o trabalho desenvolvido, através de uma retrospetiva dos capítulos anteriores. Neste, também são expostas as limitações e propostas de investigações futuras.

Após este capítulo, seguem-se as referências bibliográficas e os apêndices que incluem os códigos utilizados para a preparação dos dados e para a solução proposta.

página propositadamente em branco

# 2. REVISÃO BIBLIOGRÁFICA

O presente capítulo contextualiza a temática da gestão de inventários, seguindo-se uma revisão acerca de *Machine Learning* e dos seus principais modelos. No âmbito das estratégias de *Machine Learning*, são abordados algoritmos dos modelos de *Reinforcement Learning* e *Deep Learning*, para posteriormente comparar os dois métodos.

## 2.1. Gestão de inventários

A disponibilidade de inventário é um fator de sucesso no setor do retalho (Kumari et al., 2021). O inventário é percecionado de diferentes formas pelos autores, sendo que Albayrak Ünal et al. (2023) definem-no como os materiais e produtos finais ao longo da cadeia de abastecimento, para uso na produção ou distribuição ao cliente final. Por outro lado, Vrat (2014) refere que o inventário de uma empresa inclui matérias-primas, produtos em via de fabrico e produtos acabados, armazenados num local específico da cadeia de abastecimento. Apesar destas diferenças, a nível financeiro, o inventário é considerado para muitas empresas como o ativo mais importante a manter, representando entre 25% a 50% do total dos ativos, sendo que para os retalhistas, este valor aumenta para 75% a 80%, o que implica uma maior necessidade de controlo do inventário (James et al., 2018).

Neste sentido, Bose (2006) afirma que uma boa gestão de inventários é também uma boa gestão financeira, sendo o objetivo base manter o *stock* a um nível minimamente aceitável em relação aos seus custos (G. Michalski, 2013). No que diz respeito aos custos inerentes à gestão de inventários, Pandya & Thakkar (2016) referem os custos de manutenção relativos ao armazenamento e à perda de oportunidade de investir valor em *stock* em prol de outro investimento; custos de processamento de encomendas; e os custos de escassez que podem ser custos administrativos extras, custos de comunicação, entre outros. Além destes, Michalski (2013) indica também os custos relativos à obsolescência, desperdício e custos de deterioração do *stock*.

A fim de minimizar os custos enunciados, é necessário promover o equilíbrio entre a oferta e a procura, que só se torna possível com a gestão adequada dos inventários, caracterizada por D. Singh & Verma (2018) como um processo contínuo de planeamento, organização e controlo de fluxo. A implementação de estratégias de gestão de inventários é influenciada por fatores organizacionais e humanos, restrições financeiras e pela crescente adoção de tecnologias (Ahmad & Zabri, 2016). A componente financeira é considerada das mais importantes dado que pode afetar a liquidez de uma empresa, mas também por incluir diversas tomadas de decisão, nomeadamente quanto inventário físico se deve manter num armazém para fazer face a um pico inesperado da procura (Priniotakis & Argyropoulos, 2018). A vertente tecnológica apresenta uma relação ambígua na gestão de inventários. No estudo de Rushton et al. (2022) é possível comparar diferentes aplicações tecnológicas com diferentes envolvimentos humanos, tendo-se verificado um impacto positivo da tecnologia, na medida que a taxa de ocorrência de erros diminuiu com a menor movimentação do ser humano. No entanto, ainda há resistência por parte de algumas organizações no uso da tecnologia por razões financeiras, pois embora este tipo de métodos otimizem a cadeia de abastecimento, apresentam um custo elevado associado (Munyaka & Yadavalli, 2022).

A gestão de inventários é uma operação crítica nos processos de fabricação e da cadeia de abastecimento (Munyaka & Yadavalli, 2022). James et al. (2018) afirmam que para gerir inventários de forma eficaz, "os retalhistas devem compreender as necessidades dos clientes, as parcerias com os fornecedores, a tecnologia, a integridade dos dados e as medições de desempenho". Os autores referem também que os retalhistas devem ter especial atenção à alocação dos *stocks* em cada ponto de venda, pois pode evitar eventuais ruturas do mesmo, que consequentemente afetam o nível de satisfação do cliente. Uma das formas que os retalhistas usam para prevenir as ruturas de *stock* passa pela criação de um *stock* de segurança (Osman & Demirli, 2012). Além deste problema, Esther (2012) e Othman (2014) referem também a superprodução, que pelo contrário, se caracteriza pelo excesso de inventário e tem consequências a nível dos custos associados.

O equilíbrio desejado pelos retalhistas de simultaneamente maximizar o nível de serviço ao cliente, minimizar o *stock* disponível e minimizar os custos operacionais é afetado por 3 questões essenciais (Zietsman & van Vuuren, 2022):

- Que SKU's (*Stock Keeping Unit*) devem ser reabastecidos?
- Quando se deve fazer o pedido de reabastecimento?
- Quais os volumes apropriados de SKU's a serem solicitados no pedido?

Desta forma, é fundamental implementar métodos de gestão de inventário adequados, o que se torna um desafio devido à presença de diversas variáveis complexas nas cadeias de retalho, nomeadamente as flutuações na procura, as incertezas nos prazos de entrega e nos preços, a sazonalidade, as campanhas promocionais, entre outros. Assim, aliado ao facto da procura ser uma variável crítica e que a sua previsão deve ser precisa para auxiliar na decisão do nível de *stock* a manter, há que destacar também o elevado número de SKU's vendidos pelos retalhistas, que torna a gestão de inventários mais complexa, sendo que para tomar decisões de reabastecimento eficazes, é imperativo a existência de sistemas informatizados de apoio à decisão (Zietsman & van Vuuren, 2022).

Os estudos mais recentes revelam que os métodos apoiados em tecnologias de Inteligência Artificial (IA), nomeadamente métodos de *Machine Learning* (ML) e *Deep Learning* (DL) desempenham um papel de interesse nesta área de estudo, dado que analisam de forma rápida grandes conjuntos de dados, tornando o processo de gestão de *stocks* mais flexível e eficiente (Albayrak Ünal et al., 2023). Os autores consideram a abordagem de *Reinforcement Learning* (RL) a mais comum e eficaz para problemas de inventários, sobretudo de controlo e otimização, por ser dinâmica em ambientes de mudança e ser capaz de lidar com problemas mais complexos. Por outro lado, os métodos de ML são normalmente usados em áreas de previsão e classificação da procura (Albayrak Ünal et al., 2023).

## *2.2. Machine Learning*

O *Machine Learning* é resultado do progresso dos estudos da Inteligência Artificial (IA) e das Redes Neuronais Artificiais (RNA). Domingos (2017) define IA como o conjunto de instruções e regras que formam um algoritmo, que processa informações e resolve problemas. Coppin (2004) amplia esta definição, descrevendo a IA como a capacidade das máquinas se adaptarem a novas situações, responder a perguntas e planear dispositivos. As redes neuronais por sua vez são modelos de ML inspirados na organização do cérebro humano e consistem na criação de uma rede de neurónios

interligados em camadas (Popescu et al., 2022). Cada neurónio de entrada processa e transmite a informação a um neurónio de outra camada, sendo que cada conexão tem um peso associado que é influenciado pela camada anterior, até que a "camada de saída" seja alcançada e seja possível calcular o erro entre a saída prevista e a saída real (Popescu et al., 2022). Além da sua capacidade de aprender através de exemplos e de generalizar, os principais benefícios associados às RNA devem-se à sua estrutura não linear e paralela, que lhe permite respetivamente, classificar padrões não lineares e tolerar a falha, pois caso um neurónio falhe, os efeitos na rede não serão significativos para o seu desempenho, dado que existe mais do que um caminho de ligação entre neurónios (Alves, 2016). Relativamente ao ML, este é considerado uma subcategoria da IA, em que o algoritmo aprende modelos, através de experiências e dados históricos, sendo capaz de funcionar bem em novas amostras e não apenas nos conjuntos de treino (Zhou, 2021) .

O foco de pesquisa de ML assenta no desenvolvimento e análise de sistemas de aprendizagem para melhorar o desempenho de tarefas, na simulação computacional de processos de aprendizagem humana e na análise teórica de métodos e algoritmos (Michalski et al., 2013). Assim, a construção de um modelo de ML robusto passa por 4 fases, nomeadamente selecionar os recursos, treinar o modelo, a validação do mesmo e a sua aplicação a novos dados, de modo a verificar se o modelo está em bom funcionamento para ser posteriormente adotado (Cielen et al., 2016) .

Atualmente, o *Machine Learning* está presente em diversos ramos da ciência da computação, mas também integra áreas como a psicologia, biologia, medicina, engenharia, entre outras. O motivo da crescente popularidade no ML prende-se com o aumento da produção de dados informáticos e a crescente necessidade de processar um grande volume de dados (Hall et al., 2016).

Os principais modelos de *Machine Learning* dividem-se consoante os princípios em que se baseiam e são resultado da qualidade e quantidade de dados disponíveis e do tipo de funcionamento dos algoritmos. Deste modo é possível descrever cada modelo da seguinte forma:

- ***Supervised Learning*:** este modelo é aplicado principalmente devido à natureza rica em dados rotulados, mas escassa de conhecimento dos problemas (Lu, 1990), tratando-se de "aprender com exemplos fornecidos por um supervisor externo experiente" (Sutton & Barto, 2014). Esta abordagem concentra-se na previsão de resultados e requer casos rotulados (Johnson et al., 2018), sendo que após assinalar o problema, os dados são identificados e pré-processados, para de seguida definir-se o algoritmo e o conjunto de treino e posteriormente avaliar a capacidade do algoritmo no conjunto de teste (Bonaccorso, 2017). Assim, é fundamental que o modelo desenvolva a capacidade de generalização e evite o *overfitting* – acontece quando o modelo ajusta-se ao conjunto de dados de treino, mas apresenta um mau desempenho quando aplicado a novos dados (Wuest et al., 2016). Bonaccorso (2017) aponta a deteção de spam e de padrões, a classificação de imagens e o processamento de linguagem natural como as aplicações mais comuns deste tipo de modelo.
- ***Unsupervised Learning*:** esta abordagem baseia-se na ausência de qualquer supervisor experiente e não necessita de observações rotuladas, sendo que o próprio algoritmo concentra-se em descobrir a estrutura e relações subjacentes no conjunto de dados (Johnson et al., 2018). Este método deve identificar clusters de acordo com a sua similaridade, sendo útil na deteção de valores similares ou discrepantes, na segmentação de objetos (utilizadores, filmes, entre outros) e na rotulagem automática (Bonaccorso, 2017).

- ***Reinforcement Learning*:** embora não existam supervisores reais, esta abordagem baseia-se no *feedback* (recompensa ou penalidade) fornecido pelo ambiente, necessário para entender se determinada ação realizada é positiva ou não, sendo que o agente tem de aprender com a sequência de ações, para tomar a decisão em termos de otimizar a recompensa máxima (Bonaccorso, 2017). Este modelo, segundo Johnson et al. (2018), baseia-se na aprendizagem por tentativa-erro, de modo a perceber quais as ações que trazem mais benefícios, sendo útil a sua aplicação na robótica, jogos e navegação.
- ***Deep Learning*:** este método é considerado por Lecun et al. (2015) como um exemplo de aprendizagem de representação com múltiplos níveis de abstração, obtidos através da composição de módulos simples. O princípio subjacente a este modelo caracteriza-se por calcular uma função objetivo que mede o erro entre o produto real e o desejado (Rynkiewicz, 2019). Goodfellow et al. (2016) refere o reconhecimento facial e de voz, os carros autónomos e os *chatbots* como os principais avanços da engenharia neste tipo de modelo.

## *2.3.* Estratégias de gestão de inventários baseadas em *Machine Learning*

Neste tópico são abordadas diferentes estratégias de gestão de inventários, baseadas em modelos de *Reinforcement Learning* e de *Deep Learning*, seguida de uma breve comparação das duas abordagens.

### *2.3.1. Reinforcement Learning*

O modelo de RL pode utilizar diferentes abordagens para otimizar a tomada de decisões sob incerteza, nomeadamente o Processo de Decisão de Markov (PDM) e os Jogos de Markov (JM), sendo que para ambos o sistema tem de possuir a propriedade de Markov, isto é, o estado futuro do sistema depende apenas do estado atual, não considerando os estados passados (Alsheikh et al., 2015).

O PDM e os JM são apresentados como modelos matemáticos para a tomada de decisões sequenciais no sistema, permitindo múltiplas decisões em vários períodos (Garcia & Rachelson, 2013). Neste tipo de modelos podem-se distinguir as seguintes características: os estados que são observados pelo agente e representam todos os elementos relevantes do problema em questão (Van Otterlo & Wiering, 2012), as ações que são executadas para controlar o estado do sistema, a função de transição que ocorre quando se aplica uma ação num estado (Puterman, 2014), a definição da função de recompensa usada para controlar o sistema, sendo o seu objetivo realizar ações que otimizem as recompensas (Van Otterlo & Wiering, 2012) e a política definida pelo conjunto de todas as regras de decisão (Li et al., 2022). Assim, estes modelos surgem como uma forma de modelar processos onde as transições entre estados são probabilísticas, executando ações com uma recompensa associada, para de seguida o sistema transitar para um novo estado (Alsheikh et al., 2015).

Em ambientes de um único agente, ou seja, em ambientes onde apenas uma entidade autónoma (sistema, programa de computador, *robot*, entre outros) tem a capacidade de realizar ações e tomar decisões para atingir objetivos, é apropriado aplicar o PDM (Sutton & Barto, 2014), enquanto os JM apresentam uma visão mais ampla que inclui múltiplos agentes adaptativos com objetivos interativos ou concorrentes, sendo que as suas ações afetam indiretamente as recompensas de

outros agentes (Littman, 1994). De seguida, segue a Figura 1, onde é possível observar a diferença entre as duas abordagens, sendo que "a" representa as ações e "s" e "r" representam os estados e a recompensa, respetivamente.

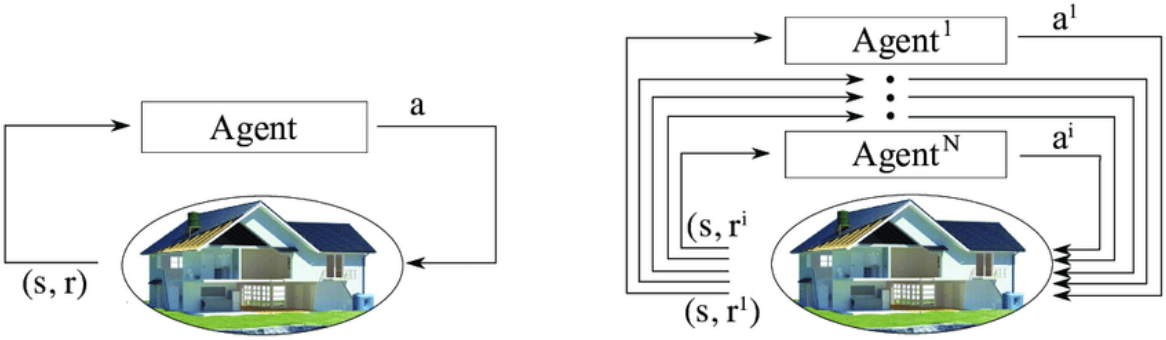


Figura 1 - Diferença entre o Processo de Decisão de Markov (à esquerda) e o Jogo de Markov (à direita) (Blad et al. (2021))

É de salientar que ao contrário do que acontece no PDM, nos JM pode não existir uma política ótima que maximize a soma esperada da recompensa com desconto, sendo necessário definir o comportamento ótimo de um agente como sendo o seu comportamento num equilíbrio de Nash (Littman, 1994). O equilibro de Nash representa um ponto na matriz de recompensas, onde nenhum agente possui vantagens em mudar de estratégia, considerando que as opções dos restantes agentes permanecem constantes (Fudenberg & Tirole, 1991).

Posteriormente, em ambas as abordagens, com base no problema deve-se optar pela perspetiva do algoritmo baseada em modelo ou sem modelo, sendo que os métodos baseados em modelos dependem do planeamento, enquanto os restantes dependem da aprendizagem (Barrett et al., 2013).

O estudo realizado por Leluc et al. (2023) aborda a técnica de RL com múltiplos agentes e múltiplos produtos, para resolver o problema de gestão de inventários numa cadeia de abastecimento com um único escalão numa linha de produção. Os autores consideraram a procura e os prazos de entrega variáveis estocásticas, sendo o objetivo esperado maximizar a recompensa com desconto esperada.

O método utilizado para modelar o problema foram os Jogos de Markov, que segundo Leluc et al. (2023) oferecem uma estrutura mais realista da gestão de inventários, capaz de identificar as dependências dinâmicas e as relações cooperativas entre produtos. Nesse sentido, o estado do agente é composto pelo nível de inventário, a quantidade de reposição, o *lead time* e os *backlogs*, consequência do custo de escassez associado ao nível da procura exceder o nível de inventário, enquanto a ação de cada agente diz respeito à quantidade do pedido (Leluc et al., 2023). É de salientar ainda que cada agente incorre em custos de stock associados à função de recompensa, nomeadamente custos de pedido, custos de manutenção e custos de escassez (Leluc et al., 2023).

Posto isto, os autores utilizaram o algoritmo *Proximal Policy Optimization* (PPO), sendo as principais vantagens apontadas por Schulman et al. (2017), a simples implementação, a confiabilidade e a estabilidade do algoritmo que evita oscilações na atualização de políticas. Este algoritmo é livre de modelo e utiliza uma estrutura *actor-critic*, onde existem dois componentes, o *actor* (rede de

política) que decide a melhor ação para um dado estado e o *critic* (função de valor) que avalia a política que está a ser criada pelo *actor* (Cunha et al., 2020).

A estratégia de RL utilizada por Leluc et al. (2023) é comparada com a abordagem *MinMax* onde o controlador faz a encomenda quando o nível de inventário se encontra abaixo do *stock* de segurança e com a abordagem *Oracle*, onde os agentes otimizam a quantidade do pedido observando valores futuros de diferentes variáveis. Os resultados indicaram que os agentes baseados no algoritmo PPO apresentam o melhor desempenho no que diz respeito aos custos cumulativos médios e ao número médio de *stock* em falta, sendo que o nível de *stock* dos agentes PPO está sempre acima da procura de modo a evitar ruturas de *stock* (Leluc et al., 2023).

Kosasih & Brintrup (2021) apresentam um estudo para otimizar o *stock* de segurança e a quantidade de encomenda em cada escalão da cadeia de abastecimento, composta por um retalhista, um armazém e uma fábrica. Os autores utilizaram o Processo de Decisão de Markov para modelar o problema e recorreram, de seguida, a um modelo analítico de base para posteriormente comparar o seu desempenho com o desempenho de 3 algoritmos de RL:

- *Tabular Q-Learning*: neste método, os autores definem uma tabela para armazenar os valores de qualidade (*Q-values*) para cada entrada de estado-ação e cada agente toma decisões com base nos *Q-values*, sendo o objetivo otimizar o *stock* de segurança (Kosasih & Brintrup, 2021).
- *Temporal Difference Advantage Actor-Critic* (A2C): este algoritmo utiliza redes neuronais para maximizar recompensas e aprender políticas e valores de estado. O *actor* decide a quantidade do pedido e o *critic* estima o nível do *stock* de segurança (Kosasih & Brintrup, 2021).
- *Multi-Agent Temporal Difference Advantage Actor-Critic* (*Multi-Agent* A2C): esta abordagem é uma extensão da anterior (A2C), onde cada agente apenas perceciona o seu próprio inventário e a sua procura (Kosasih & Brintrup, 2021).

Os resultados deste estudo salientam que embora os algoritmos de RL apresentem resultados ligeiramente abaixo do ideal em comparação com os resultados do modelo analítico, estes têm capacidade de otimizar o nível do *stock* de segurança e a quantidade a encomendar, sendo que este último parâmetro não é possível cumprir com o método clássico por ser definido *a priori*. Considerando as abordagens de RL, o algoritmo *Multi-Agent* A2C foi o que apresentou maior eficácia na otimização do nível do *stock* de segurança, com uma aprendizagem mais rápida (Kosasih & Brintrup, 2021). Os autores apontam como fator positivo a simplicidade de implementar a técnica *Tabular Q-Learning* e a boa performance em problemas com espaços de ação finitos, o que se torna mais difícil em espaços de ação com muitos estados, pois a tabela precisa de ser atualizada frequentemente.

A pesquisa conduzida por Yin et al. (2022) desenvolve um cenário de gestão de inventários com procura estocástica e não estacionária, que tem em consideração o impacto das vendas durante períodos sazonais, períodos de promoções e fins-de-semana.

Os autores criaram o ambiente em *Python*, modelando o problema como PDM, sendo o espaço de estados composto por vetores de estado, incluindo o inventário fechado do último período, se é fim-de-semana e se é período de promoção. A ação diz respeito à quantidade do pedido, que varia entre zero e o limite máximo da capacidade de inventário, enquanto a recompensa corresponde ao valor monetário líquido de tomar a ação (Yin et al., 2022).

De seguida, os autores implementaram o algoritmo SARSA (*State-Action-Reward-State-Action*) em dois contextos, assumindo inicialmente que a procura segue uma distribuição normal e posteriormente uma distribuição de Poisson. O algoritmo SARSA tem como objetivo aprender o valor Q (recompensa) de uma ação de um determinado estado, para obter uma tabela Q otimizada e atualizada com base na recompensa obtida pela interação com o ambiente (Yin et al., 2022). A sua aplicação mostrou que em ambas as distribuições, esta abordagem é capaz de captar potenciais mudanças desconhecidas no ambiente e ajustar a sua estratégia às mesmas, apresentando uma convergência mais rápida para um nível médio de recompensa alto (Yin et al., 2022).

### *2.3.2. Deep Learning*

As estratégias baseadas em *Deep Learning*, à semelhança do RL também utilizam o Processo de Decisão de Markov para modelar problemas sob incerteza.

Madeka et al. (2022) estudaram uma estratégia onde implementaram uma abordagem de *Deep Reinforcement Learning* (DRL) baseada no modelo de *Direct Backpropropagation* (DB) para encontrar o nível de inventário ótimo num sistema de controlo de revisão periódica, que lida com prazos de entrega estocásticos, vendas perdidas, procura correlacionada e correspondência de preços.

Para tal, os autores utilizaram o Processo de Decisão Interativo (PDI) que se diferencia do PDM pela sua abrangência, não tendo a necessidade de obedecer à propriedade de Markov. A construção do PDI passou por duas etapas. Inicialmente, Madeka et al. (2022) descreveram os processos aleatórios, influenciados por ações fora do seu controlo, nomeadamente a procura, os prazos de entrega, a receita das vendas e o custo de compra dos produtos, para de seguida descrever o processo de tomada de decisão, dependente das informações dos processos referidos acima. Esta segunda etapa engloba as ações possíveis de realizar para cada produto, o *stock* disponível, a política ou o conjunto de políticas e o vetor "histórico" – que permite verificar a evolução da procura e não apenas o estado atual. O PDI destaca também a função de recompensa que depende da receita das vendas em cada época de decisão, do custo de compra e do *stock* no início do período (Madeka et al., 2022).

Nos casos onde todas as variáveis aleatórias são totalmente observadas, a política do produto depende apenas do histórico do próprio produto, representando uma melhoria exponencial em relação à complexidade da amostra de um problema. No entanto, a natureza deste PDI caracteriza-se pela necessidade de imputar dados que estão naturalmente censurados para recuperar a trajetória do vetor "histórico" de qualquer produto, através do contexto adicional disponível, isto é, pela descrição do produto, através de dados macroeconómicos, entre outros, de modo a criar uma previsão precisa (Madeka et al., 2022).

Desta forma, os autores criaram técnicas de correção de dados históricos através de um simulador diferenciável que utiliza o algoritmo *Direct Backpropagation*. O DB é aplicado como uma técnica de otimização para treinar redes neuronais que representam políticas de controlo e que maximizam a recompensa acumulada ao longo do tempo no simulador, sendo este utilizado apenas nos processos externos, como é o caso da procura não observada durante períodos de rutura de *stock*, prazos de entrega censurados durante semanas em que não foi realizado nenhum pedido, entre outros (Madeka et al., 2022). Assim, o simulador atua em 3 níveis de inventário inicial diferentes

para estudar o comportamento dos agentes, sendo que no primeiro caso o retalhista não tem *stock*, o segundo caso inicia-se com o *stock* histórico mantido pelo retalhista e no último cenário, o inventário é inicializado onde a antiga política parou. Deste modo, uma mudança nas entradas afeta a saída do sistema, sendo que em cada momento de decisão, o agente tem acesso ao nível de inventário atual, às ações realizadas anteriormente, aos recursos da série temporal (distâncias até feriados), aos recursos estáticos do produto (grupo e descrição do produto) e à economia do produto (preço e custo) (Madeka et al., 2022).

A estratégia de DRL é comparada com algoritmos de RL como o *Augmented Random Search*, *Soft Actor-Critic* e *Asynchronous Advance Actor Critic* e com políticas de linha de base nomeadamente a *Oracle*, *Myopic Approximation*, *Newsvendor* e *Planning Horizon Normalized* (PHN):

- *Oracle*: esta política representa um cenário idealizado, onde são conhecidas a procura futura, o preço, o custo e o prazo de entrega de todos os produtos;
- *Myopic Approximation:* esta política solicita um valor ao longo de um horizonte de tempo de entrega fixo, sem considerar a tendência da procura a longo prazo, custos de armazenamento e outros fatores que afetam o desempenho do sistema;
- *Newsvendor*: esta política otimiza a procura e os prazos de entrega e encomenda um valor ao longo do horizonte de planeamento esperado, considerando a condição de estacionariedade como encerramento, ou seja, o processo de tomada de decisão é interrompido quando o sistema atinge um estado em que a procura e os prazos de entrega atingem um padrão considerado estável e sem necessidade de pedidos adicionais.
- *Planning Horizon Normalized*: esta política é uma atualização da *Newsvendor*, que absorve informação extra sobre a distribuição dos prazos de entrega dos fornecedores, sendo que a mediana destes prazos de entrega será multiplicada pelo valor normalizado dado pela política *Newsvendor* para obter o valor de compra em cada período de revisão.

Após a execução de todos os modelos referidos em ambiente simulado e considerando os 3 casos de inicialização abordados, foi testado o seu desempenho em termos de recompensa cumulativa no final do período de teste, sendo que no último cenário, a política utilizada foi a *Newsvendor*. Tal como seria de esperar, a política *Oracle* atingiu a maior recompensa nos 3 cenários. Relativamente às políticas de base e excetuando a *Oracle*, a política PHN é a que apresenta maior recompensa cumulativa. No entanto, considerando todas as políticas, as baseadas em RL apresentam um desempenho superior e estável nas diferentes inicializações, quando comparado com as políticas consideradas tradicionais (Madeka et al., 2022).

Os autores compararam posteriormente, o método atual do sistema (*Newsvendor*) de uma das maiores cadeias de abastecimento do mundo com a sua estratégia de DRL, tendo obtido resultados bastante favoráveis na diminuição do *stock* em aproximadamente 12% sem comprometer a receita total.

### *2.3.3.* Comparação dos modelos de *Reinforcement Learning* e *Deep Learning*

As estratégias de *Reinforcement Learning* e *Deep Learning* têm sido amplamente aplicadas para melhorar a eficiência da gestão de inventários.

O *Reinforcement Learning* tem a capacidade de aprender por tentativa erro e de lidar com ambientes dinâmicos e de incerteza, generalizando o seu conhecimento para que os algoritmos sejam aplicados em diferentes contextos (Johnson et al., 2018). Por outro lado, este modelo pode apresentar um tempo e um custo computacional elevado em ambientes complexos, sendo caracterizado também pela sensibilidade dos seus parâmetros (Sutton & Barto, 2014) .

O *Deep Learning* é importante para modelar problemas complexos com dados históricos não lineares, sendo aplicável na previsão da procura e na otimização dos inventários, de forma a determinar o nível ótimo de *stock* com o menor custo (Zhang et al., 2021). Este modelo apresenta um bom desempenho em grandes conjuntos de dados, no entanto necessita de uma grande quantidade de dados para treinar o modelo, o que pode limitar o seu uso em determinadas áreas (Mnih et al., 2013). É ainda de salientar a dificuldade de interpretar os resultados obtidos por algoritmos de DL, sobretudo em ambientes com redes neuronais profundas (Zhang et al., 2021).

Desta forma, os autores consideram que o DL é utilizado em tarefas muito específicas, enquanto o RL apresenta uma maior flexibilidade em adaptar-se a diferentes ambientes. A escolha de utilizar um algoritmo de RL ou de DL deve depender das características do problema a resolver, sendo que em muitos casos da literatura os autores estão a optar por combinar as duas abordagens, considerando o *Deep Reinforcement Learning* um novo modelo de Machine Learning ((Mnih et al., 2013); (Zhang et al., 2021); (Goodfellow et al., 2016)).

página propositadamente em branco

# 3. MÉTODOS E APLICAÇÃO

Este capítulo é inicializado pela descrição, contextualização e análise da situação atual da empresa, assim como das suas principais limitações. Posto isto, é possível apresentar a proposta de solução para resolver o problema abordado. Para tal, são necessárias as fases de seleção de dados, seguida de análise e preparação dos mesmos para finalmente apresentar o modelo proposto. Por fim, é realizada uma breve síntese do processo proposto, que inclui as 5 fases necessárias e a lógica associada para atingir a solução.

## 3.1. Estado atual

A operação de reabastecimento das lojas está diretamente ligada à eficiência das operações internas do centro de distribuição, sobretudo ao processo de *picking*. Embora este envolva a separação e o envio dos artigos para as lojas, não encerra o ciclo logístico, mas dá início a uma nova fase, o abastecimento das posições de *picking*, que garante o fluxo contínuo de artigos em armazém. Assim, segue a contextualização da situação atual da empresa nas 3 fases referidas.

### 3.1.1. Reabastecimento das lojas

O reabastecimento das lojas é realizado de segunda a sexta-feira com base no cálculo diário do nível de *stock* mínimo e máximo para cada artigo de cada loja. Assim, seguem as fórmulas de cálculo que a empresa utiliza para ambos:

$$\mathbf{S_{min}} = \sqrt{t} \times \sigma \times Z$$

t = tempo do *stock* em loja

σ = desvio padrão dos 30 dias de vendas mais recentes

Z = nível de serviço

$$\mathbf{S_{max}} = \bar{x} \times t + S_{min}$$

$\bar{x}$ = média dos 30 dias de vendas mais recentes

t = tempo do *stock* em loja

$S_{min}$ = *stock* mínimo

O tempo de *stock* em loja utilizado no cálculo do nível de *stock* mínimo e máximo é definido estrategicamente pela empresa. Por outro lado, o nível de serviço varia de acordo com a classificação ABC de cada artigo, em valor e em quantidade, e é retirado de uma tabela de distribuição normal.

Na Figura 2 pode-se observar a distribuição das vendas em valor nos últimos 30 dias, segundo o diagrama de Pareto.

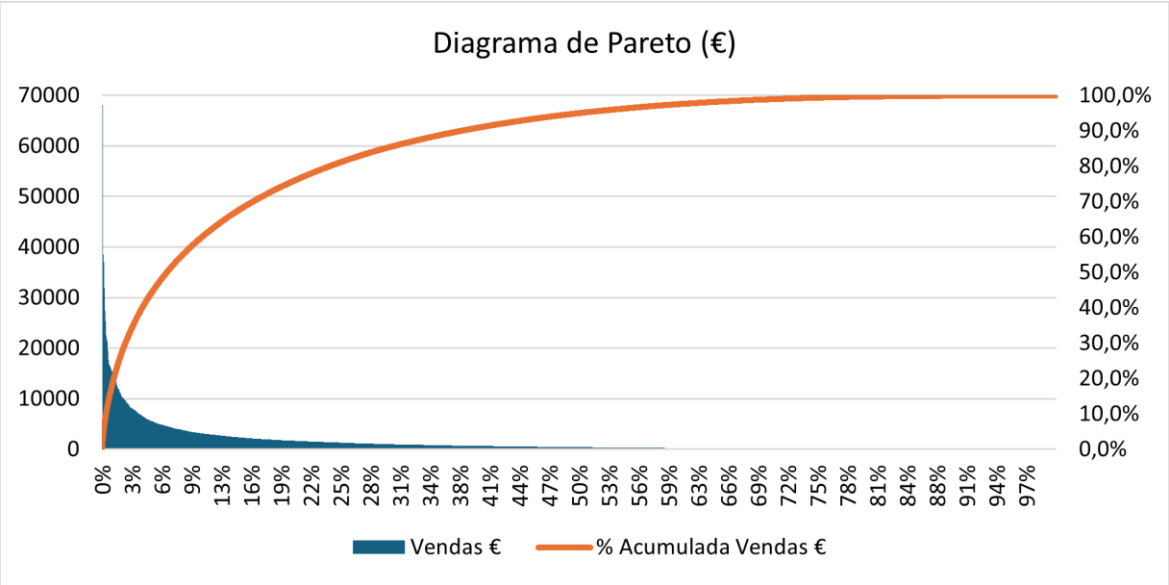


Figura 2 - Distribuição das vendas nos últimos 30 dias (€)

Na Tabela 1, verifica-se que os artigos das classes A e B representam cerca de 24% dos artigos da empresa e correspondem a 80% da sua faturação, enquanto a classe C reflete os restantes 20% do volume de vendas.

| Classe | Nº artigos | Peso artigos | Volume vendas | Peso vendas |
|---|---|---|---|---|
| A | 89 | 1,2% | 2 016 796 € | 20% |
| B | 1657 | 22,6% | 6 103 878 € | 60% |
| C | 5582 | 76,2% | 2 031 177 € | 20% |
| Total | 7328 | 100% | 10 151 851 € | 100% |

Tabela 1- Distribuição das vendas nos últimos 30 dias segundo análise ABC (€)

Uma vez que existem artigos de baixo valor, mas que apresentam uma grande relevância a nível de quantidade, como é o caso dos oxidantes para tintas de cabelo (essencial para os clientes profissionais), a empresa considera também a análise da classificação ABC em quantidade na atribuição do nível de serviço. Assim, segue a distribuição das vendas em quantidade na Figura 3.

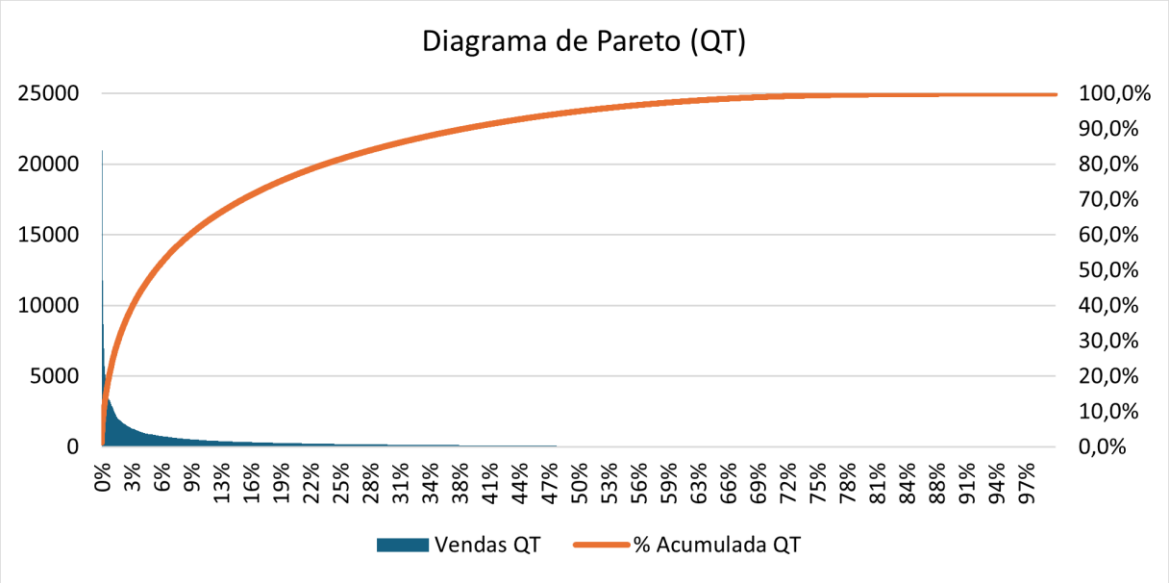


Figura 3 - Distribuição das vendas nos últimos 30 dias (QT)

Os dados extraídos da análise ABC em quantidade são apresentados na Tabela 2.

| Classe | Nº artigos | Peso artigos | Volume vendas | Peso vendas |
|---|---|---|---|---|
| A | 57 | 0,8% | 350152 | 20% |
| B | 1655 | 22,6% | 1057996 | 60% |
| C | 5616 | 76,6% | 352104 | 20% |
| Total | 7328 | 100% | 1760252 | 100% |

Tabela 2 - Distribuição das vendas nos últimos 30 dias segundo análise ABC (QT)

O nível de serviço de cada artigo é atribuído através do cruzamento das duas análises ABC referidas. Os valores em uso pela empresa são atualizados diariamente e estão apresentados na Tabela 3, sendo que a taxa de cobertura é consequência da decisão de negócio.

| Classe € | Classe QT | Z | Taxa cobertura | Nº artigos | Peso artigos | QT vendida | Volume vendas |
|---|---|---|---|---|---|---|---|
| A | A | 3,10 | 99,9% | 25 | 0,3% | 186904 | 740 007 € |
| B | A | 3,10 | 99,9% | 30 | 0,4% | 157103 | 187 713 € |
| A | B | 2,80 | 99,7% | 62 | 0,8% | 100525 | 1 249 012 € |
| A | C | 2,40 | 99,2% | 2 | 0,0% | 288 | 35 107 € |
| C | A | 2,40 | 99,2% | 2 | 0,0% | 7597 | 1 506 € |
| B | B | 2,39 | 99,2% | 1208 | 16,5% | 765156 | 4 790 146 € |
| B | C | 2,33 | 99,0% | 419 | 5,7% | 61377 | 1 116 561 € |
| C | B | 1,80 | 96,4% | 385 | 5,3% | 190947 | 325 891 € |
| C | C | 1,60 | 94,5% | 5195 | 70,9% | 290355 | 1 705 908 € |
| Total | | | | 7328 | 100% | 1760252 | 10 151 851 € |

Tabela 3 - Nível de serviço nos últimos 30 dias

Posto isto, quando o nível de *stock* atual se encontra abaixo do *stock* mínimo, é gerado um novo pedido de reabastecimento, sendo que as lojas situadas em Portugal continental recebem o pedido no prazo de 1 dia, em Espanha o prazo de entrega é de 2 dias e na Madeira 7 dias. A quantidade de reabastecimento solicitada é obtida através da diferença entre o *stock* máximo e a quantidade

disponível "real" de *stock* em loja, ou seja, ao *stock* atual subtrai-se o *stock* reservado e adiciona-se o *stock* em trânsito, como se verifica na fórmula apresentada abaixo:

$$\textbf{Pedido Reab.} = S_{max} - (S - S_{res} + S_{tr})$$

S = *stock* em loja

$S_{res}$ = *stock* reservado em loja

$S_{tr}$ = *stock* em trânsito (por validar)

É de realçar que há artigos que apresentam um múltiplo de envio. Este tipo de artigos é enviado na caixa ou embalagem do fornecedor e por isso mesmo quando existe um pedido de reabastecimento podem ser enviados em múltiplos de por exemplo 6 ou 12, conforme o artigo em questão. Neste tipo de artigos, a ordem de reabastecimento é efetuada automaticamente para o múltiplo por excesso.

Na Figura 4 estão representadas as linhas originais e as linhas efetivamente separadas para o reabastecimento das lojas durante 30 dias. Uma linha de pedido representa as necessidades de um SKU para determinada loja. A capacidade média diária do centro de distribuição situa-se geralmente acima das 20000 linhas, mas varia com base no número de trabalhadores.

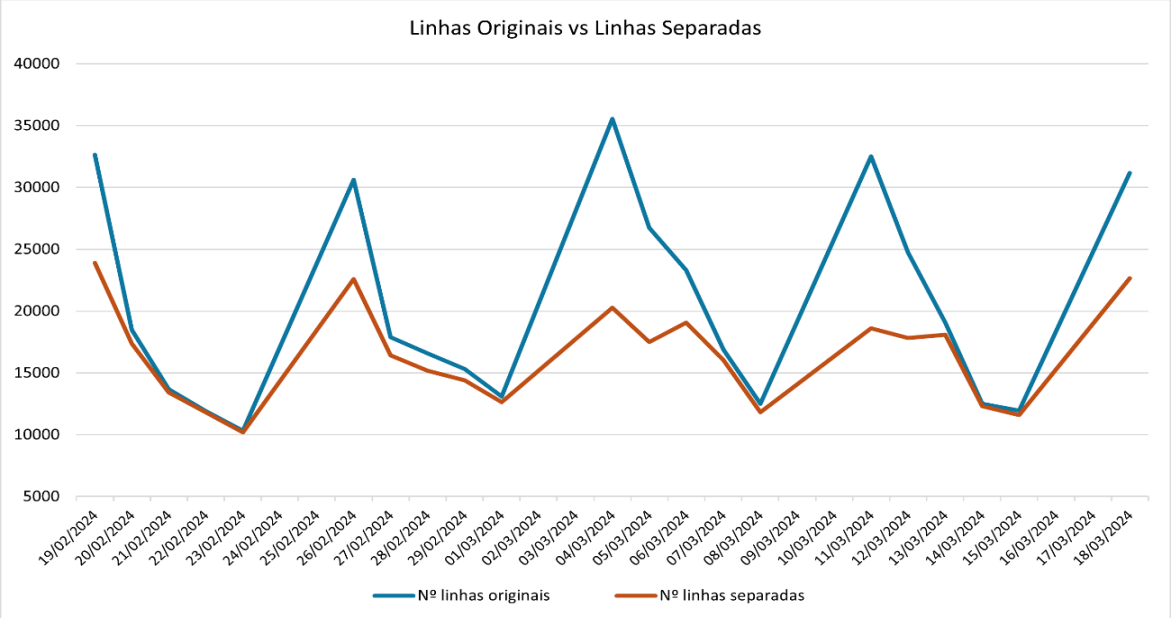


Figura 4 - Quantidade de linhas originais e separadas para pedidos de reabastecimento durante um mês

Os picos existentes acontecem às segundas-feiras, visto que durante o fim-de-semana não existem entregas de artigos nas lojas e por isso as necessidades de sábado e domingo acumulam, traduzindo-se numa maior necessidade de reabastecimento à segunda-feira. O facto deste ser por norma o dia de encerramento dos estabelecimentos dos clientes profissionais, gera também um pico nas vendas, que poderá afetar as necessidades das lojas de terça-feira. Nos restantes dias, o número de linhas tende a diminuir, com algumas exceções nas linhas efetivamente separadas. Este tipo de exceções verifica-se na semana de 11/03, onde o pico das linhas separadas à segunda-feira é menos evidente quando comparado com as restantes semanas, e onde há uma diminuição do

número de linhas na terça-feira e posteriormente um ligeiro aumento de linhas no dia seguinte, que acontece pela possibilidade do corte de linhas em armazém. Assim, a diferença entre a quantidade de linhas separadas e as linhas originais deve-se a este corte, ou seja, o sistema corta linhas consoante a capacidade diária do centro de distribuição, e embora seja um corte sem critério, obedece obrigatoriamente a 3 regras, não são cortadas linhas que apresentam um *stock* de zero unidades, linhas que os artigos pertençam à classe A e linhas que abranjam artigos com *stock* reservado em loja.

Desta forma, o abastecimento das lojas é efetuado todos os dias, de segunda a sexta-feira, contudo apenas 80% das lojas recebem realmente produto durante esses 5 dias. As restantes apresentam uma programação específica, por exemplo de 2 dias por semana, terça e quinta-feira, sendo que este planeamento é efetuado com base na faturação e nas necessidades das lojas. Nesse sentido, é fundamental estudar o custo associado a esta variável.

Os custos de transporte representam 35,8% dos custos totais de reabastecimento às lojas e variam consoante o peso do pedido de cada loja como se verifica na Tabela 4.

| Escalões Peso (Kg) | Valor (€) |
|---|---|
| 1 a 10 | 4,38 |
| 11 a 20 | 5,56 |
| 21 a 30 | 6,79 |
| 31 a 40 | 7,81 |
| 41 a 50 | 9,88 |
| 51 a 60 | 11,73 |
| 61 a 70 | 12,98 |
| 71 a 80 | 14,76 |
| 81 a 90 | 15,93 |
| 91 a 100 | 17,05 |
| 101 a 125 | 19,84 |
| 126 a 150 | 27,16 |
| 151 a 175 | 31,82 |
| 176 a 200 | 35,81 |
| 201 a 250 | 41,87 |
| 251 a 300 | 50,02 |
| 301 a 350 | 58,16 |
| 351 a 400 | 67,30 |
| 401 a 450 | 75,95 |
| 451 a 500 | 84,58 |
| 501 a 600 | 98,17 |
| 601 a 700 | 115,73 |
| 701 a 800 | 133,36 |
| 801 a 900 | 150,03 |
| 901 a 1000 | 166,66 |
| Mais de 1000 | 150,74 + 0,14/Kg |

Tabela 4 - Preço de transporte (€)

Nos últimos 30 dias foram efetuados 2.100 pedidos que se traduziu num custo total de 41.220,52€.

Analisando o comportamento isolado da loja 036 na Figura 5, verifica-se que esta segue o mesmo padrão acima descrito, ou seja, evidenciam-se os picos no número de linhas separadas às segundas-feiras e a quantidade de artigos correspondentes às linhas separadas, sendo que há casos onde para um menor número de linhas separadas, a quantidade total correspondente de artigos

separados pode ser superior. Além disso, é possível observar o peso total diário dos artigos separados para a loja 036, que consequentemente afeta o preço de transporte.

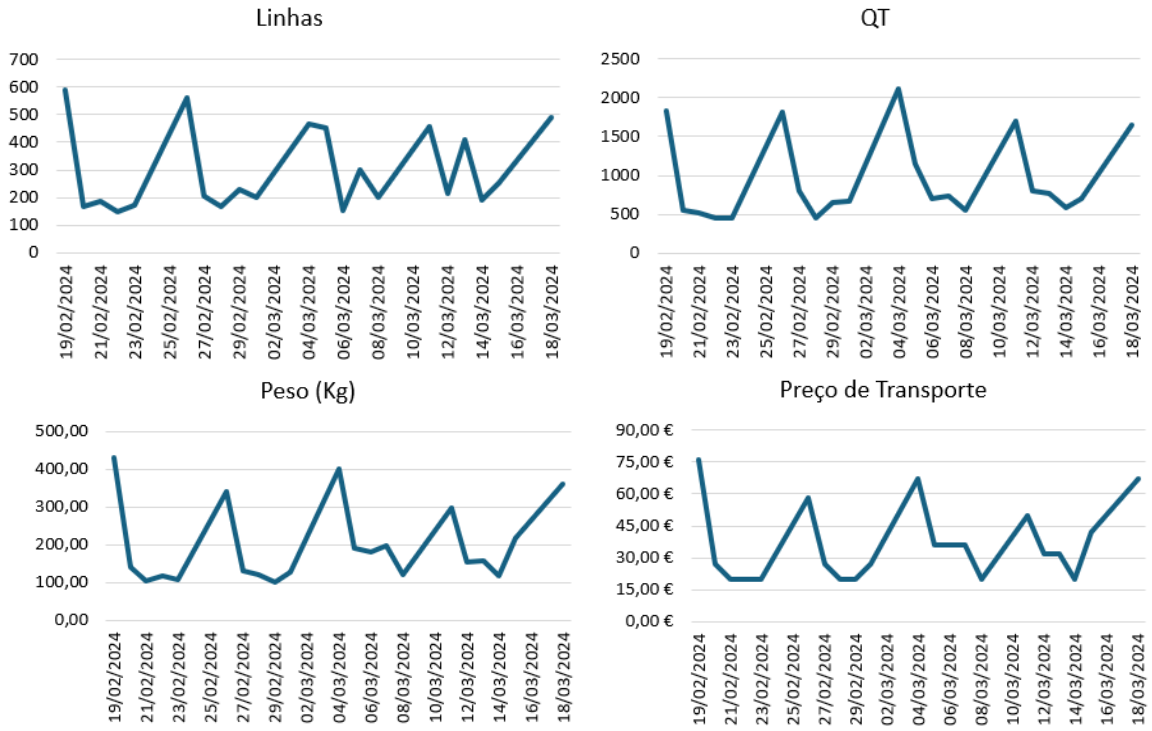


Figura 5 - Linhas separadas, quantidade, peso e preço de transporte para reabastecimento da loja 036 durante um mês

Nas figuras seguintes são apresentados 4 casos representativos do comportamento de reabastecimento das lojas, nomeadamente da loja 036. É de salientar que em nenhum caso se verificou a reserva de *stock* em loja.

A Figura 6 apresenta o desempenho do artigo X1063, classificado como um artigo AA, ou seja, é um artigo de alta importância tanto em valor, como em quantidade. Assim, após o artigo atingir o ponto de reabastecimento, isto é, o ponto em que o *stock* atual (de início do dia) se situa abaixo do nível de *stock* mínimo, é gerado um novo pedido de reposição, que chega à loja imediatamente no dia seguinte.

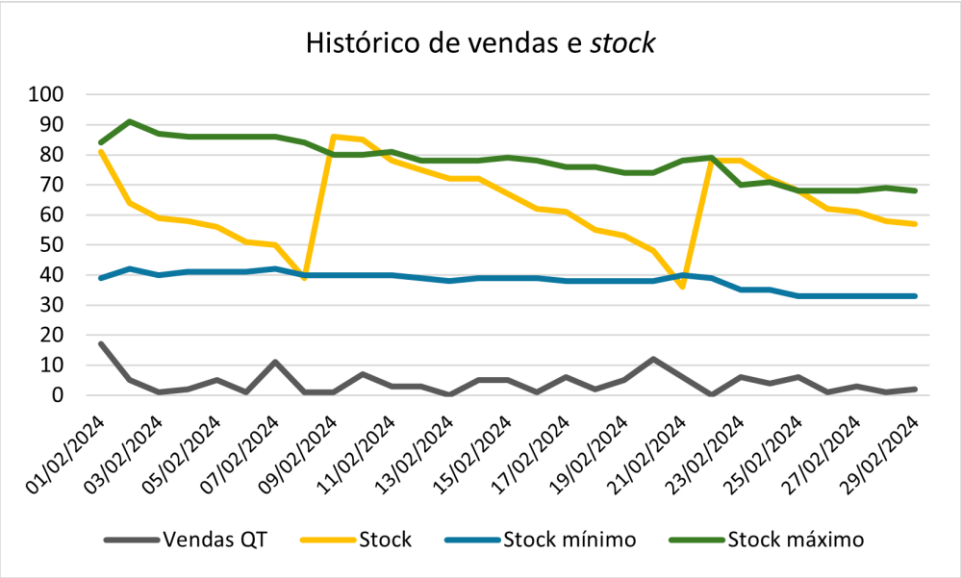


Figura 6 - Histórico de vendas e *stock* do artigo X1063 da loja 036 em fevereiro

No entanto, o mesmo não acontece com o artigo X1815, que embora seja um artigo AA, apresenta um comportamento ligeiramente diferente. Como se verifica na Figura 7, o artigo desceu ao ponto de reabastecimento a um domingo, dia 04/02/2024, e por isso mesmo a ordem de reposição apenas foi considerada na segunda-feira seguinte, sendo que o *stock* chegou à loja após um dia, ou seja, na terça-feira, dia 06/02/2024.

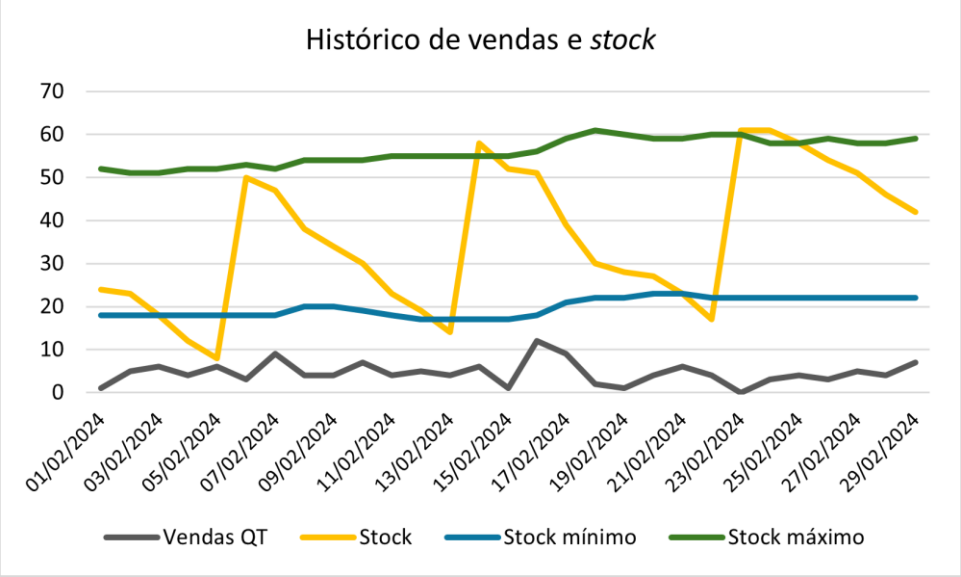


Figura 7 - Histórico de vendas e *stock* do artigo X1815 da loja 036 em fevereiro

Na Figura 8 é possível observar o desempenho do artigo X1432, que gera 3 ordens de reabastecimento ao longo do mês.

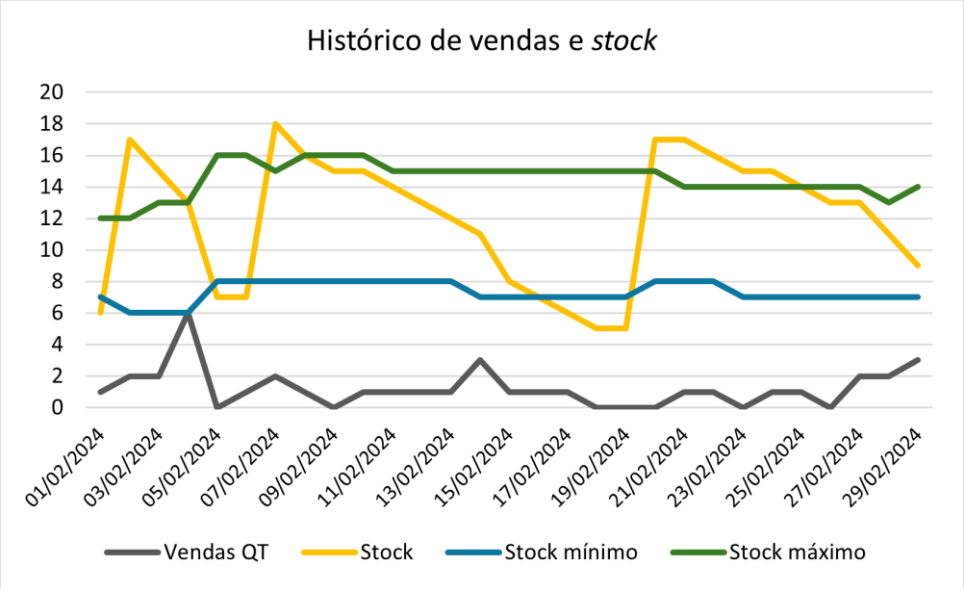


Figura 8 - Histórico de vendas e *stock* do artigo X1432 da loja 036 em fevereiro

À semelhança do que acontece com o artigo anterior, no último pedido de reposição, o artigo X1432 desce ao ponto de reabastecimento a um sábado, dia 17/02/2024, e como tal, a linha de reposição é gerada na segunda-feira, tendo chegado à loja no dia seguinte, dia 20/02/2024.

Enquanto na segunda ordem de reabastecimento, o *stock* encontra-se abaixo do nível de *stock* mínimo na segunda-feira, dia 05/02/2024, sendo gerado um novo pedido. No entanto, esse pedido não foi satisfeito e por isso, no dia seguinte, 06/02/2024 há uma nova ordem de reposição que abastece a loja no dia 07/02/2024. A falta de reposição inicial aconteceu por este artigo ser classificado como BB e por apresentar um *stock* maior do que zero unidades, o que possibilitou o corte desta linha de reabastecimento. Tal como já foi referido, este corte acontece pela capacidade diária do centro de distribuição ser menor do que as suas necessidades.

O comportamento do artigo X1017 representado na Figura 9, apresenta um momento de rutura de *stock* no dia 20/02/2024. Assim, no dia anterior, segunda-feira, o *stock* já se encontrava abaixo do nível de *stock* mínimo, no entanto por ser um artigo BB e contar com um *stock* maior do que zero, o pedido de reposição foi cortado. No dia 20/02/2024, há rutura de stock e é gerada uma nova linha de reabastecimento, que chega à loja no dia seguinte.

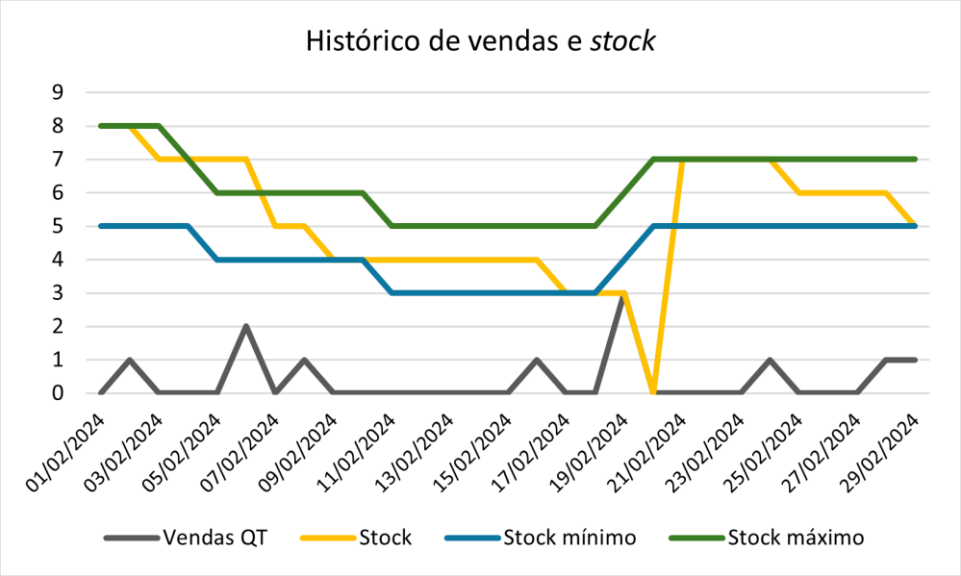


Figura 9 - Histórico de vendas e *stock* do artigo X1017 da loja 036 em fevereiro

Após a análise dos diferentes casos, verifica-se a necessidade de antecipar a decisão de separar os artigos, não só para balancear as necessidades com a capacidade diária do centro de distribuição, mas também para não haver casos de rutura de produto durante os fins-de-semana ou até durante a semana (exemplo do artigo X1017), que podem resultar em perda de vendas para a empresa. Como já foi referido, o artigo X1017 entrou em rutura na terça-feira, sendo que na segunda-feira o seu *stock* já se encontrava abaixo do nível de *stock* mínimo, mas por ser um artigo BB teve o seu pedido de reposição cortado. Se este pedido inicial tivesse sido satisfeito, o artigo não teria entrado em rutura.

### 3.1.2. Processo de *picking*

A operação de *picking* inicia-se com a criação do pedido de reabastecimento no sistema. De manhã, o responsável verifica o número de operadores disponíveis e quantifica o número de linhas/pedidos que será possível separar, considerando que cada operador separa em média 180 linhas por hora e que são contabilizadas 6 horas efetivas de separação, o que se traduz em 1080 linhas diárias por trabalhador. No que diz respeito aos trabalhadores temporários, contratados sobretudo nos períodos de julho e agosto, por motivo de férias dos *pickers*, a produtividade destes situa-se nas 130 linhas por hora. Estes valores foram testados anteriormente e adaptados, atendendo que há momentos em que o operador tem menos produtividade seja por num corredor apenas recolher um artigo ou por ser um momento de pausa. Na Figura 10 está representada a produtividade real por trabalhador e por hora nos primeiros 5 meses do ano de 2024.

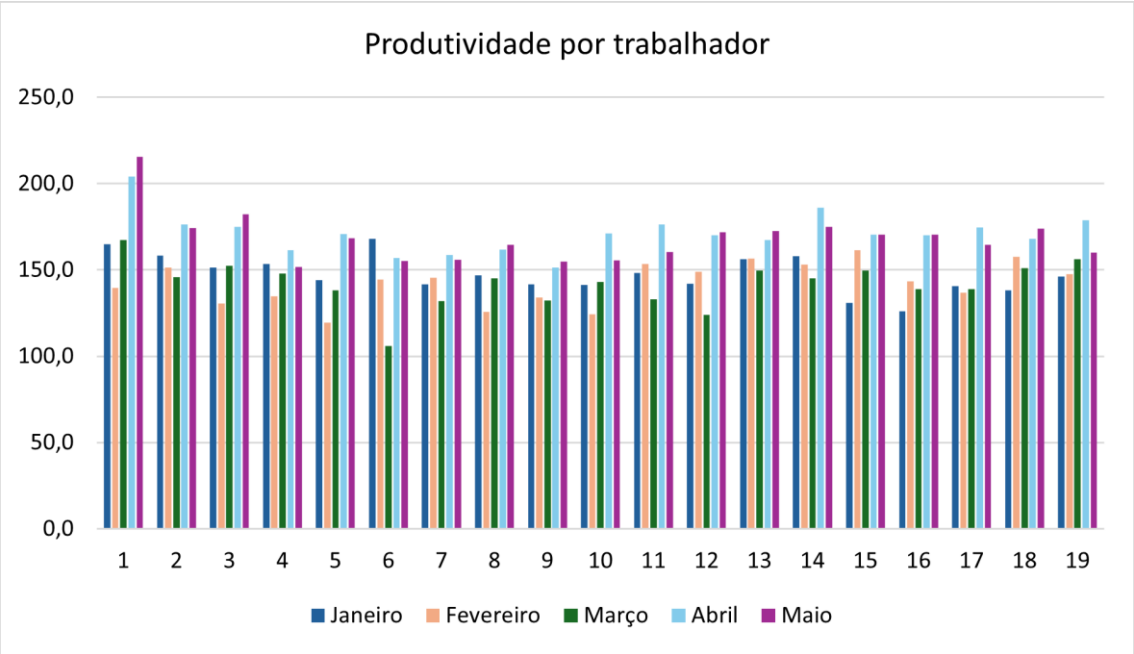


Figura 10 - Produtividade real (em linhas) por trabalhador e por hora nos primeiros 5 meses de 2024

Apesar destes valores de referência, o responsável é que toma a decisão sobre a capacidade máxima do centro de distribuição consoante a sua perspetiva, ou seja, este pode considerar que não é possível cada trabalhador separar 1080 linhas em determinado dia e por isso atribuir um valor menor ou maior. Nesse sentido, o programa gera uma estimativa do valor total de linhas e o responsável insere a capacidade máxima do centro de distribuição para o dia em questão. Caso haja necessidade de eliminar linhas, o próprio sistema inicia o corte pelos artigos de classificação C. Tal como já foi referido, o corte das linhas não acontece em artigos que não apresentem *stock* (*stock*=0), em artigos com *stock* reservado em loja e quando o artigo pertence à classe A. É de realçar que as ordens de reposição são cortadas de forma igual, ou seja, se a necessidade total de cortar linhas for de 10%, em cada loja são eliminados 10% dos pedidos de reabastecimento, salvo algumas exceções. A causa destas exceções é justificada pela variação do total de linhas geradas ao longo do dia, ou seja, como o processo de separação dos artigos acontece loja a loja e as linhas de cada loja são geradas apenas quando o *picker* inicia o reabastecimento dessa mesma loja, é natural que haja novas vendas ou pedidos de reserva de artigos que não existiam inicialmente, e que resultam em alterações nas variáveis de cálculo do pedido de reabastecimento e consequentemente na criação de novas linhas. Assim, podem existir diferenças no corte das linhas das diferentes lojas.

Conforme indicado, cada *picker* fica responsável pelas linhas de determinada loja, sendo que terminada a separação dessa loja, procede à separação da seguinte. Este processo inicia-se geralmente pelas lojas com maior número de linhas, sendo a ordem de separação definida pelo responsável. Nesse sentido, embora os operadores sigam sempre a mesma rota, há a possibilidade de iniciar a recolha em diferentes corredores de forma a evitar congestionamentos. Como se verifica na Figura 11, os *pickers* podem iniciar a recolha dos artigos no local de início sugerido, mas também entre corredores, sendo que o sistema indica as posições de recolha de artigos mais próximas pela ordem da rota, independentemente do sítio onde esta foi iniciada. Assim, se a recolha tiver início, por exemplo no ponto 1, o *picker* efetua a seleção dos artigos necessários nos

corredores seguintes, sendo direcionado posteriormente para os corredores superiores caso tenha linhas situadas nessas posições.

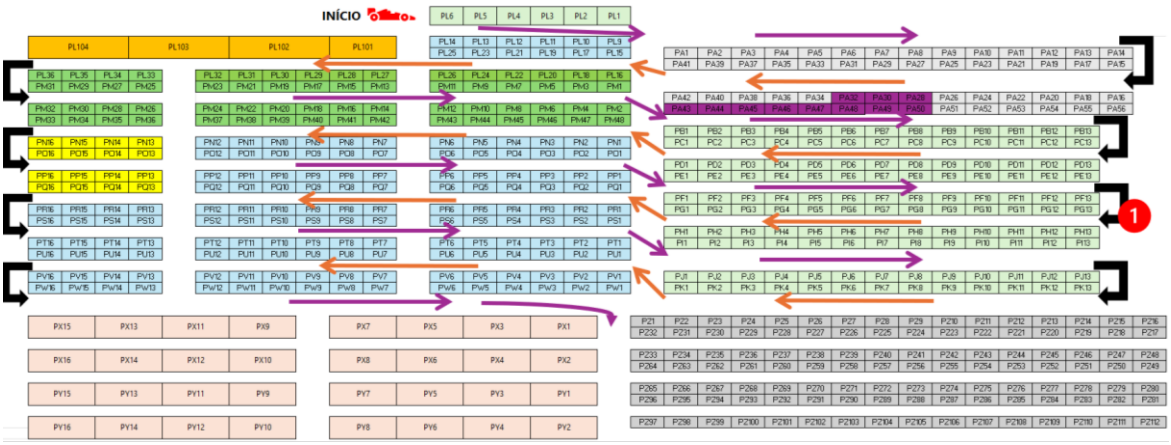


Figura 11 - Planta do centro de distribuição

Posteriormente, o *picker* coloca as caixas com os produtos recolhidos nas paletes disponibilizadas, para de seguida o operador de expedição filmar as paletes e colocá-las no camião. A quantidade de camiões utilizados varia consoante os artigos separados, sendo que um camião carrega no máximo 66 paletes. O período de carregamento do camião é entre as 06h30 e as 15h. De seguida, o camião segue para o seu armazém para proceder à separação das paletes consoante as suas rotas, chegando às lojas no dia seguinte, no caso destas se situarem em território nacional.

Posto isto, observando a situação do dia 03 de junho de 2024, segunda-feira, o sistema gerou 43200 linhas. No entanto, considerando que neste dia havia disponíveis 18 operadores, o responsável definiu uma capacidade máxima de separação de 20500 linhas, o que gerou um corte muito agressivo. Contudo, foram efetivamente separadas 21313 linhas, representadas na Figura 12, juntamente com as linhas originais para a reposição das 105 lojas geradas no pedido de reabastecimento.

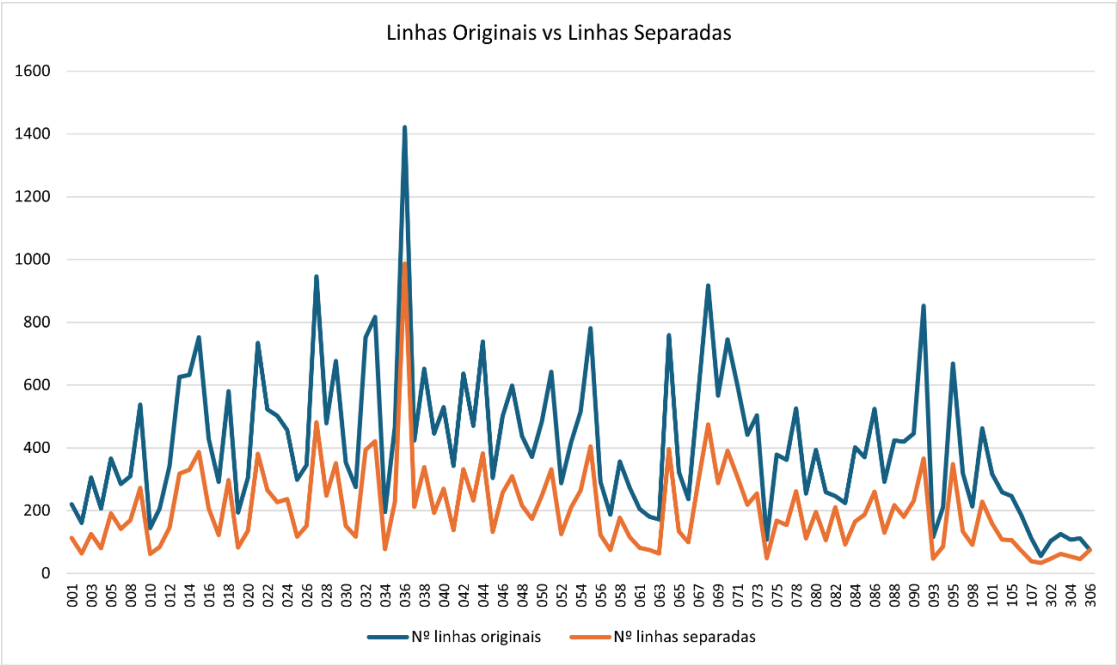


Figura 12 - Quantidade de linhas originais e separadas para o pedido de reabastecimento de 03 de junho

As linhas cortadas correspondem a 50,7% do total de linhas geradas, sendo que em nenhuma se verificou o *stock* de zero unidades ou *stock* reservado. Em relação à classe ABC, como se verifica na Figura 13, apenas foram cortadas linhas das classes CC e BC (B em valor e C em quantidade).

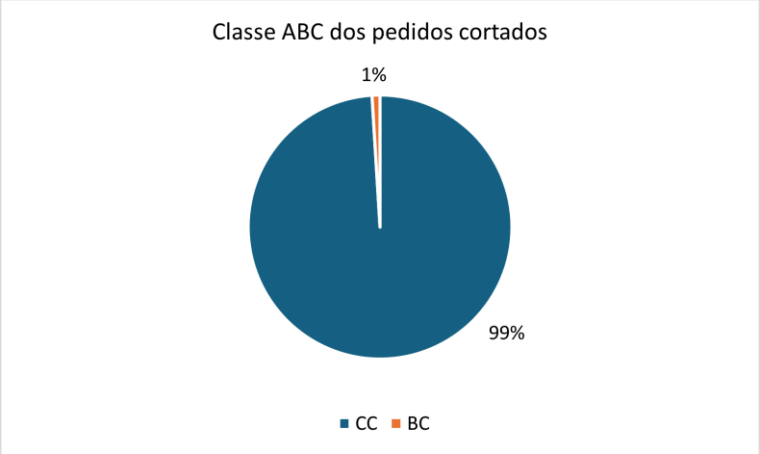


Figura 13 - Classe ABC dos pedidos cortados

Na Figura 14 está representada a percentagem das linhas cortadas em cada loja, destacando-se algumas diferenças decorrentes da situação já referida anteriormente – a variação das linhas geradas ao longo do dia – com exceção da loja 036. A maior diferença no corte das linhas na loja 036 deve-se ao facto de terem sido feitos 2 pedidos de reabastecimento. No primeiro pedido de reposição, devido ao corte muito agressivo, foram separadas 470 linhas. No final do dia, como ainda havia capacidade e por ser a loja com maior valor de faturação, houve um segundo pedido de reabastecimento, que resultou na separação de mais 517 linhas, contabilizando um total de 987 linhas separadas e um corte de 435 linhas ao fim da segunda reposição.


Figura 14 - % de linhas cortadas por loja

Assim, foram satisfeitos os pedidos de reposição de 105 lojas, através do carregamento de 2 camiões com 30 paletes cada um, sendo que o pedido deste dia correspondeu a 23 rotas da empresa transportadora.

### 3.1.3. Abastecimento das posições de *picking*

A operação de abastecimento das posições de *picking* é um trabalho contínuo, que envolve a deslocação do *stock* das estanterias de reserva para as respetivas posições na área de *picking*.

Inicialmente, um operador procede à separação dos artigos na zona de reserva, colocando-os numa zona neutra, onde os restantes operadores, após concluírem as suas tarefas, procedem à colocação dos produtos nas localizações de *picking* designadas pelo sistema. Caso o operador considere que a posição sugerida não é a mais adequada do ponto de vista ergonómico, o próprio pode analisar visualmente outra posição livre que seja mais favorável, colocar os artigos e inserir no sistema a nova posição.

Esta operação ocorre geralmente no fim do processo de separação dos artigos para as lojas e para o *online*, salvo algumas exceções em que não haja produto no *picking*. Considerando que o objetivo desta operação é deixar as posições de *picking* totalmente abastecidas para fazer face às necessidades das lojas do dia seguinte, o sistema considera os artigos com necessidades de reposição, aqueles, que após o processo de reabastecimento das lojas e do *online*, apresentam um *stock* menor que as necessidades das lojas do dia seguinte. Nesse sentido, após perceber quais os artigos com necessidades de reposição, as quantidades a repor são obtidas através da subtração das necessidades das lojas do dia seguinte pela quantidade existente do artigo na posição. No entanto, a alocação dos artigos às posições, é efetuada até ao limite da capacidade máxima da posição.

É de salientar que não existe nenhum tipo de ponderação considerado pela empresa na alocação dos artigos com necessidades a repor, ou seja, o sistema indica os artigos com necessidades de reposição pela ordem das posições da rota representada na Figura 11, não considerando, por exemplo, a margem bruta do artigo ou a sua classificação ABC como um sinal prioritário.

Relativamente às posições, estas podem ser caracterizadas como super pequenas, pequenas, médias, grandes e paletes. Os artigos dispostos nas paletes geralmente são os artigos de maior volume e de maior saída. É de salientar que todas as posições apresentam um código com o mesmo formato, onde cada caracter representa uma informação sobre a localização. Assim, o código PL19B1 indica que se trata de um código de posição (P), localizado no corredor L, na prateleira 19, correspondente à altura B e com profundidade de 1. A altura pode variar entre as letras A e M, sendo que a letra A reflete o nível mais alto da prateleira e o M o nível mais baixo.

Na Figura 15 estão representadas as necessidades totais das lojas do dia seguinte e as quantidades efetivamente repostas nas posições de *picking* ao longo da semana de 08 a 12 de julho de 2024. As quantidades repostas, assinaladas a vermelho, com exceção dos dias 10 e 12 estão sempre acima das quantidades com necessidades das lojas do dia seguinte.

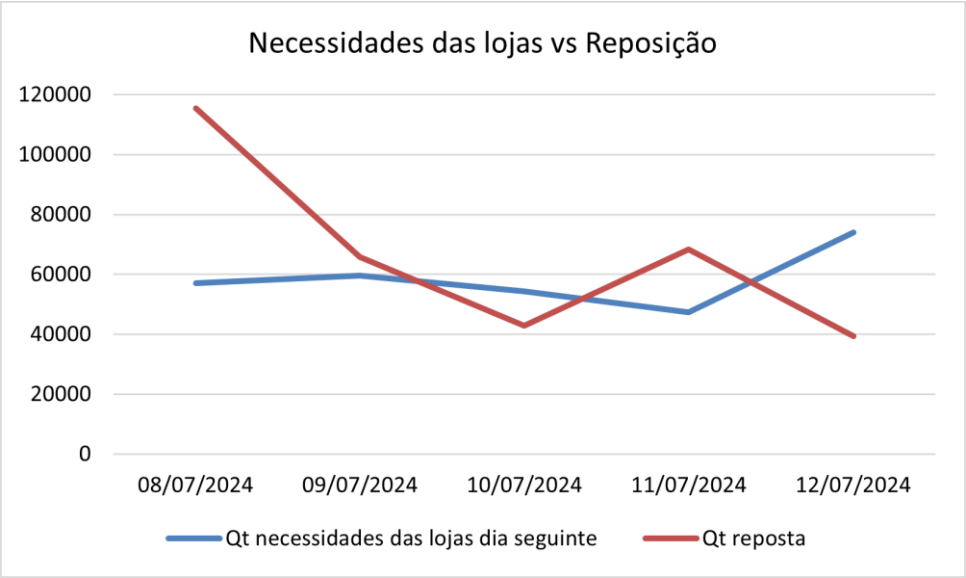


Figura 15 - Necessidades das lojas e reposição totais ao longo de 5 dias

Os dados extraídos da figura acima são apresentados na Tabela 5, juntamente com o número de artigos relativos às necessidades das lojas do dia seguinte e à sua reposição.

| Data | Necessidades das lojas | | Reposição | |
|---|---|---|---|---|
| | Nº artigos | Qt a separar | Nº artigos | Qt reposta |
| 08/07/2024 | 4429 | 57116 | 1158 | 115427 |
| 09/07/2024 | 4790 | 59678 | 824 | 65912 |
| 10/07/2024 | 4621 | 54344 | 706 | 42810 |
| 11/07/2024 | 4535 | 47304 | 870 | 68349 |
| 12/07/2024 | 4062 | 74035 | 652 | 39460 |

Tabela 5 - Distribuição das necessidades das lojas e reposição totais ao longo de 5 dias

É de salientar que o número de artigos com necessidades a separar para as lojas no dia seguinte, não é indicativo que há essa necessidade de reposição no próprio dia, ou seja, apenas há necessidade de reposição nos artigos em que o seu *stock* em posição é inferior às necessidades das lojas do dia seguinte. Tal como foi referido acima, com exceção dos dias 10 e 12, as quantidades repostas superam as quantidades com necessidades das lojas do dia seguinte, o que é natural considerando a estratégia que a empresa adota de encher as posições até ao limite máximo da sua capacidade.

A taxa de ocupação das posições de *picking* repostas no fim e durante o processo de abastecimento das mesmas ao longo de uma semana está representada na Figura 16. Considerando a ocupação das posições no fim do abastecimento durante os 5 dias, verifica-se pelo primeiro e segundo quartil, que 25% das posições apresentam um nível de utilização menor que 55% e metade das posições de *picking* uma taxa de ocupação inferior a 83%, respetivamente. A menor percentagem de utilização situa-se nos 0%, enquanto a maior estende-se aos 165%, contando ainda com a presença de *outliers* acima deste valor. A presença destes *outliers* e de taxas de utilização superiores a 100% pode ser justificada pela contabilização errada da quantidade máxima a armazenar do artigo na posição.

No entanto, ao analisar a taxa de ocupação das posições repostas durante o processo de abastecimento ao longo da semana, a situação altera-se, pois para a mesma percentagem de posições há uma menor taxa de ocupação. Ou seja, 25% das posições de *picking* repostas revelam uma utilização menor que 52% e metade das posições uma taxa inferior a 80%.

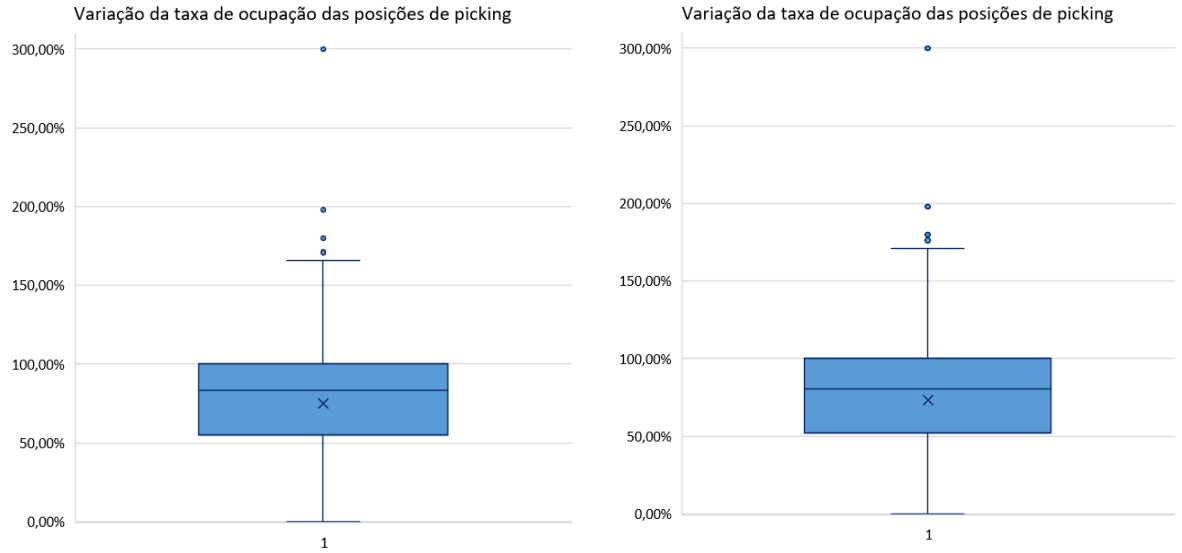


Figura 16 - Variação da taxa de ocupação das posições de *picking* repostas no fim (à esquerda) e durante (à direita) o processo de abastecimento das mesmas ao longo de 5 dias

Esta diferença é explicada pelo facto de haver posições que são ocupadas temporariamente durante o processo de abastecimento, como acontece nos casos seguintes, representados pela Figura 17 e Figura 18. É de salientar que a linha que reflete o *stock* da posição indica a quantidade que existia em determinada posição no momento antes do seu abastecimento e a capacidade máxima refere-se à quantidade máxima do artigo que é possível alocar à posição.

Através da Figura 17, observa-se que o artigo X1394, com classificação BB foi reposto nos dias 08, 10 e 11 de julho em posições grandes, sendo que no dia 8 foi reposto 3 vezes em 3 posições diferentes, com taxas de ocupação de 50%, 12% e 50%, respetivamente. A segunda posição (PH1A3) é uma posição temporária, pois no mesmo dia, às 17h03, já estava ocupada com outro artigo a uma taxa de utilização de cerca de 73%, o que indica que a primeira ocupação desta posição e consequente deslocação a este corredor podia ter sido evitada, pois a quantidade de artigos reposta nesta posição podia ter sido reposta na posição PG1A2, visto que esta ainda tinha metade da sua capacidade livre.

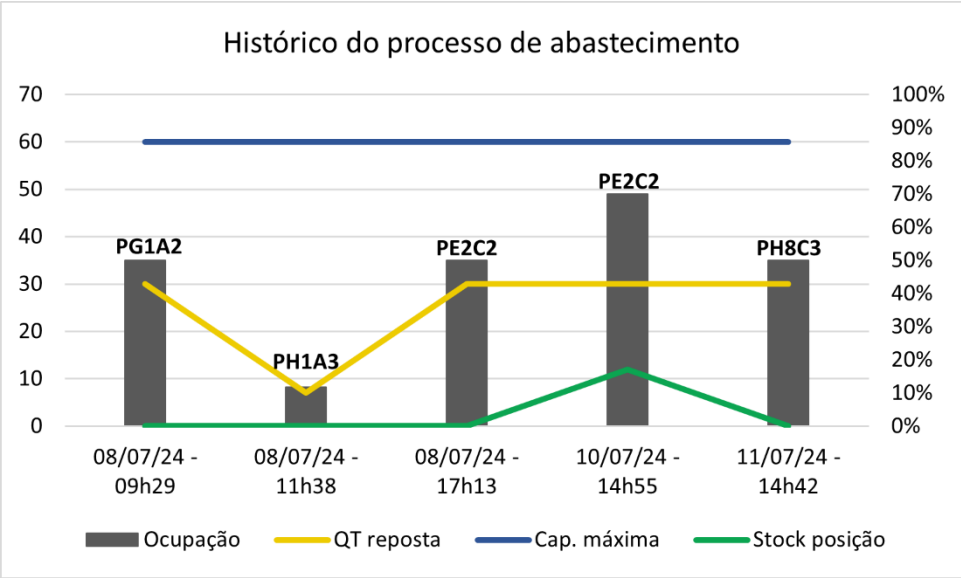


Figura 17 - Histórico do processo de abastecimento do artigo X1394

O artigo X3025, classificado como artigo B em quantidade e A em valor foi reposto apenas em posições pequenas nos dias 11 e 12 de julho. A primeira reposição é efetuada na posição PL101A1, que registou uma utilização de 6%. No mesmo dia, a posição PL101E3 foi abastecida duas vezes, tendo a sua ocupação fixada nos 80%. No dia seguinte voltou-se a repor este artigo na mesma posição, alcançando uma taxa de ocupação de 86%. Mais uma vez, há uma posição que podia não ter sido utilizada (PL101A1) e duas das 3 reposições realizadas no dia 11 podiam ter sido evitadas.

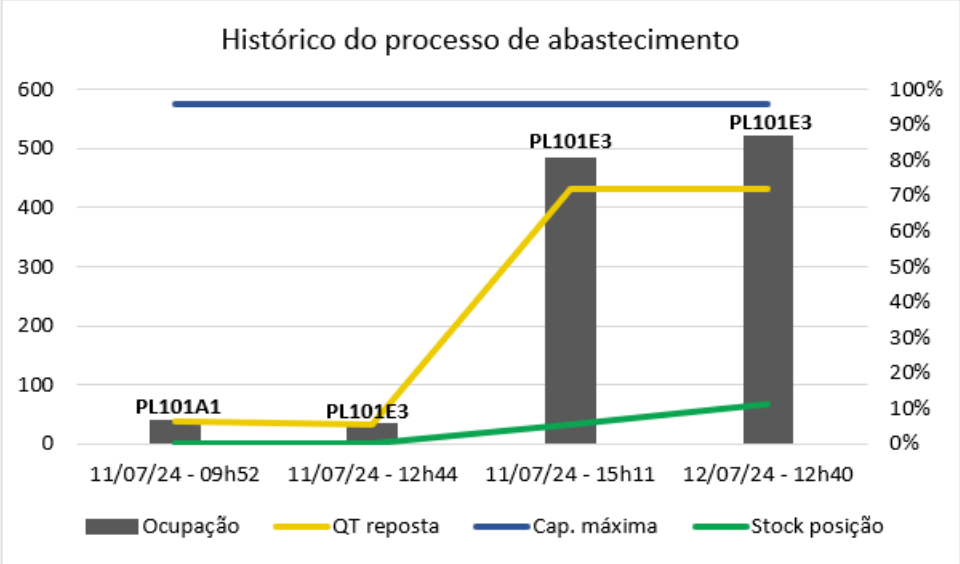


Figura 18 - Histórico do processo de abastecimento do artigo X3025

## 3.1.4. Limitações

Após o estudo e análise das 3 fases de operações do centro de distribuição foi possível tomar uma decisão sobre qual seria o ponto em que este projeto se iria debruçar. Assim, a temática a estudar é o abastecimento das posições de *picking*.

Como se verifica na descrição desta operação, a possibilidade dos operadores poderem escolher outras posições que não a proposta pelo sistema vai ocupar mais tempo no processo de reposição. Há que salientar também a inexistência de uma alocação ponderada, ou seja, na eventualidade de não ser possível alocar todos os artigos com necessidades de reposição, há a possibilidade de artigos importantes não serem armazenados e pelo contrário, artigos considerados menos importantes serem os primeiros a serem alocados, apenas porque a lista de artigos com necessidades de reposição segue a rota evidenciada na Figura 11.

Conforme analisado acima, os dados relativos ao abastecimento das posições de *picking* revelam uma ocupação desnecessária, seja temporariamente ou não de várias posições, isto é, há várias posições que no mesmo dia são ocupadas de modo provisório e parcial por diferentes artigos. Além disso, há também várias posições que armazenam o mesmo artigo com uma baixa taxa de ocupação, o que indica que o critério de alocação das posições até à sua capacidade máxima pode não estar a ser totalmente seguido.

Posto isto, o problema reside no facto do processo de abastecimento ao *picking* não estar a ser bem definido, pois apesar do processo descrito, há situações que não estão em conformidade com o mesmo, gerando ineficiências quer em termos de quantidade, localização e frequência. As principais consequências associadas a esta operação estão bastante interligadas e são:

- Rutura no *picking*, isto é, falta de artigos nas posições para posteriormente serem separados para as lojas, que pode acontecer por diversos motivos, o facto de se armazenar uma quantidade superior à necessária de determinado artigo, limita o tempo livre para abastecer os restantes artigos. Considerando a limitação do tempo e o facto de não existir nenhum tipo de prioridade na alocação dos artigos, pode gerar rutura em artigos importantes;
- Ocupação inadequada do espaço, que por ser limitado, pede uma gestão eficaz do mesmo. Assim, ao armazenar mais quantidade de determinado artigo do que é necessário, há o risco de certos artigos apresentarem um excesso de *stock* e por outro lado, o facto de se armazenar posições de modo parcial, com baixa taxa de ocupação leva a que não haja posições disponíveis para outros artigos, o que se traduz na rutura no *picking*;
- Reposição frequente do mesmo artigo, ou seja, há reposições do mesmo artigo que podiam ser evitadas, pois muitas vezes não há a necessidade de armazenar 3 vezes o mesmo artigo em posições diferentes ou até na mesma posição, obrigando o operador a deslocar-se 3 vezes para a sua reposição. Nesse sentido, há reposições em que as posições são ocupadas de forma parcial e algumas vezes, de forma temporária. Esta situação leva a que o operador perca tempo a reabastecer o mesmo artigo várias vezes, quando podia armazenar outros artigos, e consequentemente evitar a ocupação de posições em excesso por parte do mesmo artigo e evitar a rutura de outros artigos.

## 3.2. Seleção dos dados

Para conseguir responder ao objetivo pretendido com este projeto, de determinar a posição ou posições onde se vai repor determinada quantidade de cada artigo foi selecionada uma amostra de dados que contempla todas as lojas para um horizonte temporal de uma semana, de 09 a 13 de setembro de 2024. Nesse sentido, foram selecionados dados relativos às necessidades das lojas, ao *stock* atual por posição e por artigo, à dimensão do tipo de posições e posições existentes, à dimensão e margem bruta dos artigos e a uma lista de artigos importantes, composta por 80 artigos

considerados essenciais para um cliente profissional trabalhar e caso estejam em rutura, este opta por adquirir o artigo noutro estabelecimento.

## 3.3. Análise e preparação dos dados

Após a seleção e recolha dos dados, a análise e preparação destes são etapas cruciais para atingir uma solução adequada para dar resposta à questão de investigação, pois garantem que as decisões são tomadas com base em informação relevante.

### 3.3.1. Análise dos dados

O centro de distribuição conta com 13875 posições de *picking* disponíveis, que se dividem em 7 tipos com diferentes dimensões refletidas na Tabela 6.

| Tipo | Nº posições | Profundidade (cm) | Largura (cm) | Altura (cm) |
|---|---|---|---|---|
| A1 | 3528 | 40 | 10 | 15 |
| A2 | 6832 | 40 | 30 | 22 |
| A3 | 2049 | 60 | 40 | 32 |
| A4 | 140 | 320 | 30 | 22 |
| A6 | 739 | 40 | 20 | 15 |
| A9 | 330 | 35 | 20 | 30 |
| PALETE | 257 | 120 | 80 | 150 |

Tabela 6 - Caracterização das posições por tipo

A tabela seguinte é obtida através dos elementos referentes às saídas dos produtos para as lojas e *online* e aos dados de *stock* de início do dia. Assim, antes dos operadores procederem ao reabastecimento das posições de *picking*, a quantidade de posições com *stock* e disponíveis para este efeito estão quantificadas na Tabela 7. A percentagem de posições livres varia em média entre os 31% e 34%.

| Data | Posições com *stock* | Posições livres |
|---|---|---|
| 09/09/2024 | 9126 | 4749 |
| 10/09/2024 | 9394 | 4481 |
| 11/09/2024 | 9517 | 4358 |
| 12/09/2024 | 9540 | 4335 |
| 13/09/2024 | 9620 | 4255 |

Tabela 7 - Quantidade de posições com *stock* e livres para o reabastecimento das posições de *picking*

Como indicado na Tabela 7, no dia 09, havia 9126 posições com *stock* alocado e consequentemente 4749 posições livres passíveis de armazenar necessidades a repor, distribuídas pelos 7 tipos de posições como se verifica pela Figura 19.

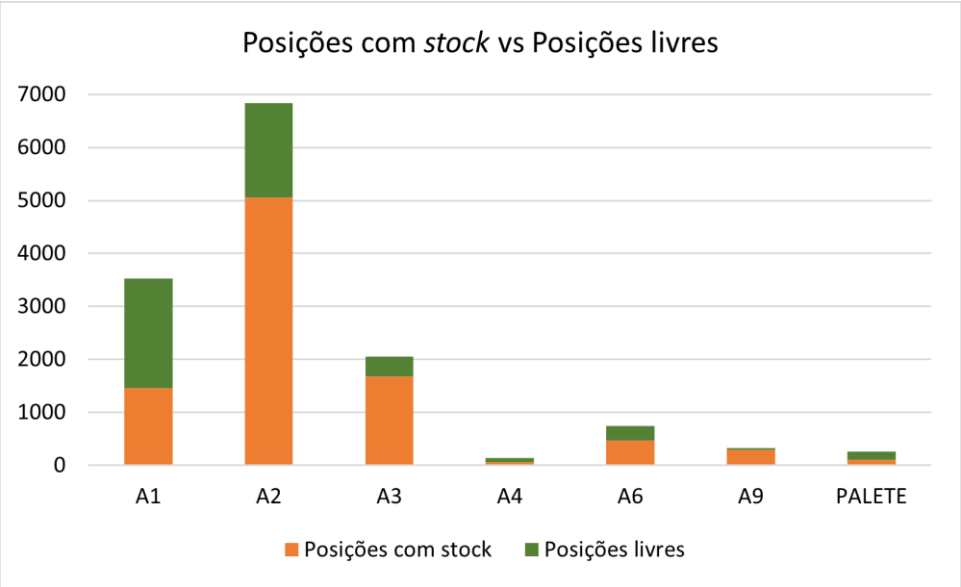


Figura 19 - Quantidade de posições com *stock* e livres a 09/09/2024

Considerando o processo de abastecimento da área de *picking*, foram repostos 885 artigos com um total de 94596 unidades. O reabastecimento destas posições foi realizado em 1198 reposições, sendo que 874 foram efetuadas em posições livres e as restantes 324 em posições com *stock*. Contudo as 324 reposições abrangeram 293 posições, o que indica que houve posições a ser "visitadas" mais do que uma vez para alocar o mesmo artigo ou ainda artigos diferentes, transformando essa posição numa posição temporária.

No dia 10, havia necessidades das lojas de 4636 artigos com um total de 65823 unidades, mas foram efetivamente enviados 4576 artigos totalizando as 54181 unidades. A loja *online* contabilizou a separação de 1961 unidades, distribuídas por 913 artigos.

Após o processo de separação dos produtos para as lojas e *online*, nesse dia, mantiveram-se com *stock* 9394 posições, enquanto as posições livres e aptas para armazenar necessidades a repor passaram para 4481. A Figura 20 apresenta as posições dispostas por tipo de posição.

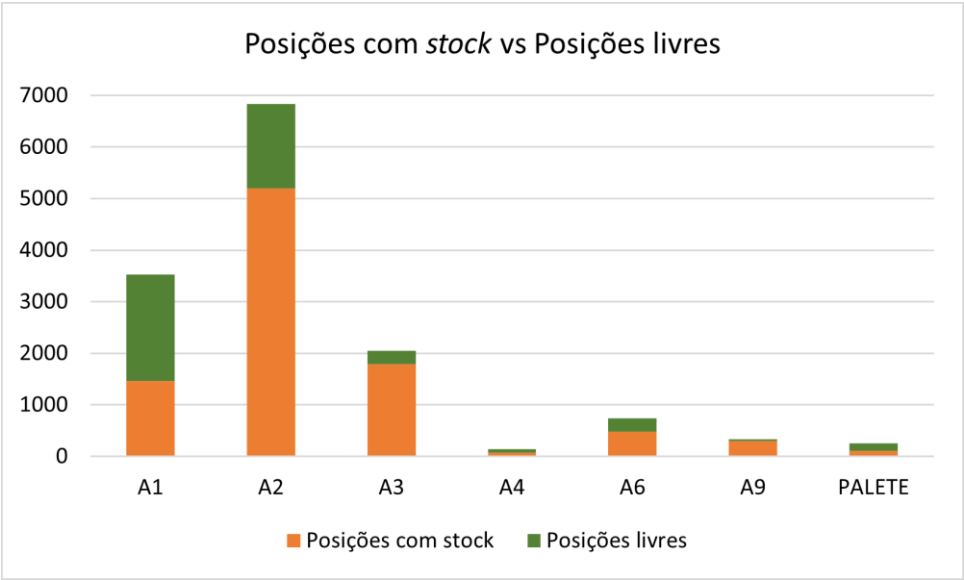


Figura 20 - Quantidade de posições com *stock* e livres a 10/09/2024

O armazenamento da zona de *picking* contou com o abastecimento de 941 artigos, através de 1149 reposições e um volume de 70668 unidades. Do total de reposições, 685 foram realizadas em posições novas, enquanto as restantes 464 reposições foram alocadas em posições com *stock*. Semelhante ao dia anterior, houve reposições que podiam ter sido evitadas, pois os 431 artigos que foram repostos 464 vezes em posições com *stock*, foram armazenados em 432 posições diferentes, existindo casos em que o mesmo artigo foi reposto na mesma posição mais do que uma vez, quando a reposição podia ter sido feita uma única vez.

No dia 11 existiam 4590 artigos com necessidades das lojas, que correspondiam a um total de 51786 unidades. O centro de distribuição enviou menos 20 artigos dos que apresentavam necessidades, separando 47217 unidades. Os 1049 artigos retirados das localizações de *picking* referentes às vendas do *online* contabilizaram 3457 unidades.

A Figura 21 exibe as 9517 posições ocupadas com *stock* e as respetivas 4358 posições livres distribuídas por cada tipo de posição existente, após o processo de *picking*, no terceiro dia do período em análise.

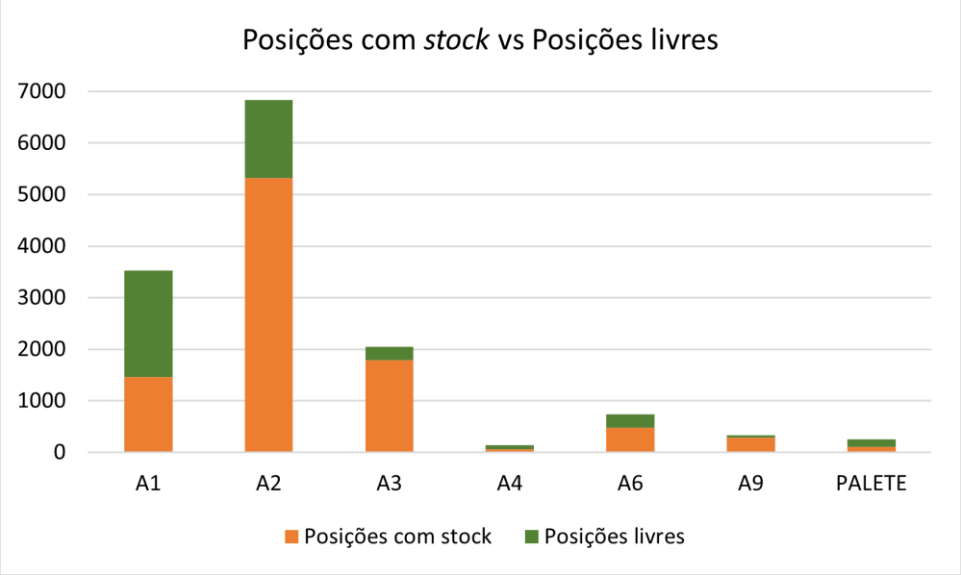


Figura 21 - Quantidade de posições com *stock* e livres a 11/09/2024

No que diz respeito ao processo de abastecimento das posições de *picking*, os 608 artigos e as 47056 unidades foram alocados através de 745 reposições, sendo que 234 foram realizadas em posições com *stock* e as restantes 511 em posições livres.

As necessidades das lojas do dia 12 atingiram as 48025 unidades distribuídas por 4266 artigos, contudo foram separados 4219 artigos e um total de 44187 unidades para as lojas. Já o *online* atingiu as 3676 unidades separadas através de 1003 artigos.

Seguida a operação de separação dos produtos das lojas e do *online*, o dia 12, permanecia com um total de 9540 posições com *stock* alocado e 4335 posições disponíveis por ocupar. A Figura 22 expõe-nas de forma repartida pelos 7 tipos de posições de *picking*.

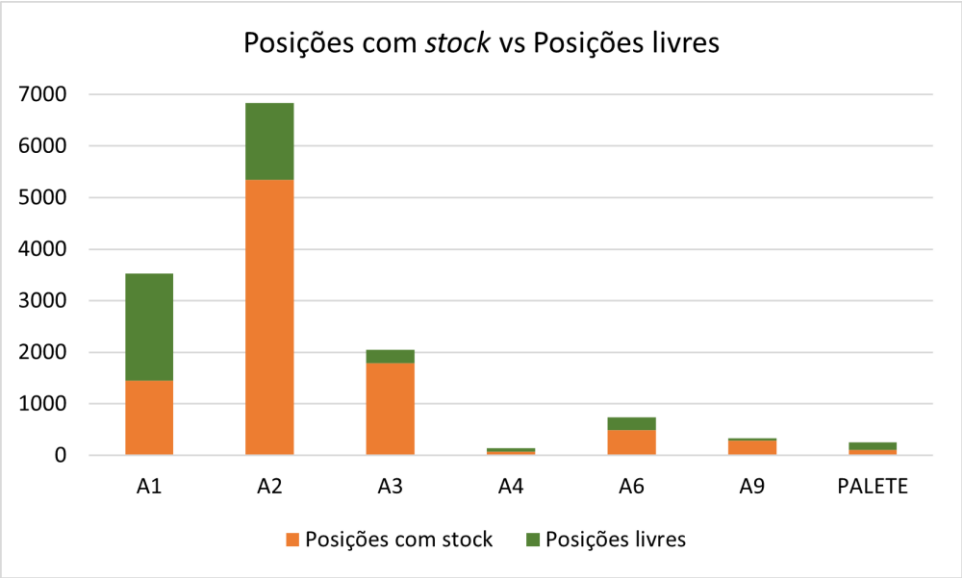


Figura 22 - Quantidade de posições com *stock* e livres a 12/09/2024

Este dia contou com o armazenamento de 948 artigos ao longo de 1259 reposições com um volume de 87877 unidades reabastecidas. Ao contrário do que acontece nos dias anteriores em que as reposições em posições livres são superiores às reposições em posições com *stock*, aqui foram efetuadas 712 reposições em posições com *stock*, enquanto as restantes 547 foram realizadas em posições livres. Uma vez mais, houve reposições que podiam ter sido evitadas, pois os 592 artigos repostos em posições com *stock*, foram armazenados em 596 posições, havendo casos em que o mesmo artigo foi reposto mais do que uma vez na mesma posição.

No último dia em análise, o dia 13, foram contabilizados 3970 artigos com necessidades das lojas, correspondendo a um total de 43718 unidades, mas apenas 39503 unidades foram efetivamente enviadas para as lojas, correspondendo a 3953 artigos diferentes. Além disso, os pedidos do *online* totalizaram as 2103 unidades, distribuídas por 688 artigos.

Antes da empresa proceder ao reabastecimento das posições de *picking*, verifica-se pela Figura 23, a disposição por tipo de posição das 9620 posições ocupadas com *stock* e das 4255 posições livres.

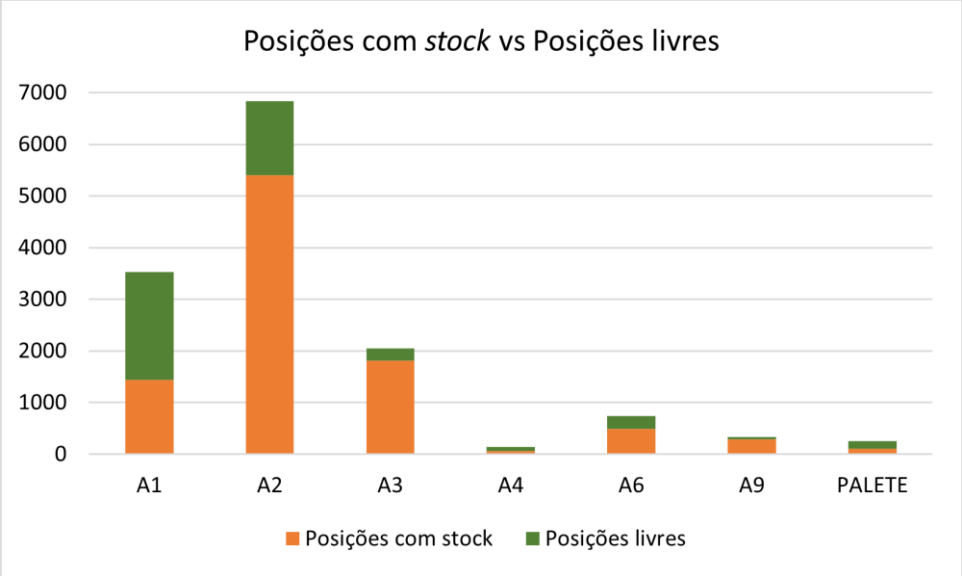


Figura 23 - Quantidade de posições com *stock* e livres a 13/09/2024

Posto isto, foram repostos 468 artigos, sendo que a operação de abastecimento às posições de *picking* foi efetuada através de 611 reposições, com um volume de 44727 unidades armazenadas. Do total das reposições foram realizadas 408 em posições livres e 203 em posições com *stock*.

É de salientar que no contexto do reabastecimento das posições de *picking*, as necessidades das lojas são consideradas no cálculo das necessidades a repor do dia anterior, ou seja, as necessidades das lojas do dia 10, por exemplo, são tidas em conta no cálculo das necessidades de reposição do dia 09. Isto acontece porque o objetivo da empresa a seguir à separação dos produtos das lojas e do *online*, é armazenar as posições de modo a fazer face às necessidades das lojas do dia seguinte.

Após esta análise, no que diz respeito à operação de reposição durante o período em análise, é possível concluir, através da Figura 24, que a empresa tem uma maior tendência para repor artigos em posições livres do que em posições com *stock* já alocado, com exceção do dia 12/09. É também evidente um declínio no número de reposições à quarta e sexta-feira, mas considerando o histórico de apenas uma semana, esta quebra não pode ser interpretada como um padrão consistente.

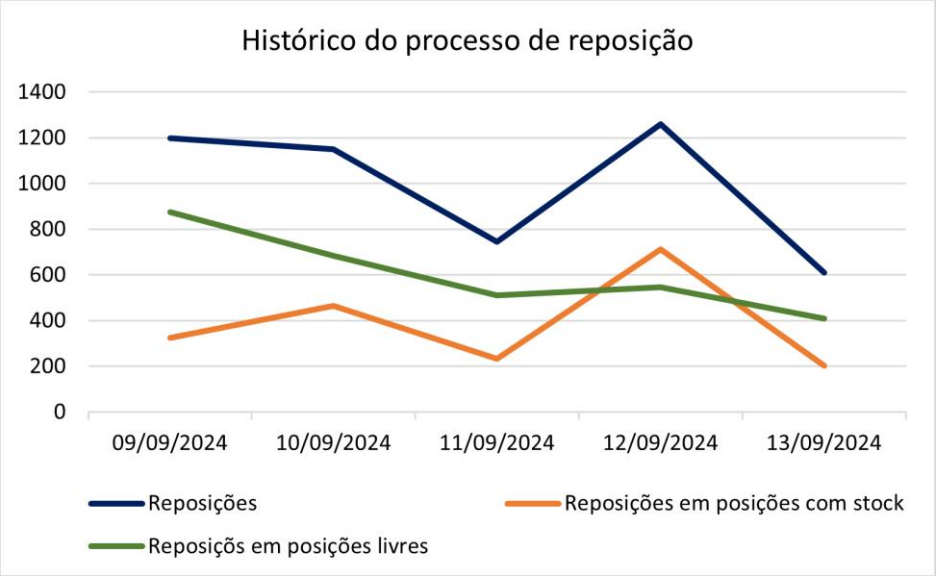


Figura 24 - Histórico do processo de reposição das posições de *picking* durante 5 dias

## 3.3.2. Preparação dos dados

Após a análise dos dados, é necessário estruturar e fazer a preparação dos mesmos para que seja possível obter uma solução para o problema já definido.

Assim, antes de determinar uma ou mais posições para alocar as necessidades a repor de cada artigo é fundamental perceber que artigos teriam de ser repostos.

Para tal, atendendo aos dados facultados acerca das necessidades das lojas, do *stock* de início do dia e das saídas de cada dia foi possível calcular as necessidades a repor de cada artigo. As saídas de cada dia dizem respeito à quantidade de artigos separados das posições de *picking* para as lojas e para as encomendas do *online*. Nesse sentido, considerando que a operação de separação dos produtos para as lojas e *online* é a primeira tarefa dos operadores e pressupondo que ocorre geralmente durante a manhã, retiraram-se as saídas de cada artigo ao *stock* de início do dia, de modo a obter um *stock* atualizado. Tal como mencionado anteriormente, sendo o objetivo do

reabastecimento das posições de *picking* para cada dia, fazer face às necessidades das lojas do dia seguinte, e tendo estas como base, o *stock* atualizado serviu para determinar que artigos apresentavam necessidades de reposição e em que quantidade. O pseudocódigo que resume esta primeira fase de tratamento dos dados encontra-se na Figura 25. O código desenvolvido para o cálculo das necessidades está disponível no APÊNDICE A

```
1. variables: necessidades_lojas_dia_seguinte, saídas, stock_início_dia
2. stock_atualizado = stock_início_dia – saídas
3. If necessidades_lojas_dia_seguinte ≥ stock_atualizado then
       necessidades_repor = necessidades_lojas_dia_seguinte – stock_atualizado
   else
       necessidades_repor = 0
```

Figura 25 - Pseudocódigo relativo ao cálculo das necessidades a repor

Ainda referente ao cálculo das necessidades a repor, é de salientar que apenas para o primeiro dia, 09/09/2024, foram utilizados os dados do *stock* inicial fornecidos pela empresa. Nos restantes dias o *stock* de início do dia foi calculado com base nas alocações geradas pela solução proposta – apresentada no subcapítulo seguinte, assim como a explicação do cálculo do *stock* do dia seguinte.

Após obter os artigos com necessidades de reposição e considerando a prática da empresa, de armazenar os mesmos nas posições até à sua capacidade máxima, tornou-se fundamental quantificar para cada artigo com necessidades a repor, a quantidade que era possível de alocar em cada tipo de posição disponível para esse artigo. Através dos dados facultados acerca da dimensão dos artigos e do tipo de posições, assim como da quantidade máxima que é possível armazenar de cada artigo num tipo de posição em específico, foi possível tomar o segundo passo do tratamento dos dados. Este, consistiu na criação de pares entre os artigos com necessidades de reposição e os tipos de posições, sendo que o par apenas era válido caso, independentemente da disposição do artigo na posição, as dimensões deste fossem inferiores e possíveis de armazenar nas dimensões associadas a cada tipo de posição. De seguida, foi associado a cada par uma capacidade máxima ajustada, que indica o limite máximo da quantidade do artigo que é possível alocar no tipo de posição. A Figura 26 apresenta o pseudocódigo referente ao código desenvolvido e disponível no APÊNDICE B

```
1. variables: altura_tipo, largura_tipo, profundidade_tipo, altura_tipo_original, largura_tipo_ original,
   profundidade_tipo_original, altura_artigo, largura_artigo, profundidade_artigo, tipo, tipo_original,
   cap_máxima_original
2. volume_tipo_posição = altura_tipo * largura_tipo * profundidade_tipo
3. volume_tipo_original = altura_tipo_original * largura_tipo_original * profundidade_tipo_ original
4. proporção_volumes = volume_tipo_posicao / volume_tipo_original
   If altura_tipo, largura_tipo, profundidade_tipo ≥ altura_artigo, largura_artigo,
   profundidade_artigo (em todas as orientações) then
       If tipo = tipo_original then
           capacidade_ajustada = cap_máxima_original
       else
           capacidade_ajustada = cap_máxima_original * proporção_volumes
   else
       descartar tipo
```

Figura 26 - Pseudocódigo relativo à criação de pares artigo e tipo e capacidade ajustada associada

## 3.4. Proposta de solução

A etapa de preparação dos dados foi determinante para o desenvolvimento da proposta de solução, pois forneceu informação acerca do *stock* atualizado, das necessidades de reposição de cada artigo e estabeleceu uma lista de artigos passíveis de armazenar em cada tipo de posição, assegurando que as dimensões destes se enquadravam nas dimensões dos tipos de posições. Além disso, associou a cada par de artigo e tipo um limite máximo de capacidade de armazenamento. Nesta fase, também foram considerados dados cruciais como a margem bruta dos artigos, a lista dos artigos considerados essenciais e prioritários, a dimensão dos tipos de posições e as posições existentes.

Assim, foi necessário dividir a proposta de solução em duas fases. Inicialmente, através das necessidades de reposição de cada artigo, foi possível criar uma lista de artigos, ordenada pela sua prioridade em função do impacto financeiro estimado pela margem bruta de cada artigo e pela sua importância. A importância dos artigos foi definida através de um parâmetro de importância ajustável, onde caso o artigo pertencesse à lista dos 80 artigos importantes, era abrangido por esse indicador, caso contrário seria atribuído o valor zero. Desta forma, através do cálculo do fator prioridade de cada artigo foi possível ordenar os artigos de forma descendente pela sua prioridade, isto é, os artigos foram listados com as necessidades a repor, pelo fator de maior prioridade para menor. Dado que não existe um limite predefinido da capacidade de reabastecimento das posições de *picking* no centro de distribuição, esta lista de artigos e necessidades a repor, ordenada pela prioridade de cada um, foi criada para ser seguida até onde o responsável determinar, garantindo que os artigos considerados essenciais de ter em loja sejam repostos nas suas localizações em primeiro lugar, tendo também em atenção a sua margem bruta. O pseudocódigo da primeira fase desenvolvida encontra-se na Figura 27 e o código no APÊNDICE C

```
1. variables: artigos_necessidades_repor, lista_artigos_importantes, margem_bruta,
   necessidades_repor, valor_importancia
2. valor_importancia = majorante do produto da margem_bruta pelas necessidades_repor
3. If artigos_necessidades_repor ⊂ lista_artigos_importantes then:
       importancia = valor_importancia
   else
       importancia = 0
4. prioridade = (margem_bruta * necessidades_repor) + importancia
```

Figura 27 - Pseudocódigo relativo à criação da lista de prioridades

É de salientar que o valor de importância foi definido pelo majorante do retorno da multiplicação da margem bruta unitária pelas necessidades a repor. No entanto, este critério pode ser alterado pelo responsável. Esta escolha foi considerada pois o objetivo da solução proposta era garantir que os artigos pertencentes à lista de artigos importantes fossem os primeiros a ser considerados na operação de reabastecimento das posições de *picking*.

Na Figura 28 segue um exemplo da primeira parte da solução proposta. É possível observar os 24 artigos com necessidades de reposição, listados da maior para menor prioridade, satisfazendo o critério de importância dos artigos considerados essenciais de ter em loja, sendo estes os primeiros da lista, para posteriormente serem os primeiros a alocar nas posições de *picking.* Além disso, considera também os artigos que multiplicado pelas necessidades a repor, apresentem maior margem bruta. A coluna à parte expõe a prioridade caso apenas fosse considerada a margem bruta.

Nesse caso, os 3 artigos destacados a verde seriam os primeiros da lista a ser alocados, priorizando artigos fora da lista de artigos essenciais. Por outro lado, os 3 artigos preenchidos a laranja são artigos considerados fundamentais e caso esta importância não fosse considerada, seriam os últimos a ser alocados.

| id_artigo | necessidades_repor | margem_bruta | importancia | prioridade |
|---|---|---|---|---|
| X0245 | 668 | 0,005179 | 4,25 | 7,71 |
| X3065 | 406 | 0,007866 | 4,25 | 7,45 |
| X3049 | 383 | 0,008098 | 4,25 | 7,36 |
| X3039 | 286 | 0,007996 | 4,25 | 6,54 |
| X9175 | 198 | 0,005957 | 4,25 | 5,43 |
| X0026 | 123 | 0,010845 | 4,25 | 5,59 |
| X9609 | 99 | 0,006735 | 4,25 | 4,92 |
| X7649 | 43 | 0,006359 | 4,25 | 4,53 |
| X2735 | 27 | 0,009538 | 4,25 | 4,51 |
| X1128 | 13 | 0,007834 | 4,25 | 4,36 |
| X6189 | 5 | 0,007861 | 4,25 | 4,29 |
| X4652 | 529 | 0,008043 | 0 | 4,25 |
| X0052 | 224 | 0,014399 | 0 | 3,23 |
| X5191 | 393 | 0,006500 | 0 | 2,55 |
| X4804 | 249 | 0,008484 | 0 | 2,11 |
| X0846 | 250 | 0,007951 | 0 | 1,99 |
| X4373 | 245 | 0,007547 | 0 | 1,85 |
| X3422 | 372 | 0,004963 | 0 | 1,85 |
| X4802 | 202 | 0,008454 | 0 | 1,71 |
| X2912 | 247 | 0,006899 | 0 | 1,70 |
| X2949 | 200 | 0,007725 | 0 | 1,55 |
| X0102 | 208 | 0,007356 | 0 | 1,53 |
| X4199 | 162 | 0,009175 | 0 | 1,49 |
| X3865 | 226 | 0,006513 | 0 | 1,47 |

| |
|---|
| 3,46 |
| 3,19 |
| 3,10 |
| 2,29 |
| 1,18 |
| 1,33 |
| 0,67 |
| 0,27 |
| 0,26 |
| 0,10 |
| 0,04 |
| 4,25 |
| 3,23 |
| 2,55 |
| 2,11 |
| 1,99 |
| 1,85 |
| 1,85 |
| 1,71 |
| 1,70 |
| 1,55 |
| 1,53 |
| 1,49 |
| 1,47 |

Figura 28 - Exemplo de lista de prioridades (1ª fase da solução proposta)

Na segunda fase de desenvolvimento da solução, o objetivo era alocar a cada artigo da lista de artigos ordenada pela sua prioridade, uma ou mais posições que comportassem as necessidades a repor do artigo. A solução encontrada segue uma hierarquia de critérios, determinando inicialmente a posição ou posições onde as necessidades a repor vão ser alocadas pela minimização da área ocupada, de seguida considera a minimização do número de posições e por consequência a minimização do número de reposições.

Em primeiro lugar, considerando o objetivo de minimizar a área das posições ocupadas, foi crucial calcular a área de cada tipo de posição, como se verifica na Tabela 8.

| Tipo | Profundidade (cm) | Largura (cm) | Área ($cm^2$) |
|---|---|---|---|
| A1 | 40 | 10 | 400 |
| A2 | 40 | 30 | 1200 |
| A3 | 60 | 40 | 2400 |
| A4 | 320 | 30 | 9600 |
| A6 | 40 | 20 | 800 |
| A9 | 35 | 20 | 700 |
| PALETE | 120 | 80 | 9600 |

Tabela 8 - Área dos tipos de posição

De seguida, foi necessário considerar os pares criados entre os 24 artigos expostos na Figura 28 e os tipos de posições adequados para os mesmos, retirados da preparação dos dados. O algoritmo ordena para cada artigo, o tipo de posição de modo ascendente, ou seja, da menor para a maior área e em caso de empate – que acontece nos tipos A4 e PALETE – a ordenação é efetuada de forma crescente com base na capacidade máxima ajustada, retirada ainda da mesma preparação dos dados, explícita na Figura 26. Este passo resultou em parte na Tabela 9. O resultado completo desta tabela encontra-se no APÊNDICE D

| id_artigo | tipo | capacidade_ajustada | Área (cm²) |
|---|---|---|---|
| X0245 | A1 | 25 | 400 |
| X0245 | A9 | 87 | 700 |
| X0245 | A6 | 50 | 800 |
| X0245 | A2 | 110 | 1200 |
| X0245 | A3 | 320 | 2400 |
| X0245 | A4 | 880 | 9600 |
| X0245 | PALETE | 6000 | 9600 |
| X3065 | A1 | 144 | 400 |
| X3065 | A9 | 504 | 700 |
| X3065 | A6 | 288 | 800 |
| X3065 | A2 | 633 | 1200 |
| X3065 | A3 | 1843 | 2400 |
| X3065 | A4 | 5068 | 9600 |
| X3065 | PALETE | 34560 | 9600 |
| X3049 | A1 | 16 | 400 |
| X3049 | A9 | 57 | 700 |
| X3049 | A6 | 32 | 800 |
| X3049 | A2 | 72 | 1200 |
| X3049 | A3 | 209 | 2400 |
| X3049 | A4 | 576 | 9600 |
| X3049 | PALETE | 3927 | 9600 |
| X3039 | A1 | 16 | 400 |
| X3039 | A9 | 57 | 700 |
| X3039 | A6 | 32 | 800 |
| X3039 | A2 | 72 | 1200 |
| X3039 | A3 | 209 | 2400 |
| X3039 | A4 | 1152 | 9600 |
| X3039 | PALETE | 3927 | 9600 |

Tabela 9 - Lista ordenada dos pares artigo e tipo

Posto isto, para minimizar a área das posições, a lógica a implementar inicialmente foi alocar as necessidades a repor em posições que já possuem *stock*, mas que ainda têm capacidade disponível. Esta foi calculada através da diferença entre a capacidade máxima e a quantidade já alocada na posição, e é ordenada de forma decrescente. A Figura 29 apresenta a quantidade em *stock*, assim como a posição e o tipo associado e a capacidade máxima do mesmo. Além disso, expõe também a capacidade disponível para armazenar as necessidades de reposição.

| id_artigo | posição | qt | tipo | Capacidade máxima | Capacidade disponível |
|---|---|---|---|---|---|
| X0245 | PX128A1 | 100 | PALETE | 6000 | 5900 |
| X3039 | PL101F1 | 164 | A4 | 1152 | 988 |
| X0026 | PAA2B2 | 27 | A3 | 3840 | 3813 |
| X9609 | PV1E3 | 45 | A2 | 360 | 315 |
| X7649 | PS10G4 | 29 | A2 | 528 | 499 |
| X2735 | PE12B3 | 77 | A3 | 307 | 230 |
| X1128 | PX46A1 | 6 | PALETE | 300 | 294 |
| X6189 | PQ3E1 | 36 | A2 | 528 | 492 |
| X6189 | PV1G1 | 139 | A2 | 528 | 389 |
| X4199 | PG9C1 | 9 | A3 | 480 | 471 |

Figura 29 - Posições com *stock* e capacidade disponível

Caso não existisse esta ordenação da capacidade disponível, nos casos em que houvesse duas posições passíveis de comportar as necessidades a repor e estas pudessem ser armazenadas em apenas uma posição com *stock* (na posição com maior capacidade disponível), o código alocava as necessidades a repor nas duas posições parcialmente ocupadas, proporcionando duas reposições, quando era possível realizar apenas uma reposição. Considerando a política da empresa de armazenar até ao limite máximo da posição, mesmo que a capacidade disponível seja superior às necessidades de reposição, o sistema vai indicar como quantidade a alocar, a capacidade disponível, isto é, o máximo que ainda é possível de armazenar na posição com *stock*. O pseudocódigo referente a esta etapa da segunda parte da solução proposta encontra-se na Figura 30.

```
1. variables: Capacidade máxima, qt, necessidades_repor
2. Capacidade disponível = Capacidade máxima – qt
3. ordenar Capacidade disponível em modo descendente
4. If Capacidade disponível ≥ necessidades_repor then:
       alocar Capacidade disponível
       necessidades_repor = 0
   else
       alocar Capacidade disponível
       necessidades_repor = necessidades repor – Capacidade disponível
```

Figura 30 - Pseudocódigo relativo à reposição das necessidades em posições com *stock*

De seguida, nas situações em que os artigos não apresentam posições com *stock* ou quando estas não são suficientes para armazenar todas as necessidades a repor, após ocupar a capacidade disponível dessas posições e considerando as necessidades que ainda faltam repor, o código vai minimizar o número de posições e reposições.

Para tal, procura um tipo de posição na Tabela 9, onde seja possível armazenar as necessidades que faltam repor, procurando um tipo que a partir da menor área, tenha uma capacidade ajustada maior ou igual à quantidade a repor em falta. Após escolher o tipo, entre as posições livres, o algoritmo seleciona uma posição desse tipo para proceder à sua alocação, sendo que aloca até ao limite máximo da posição mesmo que não haja essa necessidade. A partir do momento em que uma posição da lista de posições livres é escolhida para armazenar um artigo, essa posição é

retirada da lista. O pseudocódigo desenvolvido nesta fase da segunda parte da solução proposta está disponível na Figura 31.

```
1. variables: capacidade_ajustada, Capacidade disponível, necessidades_repor, posições_com_stock,
   posições_existentes
2. posições_livres = posições_existentes – posições_com_stock
3. while necessidades_repor > 0:
       escolher tipo com capacidade_ajustada ≥ necessidades_repor
           selecionar posição do tipo escolhido em posições_livres
               alocar Capacidade disponível
               necessidades_repor = 0
               posições_livres = posições_livres – posição selecionada
```

Figura 31 - Pseudocódigo relativo à reposição das necessidades em falta em posições livres

Há que referir ainda a possibilidade de existirem situações em que já não haja posições disponíveis do tipo selecionado, ou os tipos possíveis para alocar determinado artigo não terem uma capacidade ajustada maior ou igual às suas necessidades de reposição, sendo que nestes casos há duas possibilidades a seguir.

A primeira tentativa passa por selecionar outro tipo de posição com a maior capacidade ajustada e escolher uma posição para esse tipo, sendo que se mais uma vez não houver posições disponíveis para esse tipo, passa-se à segunda tentativa onde se seleciona o tipo com a menor capacidade ajustada disponível, seguindo posteriormente a lógica da escolha de posição para esse tipo. O método repete-se até todas as necessidades a repor do artigo estarem armazenadas, não esquecendo que em cada posição escolhida, a alocação é feita até ao limite máximo da sua capacidade.

No exemplo que se está a seguir, ocorre uma destas situações no artigo X0102. Através da Figura 28, observa-se que as necessidades de reposição para este artigo são 208 unidades. Na Figura 29, verifica-se que este artigo não dispõe de posições com *stock,* portanto segue para a alocação das necessidades a repor em posições livres. Além disso, os tipos disponíveis para alocar este artigo seguem na Tabela 10 e foram retirados do APÊNDICE D

| id_artigo | tipo | capacidade_ajustada | Área (cm²) |
|---|---|---|---|
| X0102 | A9 | 2 | 700 |
| X0102 | A6 | 1 | 800 |
| X0102 | A2 | 2 | 1200 |
| X0102 | A3 | 7 | 2400 |
| X0102 | A4 | 21 | 9600 |
| X0102 | PALETE | 144 | 9600 |

Tabela 10 - Lista dos tipos possíveis de armazenar o artigo X0102

Neste sentido, dado que não existe um tipo com capacidade ajustada maior ou igual às necessidades de reposição, o algoritmo proposto segue a primeira tentativa e seleciona o tipo com maior capacidade ajustada disponível, o tipo PALETE, escolhendo de seguida uma posição desse tipo e alocando a mesma até ao seu limite máximo de 144 unidades. No entanto continuam em falta a reposição de 64 unidades (208-144=64). Seguindo a lógica descrita acima, o modelo procura o tipo com a menor área possível que tenha uma capacidade ajustada superior ou igual às necessidades que faltam alocar. Mais uma vez esse tipo é PALETE, e são alocadas mais 144

unidades. Como se verifica na Tabela 11, que representa as alocações a realizar pela solução proposta, o exemplo demonstrado está correto, pois o artigo X0102 é reposto duas vezes em duas posições diferentes do tipo PALETE, sendo a quantidade alocada correspondente à capacidade máxima ajustada deste artigo para este tipo de posição.

| id_artigo | posição | quantidade alocada | tipo | *stock* |
|---|---|---|---|---|
| X0245 | PX128A1 | 5900 | PALETE | 100 |
| X3065 | PA43F1 | 504 | A9 | 0 |
| X3049 | PL103D3 | 576 | A4 | 0 |
| X3039 | PL101F1 | 988 | A4 | 164 |
| X9175 | PA4M1 | 218 | A3 | 0 |
| X0026 | PAA2B2 | 3813 | A3 | 27 |
| X9609 | PV1E3 | 315 | A2 | 45 |
| X7649 | PS10G4 | 499 | A2 | 29 |
| X2735 | PE12B3 | 230 | A3 | 77 |
| X1128 | PX46A1 | 294 | PALETE | 6 |
| X6189 | PQ3E1 | 492 | A2 | 36 |
| X4652 | PX23A1 | 2618 | PALETE | 0 |
| X0052 | PA43A2 | 334 | A9 | 0 |
| X5191 | PL103E2 | 594 | A4 | 0 |
| X4804 | PX15A1 | 981 | PALETE | 0 |
| X0846 | PA32B2 | 546 | A9 | 0 |
| X4373 | PL103E5 | 299 | A4 | 0 |
| X3422 | PX11A1 | 2086 | PALETE | 0 |
| X4802 | PX7A1 | 240 | PALETE | 0 |
| X2912 | PX4A1 | 1636 | PALETE | 0 |
| X2949 | PX3A1 | 818 | PALETE | 0 |
| X0102 | PX2A1 | 144 | PALETE | 0 |
| X0102 | PX1A1 | 144 | PALETE | 0 |
| X4199 | PG9C1 | 471 | A3 | 9 |
| X3865 | PL103F1 | 384 | A4 | 0 |

Tabela 11 – Alocação às posições através da proposta de solução

Caso não houvesse nenhuma posição disponível para os tipos possíveis de armazenar os artigos referenciados neste exemplo, o código transmitiria o seguinte *output* “Não há posições disponíveis para alocar o artigo X com necessidades a repor de y.".

Ainda no que diz respeito a esta segunda fase, é de salientar que após alocar as necessidades a repor em posições com *stock*, a letra ou letras correspondentes ao corredor da última posição com *stock* é “armazenada”, para no momento de escolha da posição na lista de posições livres, caso seja possível opte por uma posição com a mesma letra, indicando que as posições se situam no mesmo corredor. Caso não haja essa possibilidade, opte por escolher uma posição mais próxima do corredor indicado pela letra “armazenada” da última posição com *stock* ocupada. A proximidade das posições é determinada pela ordem dos corredores definida e exposta no código relativo à alocação nas necessidades a repor a uma ou mais posições, seguindo a lista de artigos ordenada pela sua prioridade. Este código encontra-se no APÊNDICE E

Desta forma, o objetivo da solução proposta é priorizar a alocação de posições de *picking* para artigos com maior margem bruta e relevância nas vendas, além de garantir que os artigos considerados indispensáveis estejam sempre disponíveis nas posições de *picking*. Considerando

que pode não ser possível que todos os artigos com necessidades de reposição sejam reabastecidos diariamente, é crucial que os considerados essenciais, isto é, os artigos que caso faltem, faria com que os clientes recorressem a outros estabelecimentos, tenham prioridade na alocação, evitando ruturas de *stock*, tanto em loja, como nas próprias posições de *picking*. Por outro lado, esta solução pretende também melhorar o processo de reabastecimento das posições de *picking* no centro de distribuição, considerando o espaço livre e ocupado que existe no momento, de forma a primeiramente minimizar a área das posições ocupadas (anteriormente livres) e de seguida minimizar o número de posições ocupadas (anteriormente livres), que por sua vez diminuirá o número de reposições necessárias.

Tal como referido anteriormente, apenas para o primeiro dia de teste foi considerado o *stock* de início do dia disponibilizado pela empresa. Assim, após o desenvolvimento deste modelo, foi necessário criar um algoritmo que calculasse o *stock* do dia seguinte, através do *stock* atualizado do próprio dia e com base no resultado obtido pelas alocações geradas pela solução proposta.

Nesse sentido, o algoritmo desenvolvido utiliza o *stock* atualizado do próprio dia, adiciona as alocações da solução proposta quando o artigo e a posição coincidem, e gera "novo *stock*" no caso em que as alocações são realizadas em posições livres. O pseudocódigo associado a esta operação está representado na Figura 32, enquanto o código está presente no APÊNDICE F

```
1. variables: id_artigo_stock_atualizado, id_artigo_alocação, posição_stock_atualizado, posição_alocação, qt,
   quantidade_alocada
2. If id_artigo_stock_atualizado = id_artigo_alocação and posição_stock_atualizado = posição_alocação then:
       qt = qt + quantidade_alocada
   else
       qt = quantidade_alocada
```

Figura 32 - Pseudocódigo relativo ao cálculo do stock de início do dia do dia seguinte

Posto isto, ainda de acordo com o exemplo que se está a seguir, na Tabela 12 está representado na prática a realização do pseudocódigo referido acima.

| id_artigo | posição | qt | tipo |
|---|---|---|---|
| X0245 | PX128A1 | 6000 | PALETE |
| X3039 | PL101F1 | 1152 | A4 |
| X0026 | PAA2B2 | 3840 | A3 |
| X9609 | PV1E3 | 360 | A2 |
| X7649 | PS10G4 | 528 | A2 |
| X2735 | PE12B3 | 307 | A3 |
| X1128 | PX46A1 | 300 | PALETE |
| X6189 | PQ3E1 | 528 | A2 |
| X6189 | PV1G1 | 139 | A2 |
| X4199 | PG9C1 | 480 | A3 |
| X3065 | PA43F1 | 504 | A9 |
| X3049 | PL103D3 | 576 | A4 |
| X9175 | PA4M1 | 218 | A3 |
| X4652 | PX23A1 | 2618 | PALETE |
| X0052 | PA43A2 | 334 | A9 |
| X5191 | PL103E2 | 594 | A4 |
| X4804 | PX15A1 | 981 | PALETE |
| X0846 | PA32B2 | 546 | A9 |
| X4373 | PL103E5 | 299 | A4 |
| X3422 | PX11A1 | 2086 | PALETE |
| X4802 | PX7A1 | 240 | PALETE |
| X2912 | PX4A1 | 1636 | PALETE |
| X2949 | PX3A1 | 818 | PALETE |
| X0102 | PX2A1 | 144 | PALETE |
| X0102 | PX1A1 | 144 | PALETE |
| X3865 | PL103F1 | 384 | A4 |

Tabela 12 - *Stock* do dia seguinte

## 3.5. Síntese do processo proposto

O projeto proposto inclui 5 fases, descritas aqui de uma forma bastante sintetizada:

- Determinação do *stock* atualizado com base no *stock* do início do dia e nas saídas do próprio dia e posterior cálculo das necessidades de reposição, considerando as necessidades das lojas do dia seguinte;
- Criação de pares entre os tipos de posições e artigos com necessidades de reposição, com base na dimensão dos artigos e dos tipos de posições, e considerando o volume dos tipos de posições, de modo a obter para cada combinação um limite da quantidade máxima do artigo que é possível armazenar nos tipos de posições associados;
- Criação de uma lista de prioridades, determinada pela importância do artigo e pela sua margem bruta;
- Alocação das quantidades a repor na posição ou posições mais convenientes até ao seu limite máximo de capacidade, de modo a minimizar a área e as posições ocupadas e reduzindo o número de reposições;
- Cálculo do *stock* do dia seguinte considerando o *stock* atualizado e as alocações da solução proposta.

É de salientar que este processo segue uma lógica, pois não é possível determinar o *stock* atualizado, as necessidades de reposição dos dias em análise, assim como as outras fases descritas de forma simultânea. Assim, segue na Figura 33 uma representação esquemática bastante abrangente deste processo.

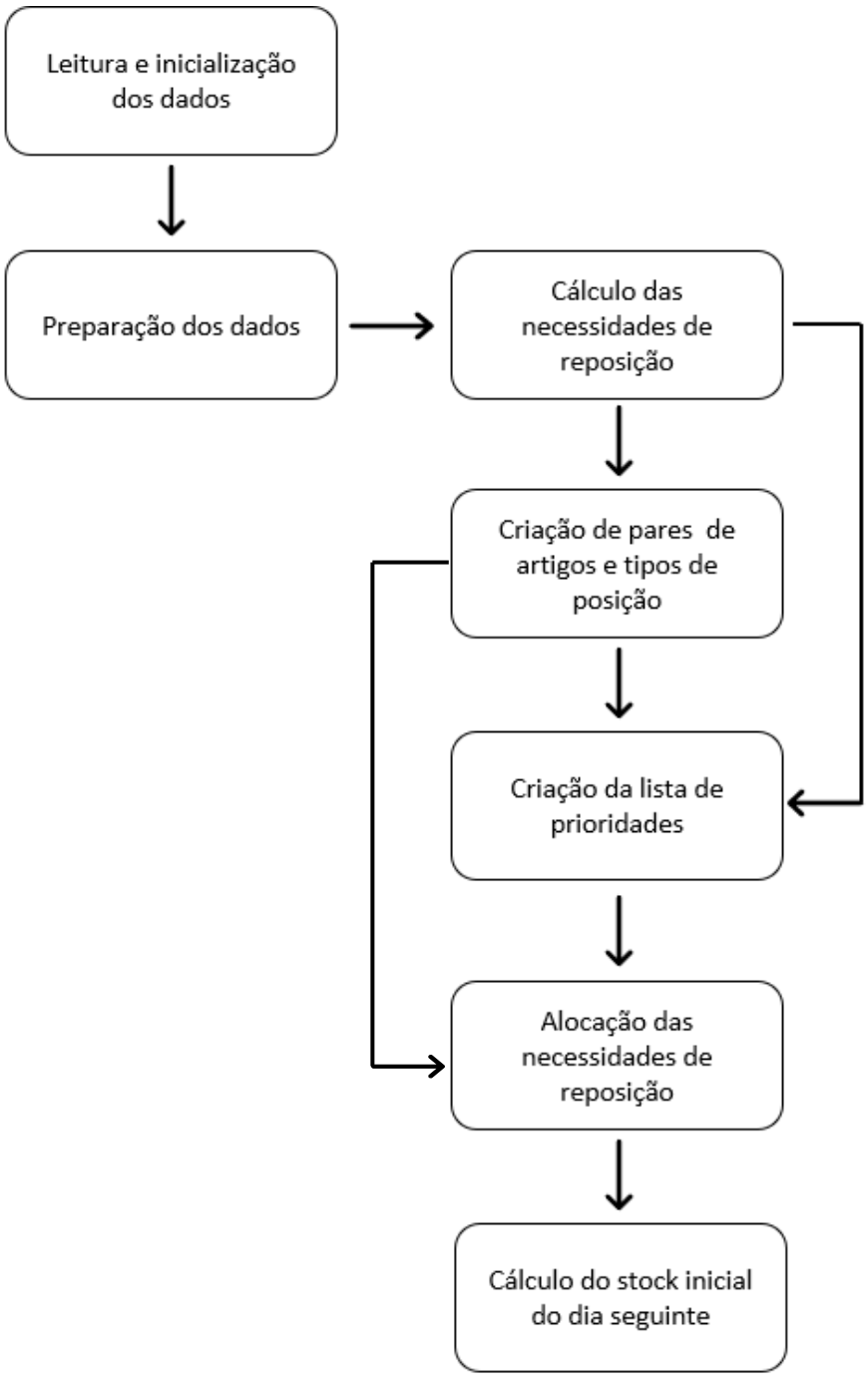


Figura 33 - Representação esquemática do processo de apoio à decisão

# 4. RESULTADOS E DISCUSSÃO

A implementação da solução proposta no capítulo anterior fornece um ficheiro *excel* com a informação relativa à alocação da quantidade a repor de cada artigo na posição ou posições necessárias. Além disso, o código gera também um *output* com os artigos e as necessidades que não são possíveis de alocar por não haver posições disponíveis para tal. É de salientar que para o período em análise, o código não revela a existência de artigos por alocar por falta de posições disponíveis. Nesse sentido, todos os artigos com necessidades de reposição foram armazenados em posições até ao limite máximo da sua capacidade.

A Tabela 13 expõe para cada dia em análise os efeitos da aplicação da solução proposta, identificando o número de artigos com necessidades de reposição, a quantidade associada a esta necessidade com o critério da alocação ser efetuada até ao limite máximo da capacidade da posição, e o número de reposições necessárias para o seu reabastecimento.

| Data | Nº artigos | Qt a repor | Nº reposições |
|---|---|---|---|
| 09/09/2024 | 867 | 144480 | 883 |
| 10/09/2024 | 443 | 42988 | 467 |
| 11/09/2024 | 509 | 53878 | 530 |
| 12/09/2024 | 401 | 59600 | 416 |
| 13/09/2024 | 758 | 106272 | 845 |
| **Total** | 2978 | 407218 | 3141 |

Tabela 13 - Resultados da alocação do modelo proposto

Por outro lado, a Tabela 14 apresenta a situação real analisada anteriormente. É importante destacar que, após obter os resultados da alocação do primeiro dia, os dados de *stock* utilizados não foram os mesmos, o que justifica a diferença observada no número de artigos com necessidades de reposição e na quantidade a alocar. Além disso, esta quantidade a alocar apresenta uma margem de erro, já que a capacidade máxima de unidades possíveis de armazenar de cada artigo em cada tipo de posição foi calculada com base numa relação proporcional entre volumes dos diferentes tipos de posições, tendo como referência a capacidade máxima de um tipo em específico para cada artigo.

| Data | Nº artigos | Qt a repor | Nº reposições |
|---|---|---|---|
| 09/09/2024 | 885 | 94596 | 1198 |
| 10/09/2024 | 941 | 70668 | 1149 |
| 11/09/2024 | 608 | 47056 | 745 |
| 12/09/2024 | 948 | 87877 | 1259 |
| 13/09/2024 | 468 | 44727 | 611 |
| **Total** | 3850 | 344924 | 4962 |

Tabela 14 - Alocação realizada pela empresa

Analisando a solução proposta ao fim de uma semana, é evidente que esta reduz em 22,6% o número de artigos efetivamente repostos e em 36,7% o número de reposições realizadas. No entanto, em termos de quantidade, permite um aumento de 18,1% na quantidade alocada em comparação com situação atual da empresa.

No que diz respeito à alocação das necessidades a repor em posições com *stock* ou em posições livres, o comportamento da solução proposta é totalmente diferente da situação real. O desempenho da empresa nesse sentido está disponível na Figura 24. Enquanto o sistema utilizado pela empresa opta por alocar as necessidades de reposição na sua maioria em posições livres, com exceção do dia 12. A solução proposta, de forma a minimizar a área das posições ocupadas, opta por armazenar, sempre que possível as necessidades a repor em posições com *stock*, como se verifica na Figura 34.

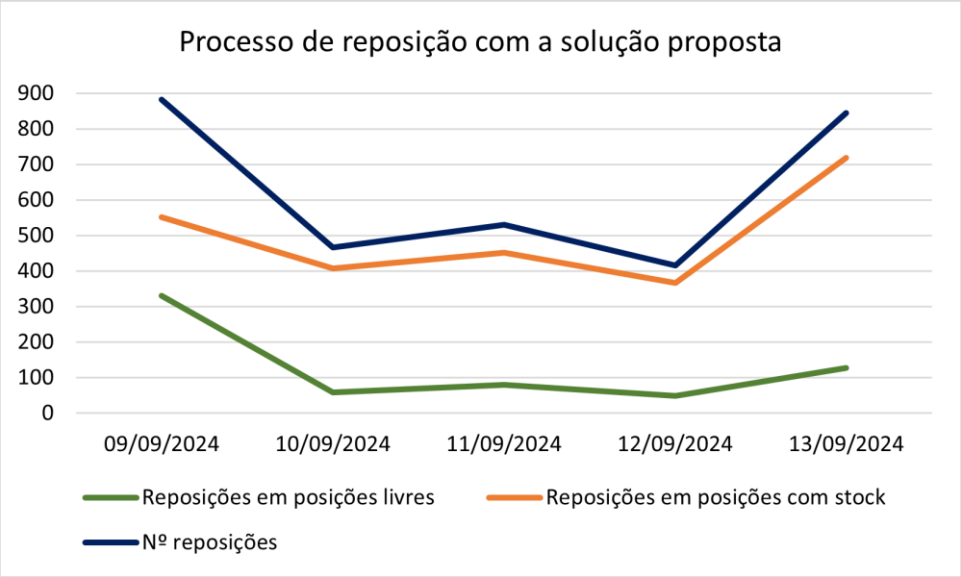


Figura 34 - Processo de reposição das posições de *picking* através da solução proposta

Considerando a diferença bastante evidente entre o número de reposições em posições livres a realizar pela proposta de solução na Figura 34 e a mesma variável na realidade atual da empresa, exposta na Figura 24, já seria de esperar que a área em $cm^2$ ocupada por posições novas, seguisse a mesma tendência. Assim, segue na Figura 35, a variação da área ocupada na reposição efetiva de posições livres e a área que seria preenchida por posições livres com o modelo proposto. Além disso, a empresa opta por armazenar artigos sobretudo em posições do tipo A2 e A3 com uma área de 1200 $cm^2$ e 2400 $cm^2$, respetivamente, enquanto a solução proposta prioriza a alocação dos artigos pela menor área, apresentando uma distribuição mais equilibrada das necessidades a repor pelos tipos de posição.

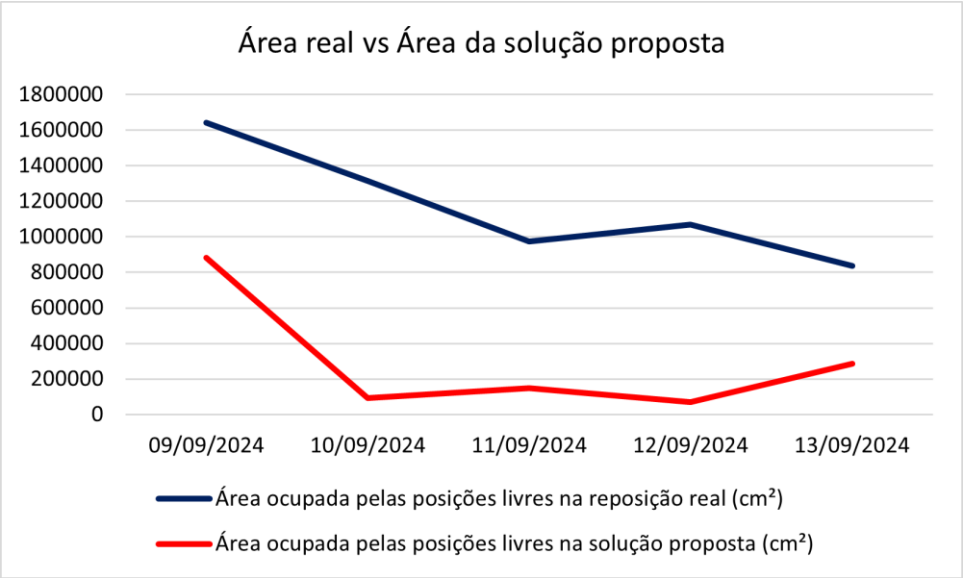


Figura 35 - Variação da área real e da solução proposta correspondente às reposições em posições livres

Após a demonstração dos resultados obtidos é possível confirmar que, para a semana em análise, o modelo proposto cumpre com os seus objetivos, e quando comparado à situação atual da empresa, apresenta melhores resultados nas 3 variáveis que se pretendiam desenvolver (área utilizada, número de posições utilizadas e número de reposições).

Assim, a solução proposta visa melhorar o processo de reabastecimento das posições de *picking* em 3 níveis. Inicialmente minimizando a área das posições, sendo que a única forma de isto acontecer seria armazenar as necessidades de reposição em posições com *stock*, de modo que as necessidades a repor em falta necessitassem do menor número possível de posições livres.

O nível seguinte está diretamente ligado ao anterior e passa pela minimização do número de posições, mas considera também a minimização da área. Caso contrário, as necessidades de reposição em falta seriam diretamente alocadas a posições do tipo PALETE e A4, pois são as posições de maior dimensão. Assim, considerando as posições livres, o modelo proposto escolhe a posição com menor área que tenha uma capacidade máxima ajustada maior ou igual à quantidade a alocar. Desta forma, o algoritmo também está a minimizar o terceiro nível, o número de reposições, pois prioriza a escolha de apenas uma posição. Apenas nos casos em que não há posições disponíveis do tipo adequado, o modelo opta por escolher o tipo com maior capacidade máxima, continuando a minimizar o número de reposições. Apenas em último recurso, na impossibilidade de existirem posições disponíveis desse tipo, é que passa à tentativa de escolher um tipo com a menor capacidade máxima, obrigando a que o número de reposições aumente para alocar todas as necessidades em falta.

Posto isto, através da Figura 35, verifica-se que a área das posições livres, ocupadas durante o processo de reposição é menor no modelo proposto do que na realidade atual da empresa. A Figura 34 e Figura 24 revelam o motivo da diferença tão evidenciada entre as duas áreas. Na Figura 34, que expõe as reposições para a solução proposta, é possível observar que o número de reposições em posições livres é bastante inferior às posições com *stock* e por outro lado é bastante inferior ao número de reposições em posições livres da situação real da empresa, representada na Figura 24, onde as reposições em posições livres ultrapassam em 4 dos 5 dias as reposições em posições com

*stock*. Da Figura 24, retira-se que em média, no período analisado, a proporção de reposições em posições livres no total de reposições é de 61%, enquanto na solução proposta, representada pela Figura 34, este valor é de 20,5%. Em termos gerais, a utilização de reposições em posições livres diminuiu 78,7% com o modelo proposto ao fim da semana em análise.

Já na Tabela 13 e Tabela 14, verifica-se a potencialização do terceiro nível, sendo que o total do número de reposições sofreu uma redução, passando de 4962 reposições efetivas para 3141 no modelo proposto.

É ainda de salientar as diferenças obtidas no *stock* de início do dia disponibilizado pela empresa, que apenas foi usado no dia 09 e o *stock* calculado com base nas alocações da solução proposta. Como se observa na Figura 36, há uma diminuição na ocupação das posições de *picking*, na ordem dos 4% aos 6%.

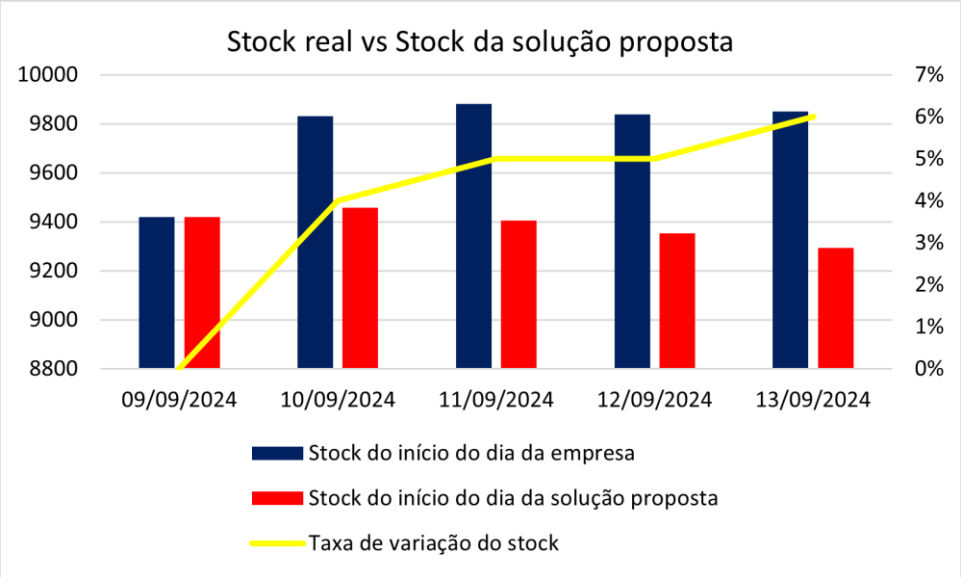


Figura 36 - Variação do *stock* de início do dia facultado pela empresa e calculado pela solução proposta

Assim, ao longo deste capítulo e dos anteriores, é possível compreender a complexidade do tema abordado, tanto pela sua amplitude como pela execução num cenário de operações real. A dificuldade em compreender e absorver todos os processos, que nem sempre são claros, tornou a realização deste projeto ainda mais desafiante.

Este caso de estudo desenvolveu um sistema de apoio à decisão para o processo de reposição de artigos nas posições de *picking* do centro de distribuição. O modelo proposto apresenta uma abordagem inovadora, ao considerar fatores que a empresa atualmente não contempla na lógica da sua operação. O principal objetivo deste projeto, além de alocar artigos com necessidades de reposição às posições, é considerar esta reposição com base na prioridade dos artigos. Há artigos que caso estejam em rutura nas lojas, os clientes dirigem-se a outros estabelecimentos para os adquirirem. Por isso, a solução proposta atribui prioridade a este tipo de artigos essenciais, fundamentais para a satisfação dos clientes. Além disso, são priorizados artigos com maior margem bruta, com o intuito de maximizar os resultados financeiros da empresa.

Destaca-se também que o algoritmo desenvolvido permite uma aplicação ágil, oferecendo ao responsável a capacidade e liberdade de tomar decisões com base na lista ordenada de prioridades e de realizar ajustes que considere necessários, permitindo alterar parâmetros de forma rápida e obter diferentes cenários.

Posto isto, este trabalho minimiza o espaço ocupado no centro de distribuição, o que é uma mais-valia, ao mesmo tempo que reduz o número de reposições a serem realizadas pelos operadores. Assim, as poupanças obtidas no final do período de análise com o modelo proposto resultam numa redução de 74,7% na área das reposições em posições livres, e uma diminuição de 78,7% no número de reposições em posições livres, o que se traduziu numa diminuição na ocupação das posições de *picking* na ordem dos 4% aos 6%. No que diz respeito à carga de trabalho, os operadores teriam evitado a realização de cerca de 36,7% das reposições no final da semana. Por fim é de salientar que o modelo de apoio à decisão proposto carregou ficheiros com um máximo de cerca de 10000 linhas, sendo que o processamento de todas as suas fases foi sempre inferior a 1 minuto.

página propositadamente em branco

# 5. CONCLUSÃO

Este capítulo agrega as conclusões obtidas com a realização do projeto, bem como a principal limitação enfrentada. Adicionalmente, são propostas direções para trabalhos futuros, com vista a dar continuidade às questões abordadas neste estudo.

## 5.1. Conclusões finais

Esta dissertação centra-se na gestão de inventários no setor do retalho. No entanto, por ser um tópico tão abrangente e complexo, a sua incidência e principal objetivo passa por tornar o reabastecimento das posições de *picking* mais eficiente. Idealmente, uma empresa de retalho deve garantir que o centro de distribuição disponha de inventário suficiente para a operação de reposição das lojas, evitando assim a rutura de artigos em loja e antecipando a rutura nas posições de *picking*. Por outro lado, o nível de inventário pode escalar rapidamente, e o armazenamento de *stock* em excesso revela consequências não só financeiras, mas também ao nível de possíveis ruturas de artigos por falta de espaço. Assim, é um desafio e um fator de sucesso equilibrar a oferta de *stock* nas posições de *picking* com as necessidades das lojas. Desta forma, é crucial adotar técnicas e modelos de gestão de inventários eficazes para uma empresa de retalho se tornar competitiva no mercado atual, pois uma simples mudança pode resultar na redução de custos, no adiamento da necessidade de um novo centro de distribuição com mais condições, no melhor aproveitamento do espaço existente ou até na melhoria dos níveis de serviço ao cliente ao garantir a disponibilidade de artigos nas lojas.

Inicialmente, foi fundamental estudar toda a operação interna do centro de distribuição, ligada ao processo de reabastecimento das posições de *picking*, nomeadamente o próprio processo de *picking*, mas também a operação de reposição das lojas. Depois de analisar as principais limitações, sobretudo relacionadas com a ineficiência das posições utilizadas, com a determinação aleatória das posições escolhidas e a frequência de reposições, sendo algumas destas em vão por ocuparem temporária e parcialmente as posições, foi possível concluir que as principais consequências resultantes desta gestão ineficaz são a rutura nas posições de *picking*, a ocupação inadequada do espaço disponível e a reposição frequente do mesmo artigo. Posto isto, o objetivo era criar um sistema de apoio à decisão que permitisse aprimorar a gestão deste processo. É de salientar que a proposta de solução apresentada pretende controlar estas limitações, de forma a minimizar a área do espaço ocupado, por conseguinte evitar o excesso de posições ocupadas e o excesso do número de reposições.

Desta forma, o modelo proposto determina uma ou mais posições para alocar as necessidades a repor de cada artigo com necessidades de reposição até ao limite máximo da capacidade da posição. O objetivo da solução era priorizar a alocação das posições de *picking* a artigos com maior prioridade, sendo esta prioridade estimada pela margem bruta de cada artigo e pela sua importância, isto é, caso o artigo estivesse inserido na lista de artigos indispensáveis, era abrangido por um indicador ajustável, caso contrário, nesse parâmetro não teria relevância. Além disso, a solução obtida segue uma hierarquia de critérios na sua concretização, inicialmente minimiza a área das novas posições a ocupar, de seguida tem em consideração a redução do número de novas posições e por fim a diminuição do número de reposições, garantindo que não há reposições

efetuadas para ocupar posições temporariamente. Nesse sentido, não havendo um limite predeterminado da capacidade máxima de reposição das posições de *picking*, cabe ao responsável tomar a decisão de até onde é possível seguir a lista de alocações ordenada pela prioridade dos artigos.

Os resultados obtidos pelo modelo proposto para a semana em análise foram superiores nas 3 variáveis que se pretendiam melhorar e consequentemente, no *stock* ao início de cada dia. Através da comparação entre a situação real apresentada pela empresa e o modelo da nova solução, esta proporcionaria uma diminuição média de 74,7% na área das novas posições utilizadas. No que diz respeito ao número de posições livres repostas, na realidade atual da empresa estas situam-se nos 61%, enquanto com o modelo aplicado teriam um efeito menor, encontrando-se nos 20,5%, revelando uma redução na utilização de reposições em posições livres na ordem dos 78,7% quando comparado com o modelo proposto no final da semana analisada. Já as reposições obteriam uma diminuição de 36,7%, reduzindo a carga de trabalho dos operadores. Por fim, comparando o *stock* de início do dia facultado pela empresa, com o *stock* calculado com base nas alocações da solução proposta, existe uma diminuição na ocupação das posições de *picking* compreendida entre os 4% e os 6%. É ainda de salientar que em termos de tempo computacional, este foi sempre inferior a 1 minuto.

Posto isto, é possível concluir que para a semana em análise, o modelo proposto é mais adequado, pois apresenta um melhor desempenho que o método atualmente usado pela empresa, cumprindo os objetivos de minimizar as 3 variáveis referidas e alocando todos os artigos que fazem parte da lista dos artigos indispensáveis.

Por fim, é importante destacar a complexidade que este projeto exigiu, não apenas em termos de compreensão de todos os processos inerentes, mas sobretudo por se tratar de processos num ambiente empresarial real, que acabam por revelar exceções no sentido do objetivo e critérios definidos para cada operação não serem seguidos na sua totalidade, por parte de operadores, por exemplo. No entanto, foi possível propor uma abordagem inovadora ao introduzir a alocação dos artigos às posições, consoante a prioridade dos mesmos, algo que o modelo atual da empresa não contempla. Além disso, o algoritmo desenvolvido oferece grande flexibilidade e permite rapidamente a realização de ajustes, facilitando a tomada de decisões por parte do responsável.

## 5.2. Limitações e investigação futura

Apesar dos resultados obtidos com a aplicação do algoritmo proposto de reabastecimento das posições de *picking*, é de salientar que pode haver uma pequena margem de erro no sentido que foram calculadas capacidades máximas ajustadas dos artigos para cada tipo de posição, através de uma relação proporcional entre os diferentes volumes de tipos de posições, tendo como base os dados facultados para a capacidade máxima de um tipo de posição em específico, sendo que em algumas situações essa capacidade máxima suportava mais unidades do que a definida. Por isso mesmo, poderiam ter sido evitadas a utilização de algumas posições e consequentemente reposições. No entanto, a ordem de grandeza dos resultados absorve esta margem de erro.

Relativamente a possíveis direções a seguir, é de salientar a integração de previsões da procura, isto é, a substituição das atuais necessidades das lojas por previsões da procura, utilizando técnicas de *machine learning*. A adição de algoritmos preditivos baseados em dados históricos e fatores

externos, como lançamento de campanhas, sazonalidade, fins-de-semana e feriados, podem proporcionar uma gestão de inventários mais eficiente e adaptável às flutuações da procura. Através da previsão da procura seria possível alargar o horizonte temporal, de modo a evitar que o trabalhador tenha de efetuar a reposição do mesmo artigo todos os dias.

Outra linha de investigação pode focar-se no abastecimento direcionado a zonas específicas da área de *picking*. Ao identificar áreas de maior rotatividade de artigos, necessidades de reposição mais frequentes, ou mesmo artigos com maior margem bruta ou que sejam indispensáveis de ter em loja, o modelo poderia sugerir locais estratégicos de reposição. Nesse sentido, o modelo atual pode ser expandido para lidar com *clusters* de artigos, permitindo que estes fossem agrupados através de uma característica em comum previamente definida. A aplicação de algoritmos de *clustering* facilitaria o processo de reabastecimento e poderia aumentar a eficiência do armazenamento, otimizando a utilização do espaço.

Outra vertente a ser explorada é a incorporação de outro tipo de métricas, além da área, número de posições e reposições, nomeadamente indicadores que envolvam custos associados à reposição, por exemplo.

página propositadamente em branco

# REFERÊNCIAS BIBLIOGRÁFICAS


Agù, A. O., & Nnate, E. C. (2016). EFFECT OF INVENTORY MANAGEMENT ON THE ORGANIZATIONAL PERFORMANCE OF THE SELECTED MANUFACTURING FIRMS Obi-Anike, Happiness Ozioma. In *Singaporean Journal of BuSineSS economicS, and management StudieS (SJBem)* (Vol. 5, Issue 4). www.singaporeanjbem.com

Ahmad, K., & Zabri, M. (2016). Inventory management practices among Malaysian micro retailing enterprises. *Www.Jbrmr.Com A Journal of the Academy of Business and Retail Management*, *11*. www.jbrmr.com

Albayrak Ünal, Ö., Erkayman, B., & Usanmaz, B. (2023). Applications of Artificial Intelligence in Inventory Management: A Systematic Review of the Literature. *Archives of Computational Methods in Engineering*, *30*(4), 2605–2625. https://doi.org/10.1007/S11831-022-09879-5/FIGURES/3

Alsheikh, M. A., Hoang, D. T., Niyato, D., Tan, H.-P., & Lin, S. (2015). Markov Decision Processes with Applications in Wireless Sensor Networks: A Survey. *IEEE Communications Surveys and Tutorials*, *17*(3), 1239–1267. https://doi.org/10.1109/COMST.2015.2420686

Alves, R. (2016). *Aplicação de Modelos de Redes Neuronais para Previsão de Consumos de Energia Engenharia Mecânica*.

Aro-Gordon, S., & Gupte, J. A. (2016). *Review of modern inventory management techniques*. https://www.researchgate.net/publication/307966411

Ballou, R. H., & Srivastava, S. K. (2007). *Business logistics/supply chain management: planning, organizing, and controlling the supply chain.* 13–16.

Barrett, E., Howley, E., & Duggan, J. (2013). Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and Computation: Practice and Experience*, *25*(12), 1656–1674. https://doi.org/10.1002/CPE.2864

Bonaccorso, Giuseppe. (2017). *Machine Learning Algorithms.* Packt Publishing.

Bose, D. C. (2006). *INVENTORY MANAGEMENT - D. CHANDRA BOSE - Google Livros*.

Cardoso, A. P. (2014). *Inovar com a investigação-ação: desafios para a formação de professores - Ana Paula Pereira de Oliveira Cardoso - Google Livros*.

Chebet, E., & Kitheka, S. (2019). Effects of Inventory Management System on Firm Performance-An Empirical Study. *International Journal of Innovative Science and Research Technology*, *4*(9). www.ijisrt.com91

Cielen, D., Meysman, A. D. B., & Ali, M. (2016). *Introducing Data Science*.

Claycomb, C., Germain, R., & Dröge, C. (1999). Total system JIT outcomes: Inventory, organization and financial effects. *International Journal of Physical Distribution & Logistics Management*, *29*(10), 612–630. https://doi.org/10.1108/09600039910299940/FULL/XML

Coppin, B. (2004). *Artificial Intelligence Illuminated - Ben Coppin - Google Livros*.

Creswell, J. (2012). *Educational Research*.

Cunha, B., Madureira, A. M., Fonseca, B., & Coelho, D. (2020). Deep Reinforcement Learning as a Job Shop Scheduling Solver: A Literature Review. *Advances in Intelligent Systems and Computing*, *923*, 350–359. https://doi.org/10.1007/978-3-030-14347-3_34

Domingos, P. (2017). *O Algoritmo Mestre Como a busca pelo algoritmo de machine learning*. https://www.academia.edu/69649534/O_Algoritmo_Mestre_Como_a_busca_pelo_algoritmo_de_machine_learning

Esther, U. U. (2012). *COVER PAGE EFFECTIVENESS OF INVENTORY MANAGEMENT IN A MANUFACTURING COMPANY (A CASE STUDY OF AMA GREENFIELD BREWERIES PLC, ENUGU, NIGERIA)*.

Eveline, C., Kitheka, S., Charles, C., James, O., & Abeid, T. (2019). Effects of Inventory Management Techniques on Procurement Performance: An Empirical Study. *International Journal of Innovative Research and Development*, *8*(8). https://doi.org/10.24940/IJIRD/2019/V8/I8/AUG19072

Fudenberg, D., & Tirole, J. (1991). *Game Theory*. https://mitpress.mit.edu/9780262061414/game-theory/

Garcia, F., & Rachelson, E. (2013). Markov Decision Processes. *Markov Decision Processes in Artificial Intelligence: MDPs, beyond MDPs and Applications*, 1–38. https://doi.org/10.1002/9781118557426.CH1

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. https://www.deeplearningbook.org/

Hall, P., Phan, W., & Whitson, K. (2016). *Opportunities and Challenges for Machine Learning in Business The Evolution of Analytics*.

James, A., Otieno, B., Nyang'au Paul, S., & Mbura, L. K. (2018). INFLUENCE OF INVENTORY MANAGEMENT PRACTICES ON PERFORMANCE OF RETAIL OUTLETS IN NAIROBI CITY COUNTY. *International Academic Journal of Procurement and Supply Chain Management |*, *3*(1), 18–43. http://www.iajournals.org/articles/iajpscm_v3_i1_18_43.pdf

Johnson, K. W., Torres Soto, J., Glicksberg, B. S., Shameer, K., Miotto, R., Ali, M., Ashley, E., & Dudley, J. T. (2018). Artificial Intelligence in Cardiology. *Journal of the American College of Cardiology*, *71*(23), 2668–2679. https://doi.org/10.1016/J.JACC.2018.03.521

Kosasih, E. E., & Brintrup, A. (2021). *Reinforcement Learning Provides a Flexible Approach for Realistic Supply Chain Safety Stock Optimisation*.

Kumari, K. C., Wijayanayake, A., Niwunhella, H., Kumari, A. G. K. C., Wijayanayake, A. N., & Niwunhella, D. H. H. (2021). Lateral Transhipment Inventory Models: A Systematic Literature Review of Models and Solution Approaches. In *Business Management* (Vol. 54). https://www.researchgate.net/publication/363861874

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/NATURE14539

Leluc, R., Kadoche, E., Bertoncello, A., Onetech, T., & Gourvénec, S. (2023). *MARLIM: Multi-Agent Reinforcement Learning for Inventory Management*.

Li, F., Jörg, F., Li, X., & Feenstra, T. (2022). A Promising Approach to Optimizing Sequential Treatment Decisions for Depression: Markov Decision Process. *PharmacoEconomics*, *40*(11), 1015–1032. https://doi.org/10.1007/s40273-022-01185-z

Lieberman, M. B., & Demeester, L. (1999). Inventory Reduction and Productivity Growth: Linkages in the Japanese Automotive Industry. *Https://Doi.Org/10.1287/Mnsc.45.4.466*, *45*(4), 466–485. https://doi.org/10.1287/MNSC.45.4.466

Littman, M. L. (1994). *Markov games as a framework for multi-agent reinforcement learning*.

Lu, S. C. Y. (1990). Machine learning approaches to knowledge synthesis and integration tasks for advanced engineering automation. *Computers in Industry*, *15*(1–2), 105–120. https://doi.org/10.1016/0166-3615(90)90088-7

Lwiki, T., Ojera, P. B., Mugenda, N. G., & Wachira, V. K. (2013). *The Impact of Inventory Management Practices on Financial Performance of Sugar Manufacturing Firms in Kenya*. https://www.researchgate.net/publication/353904000

Madeka, D., Torkkola, K., Eisenach, C., Luo, A., Foster, D. P., & Kakade, S. M. (2022). *Deep Inventory Management*. http://arxiv.org/abs/2210.03137

Michalski, G. (2013). Value-Based Inventory Management. *Romanian Journal of Economic Forecasting*, *9*(1), 82–90. https://doi.org/10.2139/ssrn.1081276

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (2013). *Machine Learning: An Artificial Intelligence Approach - Google Livros*.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning*.

Munyaka, J. B., & Yadavalli, V. S. S. (2022). Inventory management concepts and implementations: a systematic review. *South African Journal of Industrial Engineering*, *33*(2), 15–36. https://doi.org/10.7166/33-2-2527

Osman, H., & Demirli, K. (2012). Integrated safety stock optimization for multiple sourced stockpoints facing variable demand and lead time. *International Journal of Production Economics*, *135*(1), 299–307. https://doi.org/10.1016/j.ijpe.2011.08.004

Othman, N. N. (2014). *A study on inventory management at a manufacturing company*.

Pandya, B., & Thakkar, H. (2016). *A Review on Inventory Management Control Techniques: ABC-XYZ Analysis*. www.restpublisher.com/journals/jemm

Pereira Coutinho, C. (2011). *Metodologia de Investigação em Ciências Sociais e Humanas - Clara Pereira Coutinho - Google Livros*.

Popescu, D., El-Khatib, M., El-Khatib, H., & Ichim, L. (2022). New Trends in Melanoma Detection Using Neural Networks: A Systematic Review. In *Sensors* (Vol. 22, Issue 2). MDPI. https://doi.org/10.3390/s22020496

Priniotakis, G., & Argyropoulos, P. (2018). *Inventory management concepts and techniques*. https://doi.org/10.1088/1757-899X/459/1/012060

Puterman, M. L. (2014). *Markov Decision Processes: Discrete Stochastic Dynamic Programming - Martin L. Puterman - Google Livros*.

Rushton, A., Croucher, P., & Baker, P. (2022). *The Handbook of Logistics and Distribution Management: Understanding the ... - Alan Rushton, Phil Croucher, Peter Baker - Google Livros*.

Rynkiewicz, J. (2019). Asymptotic statistics for multilayer perceptron with ReLU hidden units. *Neurocomputing*, *342*, 16–23. https://doi.org/10.1016/J.NEUCOM.2018.11.097

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Openai, O. K. (2017). *Proximal Policy Optimization Algorithms*.

Singh, D., & Verma, A. (2018). Inventory Management in Supply Chain. *Materials Today: Proceedings*, *5*(2), 3867–3872. https://doi.org/10.1016/J.MATPR.2017.11.641

Singh, S. R., & Kumar, T. (2011). Inventory Optimization in Efficient Supply Chain Management. *International Journal of Computer Applications in Engineering Sciences*. https://www.researchgate.net/publication/267368317

Sutton, R. S., & Barto, A. G. (2014). *Reinforcement Learning: An Introduction Second edition, in progress*.

Van Otterlo, M., & Wiering, M. (2012). Reinforcement learning and markov decision processes. *Adaptation, Learning, and Optimization*, *12*, 3–42. https://doi.org/10.1007/978-3-642-27645-3_1/COVER

Vrat, P. (2014). *Basic Concepts in Inventory Management*. 21–36. https://doi.org/10.1007/978-81-322-1970-5_2

Wuest, T., Weimer, D., Irgens, C., & Thoben, K. D. (2016). Machine learning in manufacturing: Advantages, challenges, and applications. *Production and Manufacturing Research*, *4*(1), 23–45. https://doi.org/10.1080/21693277.2016.1192517

Yin, H., Jiang, Q., Ruan, J., & Zhang, C. (2022). *IEEE Xplore Full-Text PDF:* https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10021568&tag=1

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding Deep Learning (Still) Requires Rethinking Generalization. *COMMUNICATIONS OF THE ACM*, *64*(3). https://doi.org/10.1145/3446776

Zhou, Z. H. (2021). Machine Learning. *Machine Learning*, 1–458. https://doi.org/10.1007/978-981-15-1967-3/COVER

Zietsman, H. J., & van Vuuren, J. H. (2022). *A generic framework for decision support in retail inventory management*. http://arxiv.org/abs/2207.13923

# APÊNDICE A

```
# Biblioteca
import pandas as pd

# Carregar ficheiros
necessidades_dia_seguinte_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/
necessidades_lojas.xlsx')
stock_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/stock_dia_seguinte.xlsx')
saidas_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/saidas.xlsx')

# Criar um dicionário para armazenar a quantidade de saída por id_artigo
saida_dict = saidas_df.set_index('id_artigo')['qt'].to_dict()

# Função para atualizar o stock
def atualizar_stock(row):
    id_artigo = row['id_artigo']
    if id_artigo in saida_dict:
        qt_saida = saida_dict[id_artigo]
        if row['qt'] >= qt_saida:
            row['qt'] -= qt_saida
            saida_dict[id_artigo] = 0
        else:
            saida_dict[id_artigo] -= row['qt']
            row['qt'] = 0
    return row

# Atualizar o stock
stock_df = stock_df.apply(atualizar_stock, axis=1)

# Adicionar as saídas que não foram atendidas ao stock_df
saida_nao_atendida = []
for id_artigo, qt_saida in saida_dict.items():
    if qt_saida > 0:
        saida_nao_atendida.append({'id_artigo': id_artigo, 'qt': -qt_saida, 'posição': None})

# Concatenar as saídas não atendidas ao stock_df
if saida_nao_atendida:
    stock_df = pd.concat([stock_df, pd.DataFrame(saida_nao_atendida)], ignore_index=True)

# Guardar resultados em xlsx
stock_df.to_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/stock_atualizado.xlsx', index=False)

# Carregar ficheiro
novo_stock_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/stock_atualizado.xlsx')

# Agregar o stock novo por id_artigo (somando as quantidades por posição)
stock_agregado_df = novo_stock_df.groupby('id_artigo')['qt'].sum().reset_index()
stock_agregado_df.rename(columns={'qt': 'stock posição'}, inplace=True)

# Unir stock agregado às necessidades do dia seguinte pelo id_artigo
df = pd.merge(necessidades_dia_seguinte_df, stock_agregado_df, on=['id_artigo'], how='left')

# Filtrar os artigos com necessidades de reposição
filtro = df['necessidades lojas dia seguinte'] >= df['stock posição']
artigos_filtrados = df.loc[filtro].copy()

# Cálculo das necessidades a repor apenas para os artigos filtrados
```

```
artigos_filtrados.loc[:, 'necessidades repor'] = artigos_filtrados['necessidades lojas dia seguinte'] – artigos
_filtrados['stock posição']

# Exportar para xlsx
artigos_filtrados.to_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/necessidades_reposição.xlsx',
index=False)
```

# APÊNDICE B

```
# Bibliotecas
import pandas as pd
import math

# Carregar ficheiros
artigos_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/artigos.xlsx')
tipo_posicoes_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/tipo_posições .xlsx')

# Função para calcular a capacidade máxima ajustada
def calcular_capacidade_ajustada(artigo, tipo_posicao, volume_original):

    # Calcular o volume do novo tipo de posição
    volume_tipo_posicao = tipo_posicao['altura'] * tipo_posicao['largura'] * tipo_posicao ['profundidade']

    # Calcular a proporção de volumes entre o novo tipo e o tipo original
    proporcao_volumes = volume_tipo_posicao / volume_original

    # Capacidade máxima ajustada (arredondamento para baixo)
    capacidade_ajustada = math.floor(artigo['Cap. máxima'] * proporcao_volumes)
    return capacidade_ajustada

# Lista para armazenar os resultados
resultados = []

# Iterar sobre cada artigo
for _, artigo in artigos_df.iterrows():

    # Obter o tipo original do artigo e filtrar no posicoes_df para calcular o volume original
    tipo_original_artigo = artigo['tipo']
    tipo_original_posicao = tipo_posicoes_df[tipo_posicoes_df['tipo'] == tipo_original_artigo].iloc[0]

    # Calcular o volume original do tipo do artigo
    volume_original = tipo_original_posicao['altura'] * tipo_original_posicao['largura'] * tipo_original_
posicao['profundidade']

    # Iterar sobre todos os tipos de posições disponíveis
    for _, novo_tipo_posicao in tipo_posicoes_df.iterrows():
        # Verificar se o artigo cabe no novo tipo de posição (considerando todas as orientações possíveis)
        if (
            (novo_tipo_posicao['altura'] >= artigo['altura_artigo'] and
             novo_tipo_posicao['largura'] >= artigo['largura_artigo'] and
             novo_tipo_posicao['profundidade'] >= artigo['profundidade_artigo']) or

            (novo_tipo_posicao['altura'] >= artigo['altura_artigo'] and
             novo_tipo_posicao['largura'] >= artigo['profundidade_artigo'] and
             novo_tipo_posicao['profundidade'] >= artigo['largura_artigo']) or

            (novo_tipo_posicao['altura'] >= artigo['largura_artigo'] and
             novo_tipo_posicao['largura'] >= artigo['altura_artigo'] and
             novo_tipo_posicao['profundidade'] >= artigo['profundidade_artigo']) or

            (novo_tipo_posicao['altura'] >= artigo['largura_artigo'] and
             novo_tipo_posicao['largura'] >= artigo['profundidade_artigo'] and
             novo_tipo_posicao['profundidade'] >= artigo['altura_artigo']) or

            (novo_tipo_posicao['altura'] >= artigo['profundidade_artigo'] and
             novo_tipo_posicao['largura'] >= artigo['altura_artigo'] and
             novo_tipo_posicao['profundidade'] >= artigo['largura_artigo']) or
```

```
        (novo_tipo_posicao['altura'] >= artigo['profundidade_artigo'] and
         novo_tipo_posicao['largura'] >= artigo['largura_artigo'] and
         novo_tipo_posicao['profundidade'] >= artigo['altura_artigo'])
    ):

        # Se for o mesmo tipo, a capacidade ajustada é a original
        if novo_tipo_posicao['tipo'] == artigo['tipo']:
            capacidade_ajustada = artigo['Cap. máxima']
        else:
            # Caso contrário, calcular a capacidade máxima ajustada
            capacidade_ajustada = calcular_capacidade_ajustada(artigo, novo_tipo_posicao, volume_original)

        # Adicionar o resultado à lista se a capacidade ajustada for maior que 0
        if capacidade_ajustada > 0:
            resultados.append({
                'id_artigo': artigo['id_artigo'],
                'tipo': novo_tipo_posicao['tipo'],
                'Cap. máxima ajustada': capacidade_ajustada
            })

# Converter os resultados em df e guardar em xlsx
resultados_df = pd.DataFrame(resultados)
resultados_df.to_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/Cap_máx_ajust_artigo_tipo.xlsx',
index=False)
```

# APÊNDICE C

```
# Biblioteca
import pandas as pd

# Carregar ficheiros
necessidades_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/necessidades_reposição.
xlsx')
artigos_imp_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/lista_artigos_imp.xlsx')
margem_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/margem_bruta.xlsx')

# Unir os dados das necessidades de reposição e margem com base no id_artigo
df = pd.merge(necessidades_df, margem_df, on='id_artigo', how='left')

# Unir com os artigos importantes, deixando os artigos que não são importantes com NaN
df = pd.merge(df, artigos_imp_df[['id_artigo']], on='id_artigo', how='left', indicator=True)

# Substituir os valores NaN na coluna 'margem' por 0, para os artigos sem margem
df['margem'].fillna(0, inplace=True)

# Definir o parâmetro da importância (ajustável)
valor_importancia = 100

# Definir a importância: Se o artigo estiver na lista de importantes, atribuir o valor de importância, caso
contrário é zero
df['importancia'] = df['_merge'].apply(lambda x: valor_importancia if x == 'both' else 0)

# Cálculo da prioridade
df['prioridade'] = (df['margem'] * df['necessidades repor']) + (df['importancia'] * df['necessidades repor'])

# Ordenar os artigos pela prioridade
df_sorted = df.sort_values(by='prioridade', ascending=False)

# Exportar a lista de prioridades para xlsx
df_sorted.to_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/lista_prioridades.xlsx', index=False)
```

# APÊNDICE D

| id_artigo | tipo | capacidade_ajustada | Área (cm²) |
|---|---|---|---|
| X0245 | A1 | 25 | 400 |
| X0245 | A9 | 87 | 700 |
| X0245 | A6 | 50 | 800 |
| X0245 | A2 | 110 | 1200 |
| X0245 | A3 | 320 | 2400 |
| X0245 | A4 | 880 | 9600 |
| X0245 | PALETE | 6000 | 9600 |
| X3065 | A1 | 144 | 400 |
| X3065 | A9 | 504 | 700 |
| X3065 | A6 | 288 | 800 |
| X3065 | A2 | 633 | 1200 |
| X3065 | A3 | 1843 | 2400 |
| X3065 | A4 | 5068 | 9600 |
| X3065 | PALETE | 34560 | 9600 |
| X3049 | A1 | 16 | 400 |
| X3049 | A9 | 57 | 700 |
| X3049 | A6 | 32 | 800 |
| X3049 | A2 | 72 | 1200 |
| X3049 | A3 | 209 | 2400 |
| X3049 | A4 | 576 | 9600 |
| X3049 | PALETE | 3927 | 9600 |
| X3039 | A1 | 16 | 400 |
| X3039 | A9 | 57 | 700 |
| X3039 | A6 | 32 | 800 |
| X3039 | A2 | 72 | 1200 |
| X3039 | A3 | 209 | 2400 |
| X3039 | A4 | 1152 | 9600 |
| X3039 | PALETE | 3927 | 9600 |
| X9175 | A1 | 17 | 400 |
| X9175 | A9 | 59 | 700 |
| X9175 | A6 | 34 | 800 |
| X9175 | A2 | 75 | 1200 |
| X9175 | A3 | 218 | 2400 |
| X9175 | A4 | 600 | 9600 |
| X9175 | PALETE | 4090 | 9600 |
| X0026 | A1 | 300 | 400 |
| X0026 | A9 | 1050 | 700 |
| X0026 | A6 | 600 | 800 |
| X0026 | A2 | 1320 | 1200 |
| X0026 | A3 | 3840 | 2400 |

| X0026 | A4 | 10560 | 9600 |
|---|---|---|---|
| X0026 | PALETE | 72000 | 9600 |
| X9609 | A1 | 81 | 400 |
| X9609 | A9 | 286 | 700 |
| X9609 | A6 | 163 | 800 |
| X9609 | A2 | 360 | 1200 |
| X9609 | A3 | 1047 | 2400 |
| X9609 | A4 | 2880 | 9600 |
| X9609 | PALETE | 19636 | 9600 |
| X7649 | A1 | 60 | 400 |
| X7649 | A9 | 210 | 700 |
| X7649 | A6 | 120 | 800 |
| X7649 | A2 | 528 | 1200 |
| X7649 | A3 | 768 | 2400 |
| X7649 | A4 | 2112 | 9600 |
| X7649 | PALETE | 14400 | 9600 |
| X2735 | A1 | 24 | 400 |
| X2735 | A9 | 84 | 700 |
| X2735 | A6 | 48 | 800 |
| X2735 | A2 | 105 | 1200 |
| X2735 | A3 | 307 | 2400 |
| X2735 | A4 | 844 | 9600 |
| X2735 | PALETE | 5760 | 9600 |
| X1128 | A4 | 33 | 9600 |
| X1128 | PALETE | 300 | 9600 |
| X6189 | A1 | 60 | 400 |
| X6189 | A9 | 210 | 700 |
| X6189 | A6 | 120 | 800 |
| X6189 | A2 | 528 | 1200 |
| X6189 | A3 | 768 | 2400 |
| X6189 | A4 | 2112 | 9600 |
| X6189 | PALETE | 14400 | 9600 |
| X4652 | A1 | 10 | 400 |
| X4652 | A9 | 38 | 700 |
| X4652 | A6 | 21 | 800 |
| X4652 | A2 | 48 | 1200 |
| X4652 | A3 | 139 | 2400 |
| X4652 | A4 | 384 | 9600 |
| X4652 | PALETE | 2618 | 9600 |
| X0052 | A1 | 95 | 400 |
| X0052 | A9 | 334 | 700 |
| X0052 | A6 | 190 | 800 |
| X0052 | A2 | 420 | 1200 |

| X0052 | A3 | 1221 | 2400 |
|---|---|---|---|
| X0052 | A4 | 3360 | 9600 |
| X0052 | PALETE | 22909 | 9600 |
| X5191 | A1 | 16 | 400 |
| X5191 | A9 | 59 | 700 |
| X5191 | A6 | 33 | 800 |
| X5191 | A2 | 74 | 1200 |
| X5191 | A3 | 216 | 2400 |
| X5191 | A4 | 594 | 9600 |
| X5191 | PALETE | 4050 | 9600 |
| X4804 | A1 | 4 | 400 |
| X4804 | A9 | 14 | 700 |
| X4804 | A6 | 8 | 800 |
| X4804 | A2 | 18 | 1200 |
| X4804 | A3 | 52 | 2400 |
| X4804 | A4 | 144 | 9600 |
| X4804 | PALETE | 981 | 9600 |
| X0846 | A1 | 156 | 400 |
| X0846 | A9 | 546 | 700 |
| X0846 | A6 | 312 | 800 |
| X0846 | A2 | 687 | 1200 |
| X0846 | A3 | 2000 | 2400 |
| X0846 | A4 | 5500 | 9600 |
| X0846 | PALETE | 37500 | 9600 |
| X4373 | A1 | 8 | 400 |
| X4373 | A9 | 29 | 700 |
| X4373 | A6 | 17 | 800 |
| X4373 | A2 | 37 | 1200 |
| X4373 | A3 | 108 | 2400 |
| X4373 | A4 | 299 | 9600 |
| X4373 | PALETE | 2040 | 9600 |
| X3422 | A1 | 8 | 400 |
| X3422 | A9 | 30 | 700 |
| X3422 | A6 | 17 | 800 |
| X3422 | A2 | 38 | 1200 |
| X3422 | A3 | 111 | 2400 |
| X3422 | A4 | 306 | 9600 |
| X3422 | PALETE | 2086 | 9600 |
| X4802 | A9 | 18 | 700 |
| X4802 | A2 | 20 | 1200 |
| X4802 | A3 | 60 | 2400 |
| X4802 | A4 | 165 | 9600 |
| X4802 | PALETE | 240 | 9600 |

| | | | |
|---|---|---|---|
| X2912 | A1 | 6 | 400 |
| X2912 | A9 | 23 | 700 |
| X2912 | A6 | 13 | 800 |
| X2912 | A2 | 30 | 1200 |
| X2912 | A3 | 87 | 2400 |
| X2912 | A4 | 240 | 9600 |
| X2912 | PALETE | 1636 | 9600 |
| X2949 | A1 | 3 | 400 |
| X2949 | A9 | 11 | 700 |
| X2949 | A6 | 6 | 800 |
| X2949 | A2 | 15 | 1200 |
| X2949 | A3 | 43 | 2400 |
| X2949 | A4 | 120 | 9600 |
| X2949 | PALETE | 818 | 9600 |
| X0102 | A9 | 2 | 700 |
| X0102 | A6 | 1 | 800 |
| X0102 | A2 | 2 | 1200 |
| X0102 | A3 | 7 | 2400 |
| X0102 | A4 | 21 | 9600 |
| X0102 | PALETE | 144 | 9600 |
| X4199 | A1 | 37 | 400 |
| X4199 | A9 | 131 | 700 |
| X4199 | A6 | 75 | 800 |
| X4199 | A2 | 165 | 1200 |
| X4199 | A3 | 480 | 2400 |
| X4199 | A4 | 1320 | 9600 |
| X4199 | PALETE | 9000 | 9600 |
| X3865 | A1 | 10 | 400 |
| X3865 | A9 | 38 | 700 |
| X3865 | A6 | 21 | 800 |
| X3865 | A2 | 48 | 1200 |
| X3865 | A3 | 139 | 2400 |
| X3865 | A4 | 384 | 9600 |
| X3865 | PALETE | 2618 | 9600 |

# APÊNDICE E

```
# Bibliotecas
import pandas as pd
import re

# Carregar ficheiros
prioridades_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/lista_prioridades.xlsx')
posicoes_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/posições.xlsx')
stock_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/stock_atualizado.xlsx')
tipo_posicoes_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/tipo_posições.xlsx')
capacidades_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/Cap_máx_ajust_artigo_
tipo.xlsx')

# Identificar as posições com stock
posicoes_com_stock = stock_df[['posição']]

# Filtrar as posições livres
posicoes_livres_df = posicoes_df[~posicoes_df['posição'].isin(posicoes_com_stock['posição'])].copy()

# Calcular a área para cada tipo de posição
tipo_posicoes_df['área'] = tipo_posicoes_df['profundidade'] * tipo_posicoes_df['largura']

# Unir as dimensões ao dataframe de capacidades
df_final = capacidades_df.merge(tipo_posicoes_df[['tipo', 'área']], on='tipo', how='left')

# Ordenar o dataframe final conforme a área e capacidade máxima ajustada
df_final_sorted = df_final.sort_values(by=['área', 'Cap. máxima ajustada'], ascending=[True, True])

# Função para extrair as letras após o 'P' e antes do número
def extrair_letras(posicao):
    match = re.search(r'P([A-Z]+)', posicao)
    if match:
        return match.group(1)  # Retorna as letras encontradas
    else:
        return None

# Lista com as letras correspondentes à ordem dos corredores
ordem_letras = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Z',
'AA', 'AB', 'AC', 'AD']

# Função para encontrar a posição mais próxima no mesmo corredor
def encontrar_posicao_mais_proxima(ultima_letra, posicoes_livres_df):
    posicoes_mesma_letra = posicoes_livres_df[posicoes_livres_df['letras_posicao'] == ultima_letra]

    if not posicoes_mesma_letra.empty:
        return posicoes_mesma_letra.iloc[0]

    try:
        index_ultima_letra = ordem_letras.index(ultima_letra)
    except ValueError:
        return posicoes_livres_df.iloc[0]

    posicoes_livres_df = posicoes_livres_df.copy()

    posicoes_livres_df.loc[:, 'distancia'] = posicoes_livres_df['letras_posicao'].apply(
        lambda x: abs(ordem_letras.index(x) - index_ultima_letra) if x in ordem_letras else float('inf'))

    posicoes_ordenadas = posicoes_livres_df.sort_values('distancia')
    return posicoes_ordenadas.iloc[0]
```

```
# Aplicar a função extrair_letras ao dataframe de posições livres
posicoes_livres_df.loc[:, 'letras_posicao'] = posicoes_livres_df['posição'].apply(extrair_letras)

# Função para alocar necessidades de reposição
def alocar_necessidades(prioridades_df, stock_df, df_final_sorted, posicoes_livres_df):
    alocacoes = []

    for _, prioridade in prioridades_df.iterrows():
        id_artigo = prioridade['id_artigo']
        necessidades_repor = prioridade['necessidades repor']
        ultima_letra_posicao_stock = None  # Armazenar a letra do corredor da última posição com stock

        # Filtrar as posições com stock para o id_artigo atual
        posicoes_com_stock = stock_df[stock_df['id_artigo'] == id_artigo].copy()

        # Calcular capacidade Disponível e garantir que seja > 0
        posicoes_com_stock['Capacidade Disponível'] = posicoes_com_stock['Cap. máxima'] - posicoes_com_
stock['qt']
        posicoes_com_stock = posicoes_com_stock[posicoes_com_stock['Capacidade Disponível'] > 0]

        # Ordenar as posições com stock pela maior capacidade disponível
        posicoes_com_stock = posicoes_com_stock.sort_values(by='Capacidade Disponível', ascending=False)

        # Alocar às posições com stock
        for _, posicao in posicoes_com_stock.iterrows():
            if posicao['Capacidade Disponível'] >= necessidades_repor:
                # Alocar o máximo da cap. disponível em vez de apenas a necessidade a repor
                alocar = posicao['Capacidade Disponível']
                alocacoes.append({
                    'id_artigo': id_artigo,
                    'posição': posicao['posição'],
                    'quantidade alocada': alocar
                })
                ultima_letra_posicao_stock = extrair_letras(posicao['posição'])
                necessidades_repor = 0
                break
            else:
                alocar = posicao['Capacidade Disponível']
                alocacoes.append({
                    'id_artigo': id_artigo,
                    'posição': posicao['posição'],
                    'quantidade alocada': alocar
                })
                ultima_letra_posicao_stock = extrair_letras(posicao['posição'])
                necessidades_repor -= alocar

        # Se ainda restarem necessidades a repor, alocar em posições livres
        while necessidades_repor > 0:
            df_artigo = df_final_sorted[df_final_sorted['id_artigo'] == id_artigo]

            # Procurar tipos com capacidade maior ou igual às necessidades a repor
            tipos_adequados = df_artigo[df_artigo['Cap. máxima ajustada'] >= necessidades_repor].sort_values
(by='área')

            if not tipos_adequados.empty:
                alocou = False
                for _, tipo_adequado in tipos_adequados.iterrows():
                    tipo = tipo_adequado['tipo']
                    posicoes_adequadas = posicoes_livres_df[posicoes_livres_df['tipo'] == tipo]
```

```
            if not posicoes_adequadas.empty:
                if ultima_letra_posicao_stock:
                    posicao = encontrar_posicao_mais_proxima(ultima_letra_posicao_stock, posicoes_
adequadas)
                else:
                    posicao = posicoes_adequadas.iloc[0]

                capacidade_disponivel = tipo_adequado['Cap. máxima ajustada']
                alocar = capacidade_disponivel

                necessidades_repor -= alocar
                necessidades_repor = max(0, necessidades_repor)

                alocacoes.append({
                    'id_artigo': id_artigo,
                    'posição': posicao['posição'],
                    'quantidade alocada': alocar
                })

                # Remover a posição usada
                posicoes_livres_df = posicoes_livres_df[posicoes_livres_df['posição'] != posicao['posição']]
                alocou = True
                break
        if alocou:
            continue  # Volta ao início do while para verificar se ainda há necessidades a repor

    # 1ª tentativa: Quando não há posições disponíveis para o tipo original, procurar tipos com
capacidade maior que o ideal
    tipos_maior = df_artigo.sort_values(by= 'Cap. máxima ajustada', ascending=False)

    if not tipos_maior.empty:
        alocou = False
        for _, tipo_maior in tipos_maior.iterrows():
            tipo = tipo_maior['tipo']
            posicoes_adequadas = posicoes_livres_df[posicoes_livres_df['tipo'] == tipo]

            if not posicoes_adequadas.empty:
                posicao = posicoes_adequadas.iloc[0]
                capacidade_disponivel = tipo_maior['Cap. máxima ajustada']
                alocar = capacidade_disponivel

                necessidades_repor -= alocar
                necessidades_repor = max(0, necessidades_repor)

                alocacoes.append({
                    'id_artigo': id_artigo,
                    'posição': posicao['posição'],
                    'quantidade alocada': alocar
                })

                posicoes_livres_df = posicoes_livres_df[posicoes_livres_df['posição'] != posicao['posição']]
                alocou = True
                break
        if alocou:
            continue  # Volta ao início do while

    # 2ª tentativa: Se ainda não existirem posições para o tipo escolhido, procurar tipos com menor
capacidade ajustada
    tipos_menor = df_artigo[df_artigo['Cap. máxima ajustada'] > 0].sort_values(by='Cap. máxima
ajustada')
```

```
        if not tipos_menor.empty:
            alocou = False
            for _, tipo_menor in tipos_menor.iterrows():
                tipo = tipo_menor['tipo']
                posicoes_adequadas = posicoes_livres_df[posicoes_livres_df['tipo'] == tipo]

                if not posicoes_adequadas.empty:
                    posicao = posicoes_adequadas.iloc[0]
                    capacidade_disponivel = tipo_menor['Cap. máxima ajustada']
                    alocar = capacidade_disponivel

                    necessidades_repor -= alocar
                    necessidades_repor = max(0, necessidades_repor)

                    alocacoes.append({
                        'id_artigo': id_artigo,
                        'posição': posicao['posição'],
                        'quantidade alocada': alocar
                    })

                    posicoes_livres_df = posicoes_livres_df[posicoes_livres_df['posição'] != posicao['posição']]
                    alocou = True
                    break
            if alocou:
                continue  # Volta ao início do while

        # Se não encontrou nenhuma posição adequada, interrompe o loop
        print(f"Não há posições disponíveis para alocar o artigo {id_artigo} com necessidade a alocar em falta
de {necessidades_repor}.")
        break

    return pd.DataFrame(alocacoes)

# Executar a função
alocacoes_df = alocar_necessidades(prioridades_df, stock_df, df_final_sorted, posicoes_livres_df)

# Exportar as alocações para xlsx
alocacoes_df.to_excel('alocações.xlsx', index=False)
```

# APÊNDICE F

```
# Biblioteca
import pandas as pd

# Carregar ficheiros
stock_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/stock_atualizado.xlsx')
alocacoes_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/alocações.xlsx')
posicoes_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/posições.xlsx')
capacidades_df = pd.read_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/ Cap_máx_ajust_artigo_
tipo.xlsx')

# Garantir que coluna 'qt' no stock_df e 'quantidade alocada' no alocacoes_df sejam tratadas como
numéricas
stock_df['qt'] = pd.to_numeric(stock_df['qt'], errors='coerce').fillna(0)  # Preencher valores nulos com 0
alocacoes_df['quantidade alocada'] = pd.to_numeric(alocacoes_df['quantidade alocada'], errors='coerce').
fillna(0)

# Função para alocar as quantidades ao stock
def calcular_stock_dia_seguinte(stock_df, alocacoes_df, posicoes_df, capacidades_df):
    novas_linhas = []  # Lista para armazenar novas linhas

    for _, row in alocacoes_df.iterrows():
        id_artigo = row['id_artigo']
        posicao = row['posição']
        quantidade_alocada = row['quantidade alocada']

        # Verificar se já existe o artigo e posição no stock
        mask = (stock_df['id_artigo'] == id_artigo) & (stock_df['posição'] == posicao)

        if mask.any():
            # Se já existe, somar a quantidade alocada à quantidade atual em 'qt'
            stock_df.loc[mask, 'qt'] += quantidade_alocada
        else:
            # Procurar o tipo e capacidade máxima ajustada para novas entradas
            tipo_posicao = posicoes_df.loc[posicoes_df['posição'] == posicao, 'tipo'].values
            tipo_artigo = tipo_posicao[0] if len(tipo_posicao) > 0 else None

            cap_max_ajustada = capacidades_df.loc[
                (capacidades_df['id_artigo'] == id_artigo) & (capacidades_df['tipo'] == tipo_artigo),
                'Cap. máxima ajustada'
            ].values

            cap_max = cap_max_ajustada[0] if len(cap_max_ajustada) > 0 else None

            # Criar uma nova linha com os dados e quantidade alocada (somente na coluna qt)
            nova_linha = {
                'id_artigo': id_artigo,
                'posição': posicao,
                'qt': quantidade_alocada,  # Quantidade inicial = quantidade alocada
                'tipo': tipo_artigo,
                'Cap. máxima': cap_max
            }
            novas_linhas.append(nova_linha)

    # Adicionar as novas linhas ao stock_df usando concat
    if novas_linhas:
        stock_df = pd.concat([stock_df, pd.DataFrame(novas_linhas)], ignore_index=True)
    return stock_df
```

```
# Calcular o stock do dia seguinte
stock_dia_seguinte = calcular_stock_dia_seguinte(stock_df, alocacoes_df, posicoes_df, capacidades_df)

# Guardar o stock do dia seguinte em excel
stock_dia_seguinte.to_excel('C:/Users/Acer/OneDrive/Ambiente de Trabalho/stock_dia_seguinte.xlsx',
index=False)
```

página propositadamente em branco