UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS TRINDADE - CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS
ENGENHARIA DE PRODUÇÃO CIVIL

Leonardo Felipe Camargo

Implementação de uma heurística de cálculo de Planejamento Mestre de Produção considerando capacidade finita dentro de uma ferramenta comercial

Florianópolis
2022

Leonardo Felipe Camargo

Implementação de uma heurística de cálculo de Planejamento Mestre de Produção considerando capacidade finita dentro de uma ferramenta comercial

Trabalho de Conclusão do Curso de Graduação em Engenharia de Produção Civil do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Engenharia Civil.
Orientador: Prof. Guilherme Ernani Vieira, Dr.

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Camargo, Leonardo Felipe
Implementação de uma heurística de cálculo de Planejamento Mestre de Produção considerando capacidade finita dentro de uma ferramenta comercial / Leonardo Felipe Camargo ; orientador, Guilherme Ernani Vieira, 2022.
79 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Produção Civil, Florianópolis, 2022.

Inclui referências.

1. Engenharia de Produção Civil. 2. Planejamento Mestre de Produção. 3. Heurística. 4. Advanced Planning and Scheduling. 5. Opcenter. I. Vieira, Guilherme Ernani. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Produção Civil. III. Título.

Leonardo Felipe Camargo

Implementação de uma heurística de cálculo de Planejamento Mestre de Produção considerando capacidade finita dentro de uma ferramenta comercial

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro Civil" e aprovado em sua forma final pelo Curso Engenharia de Produção Civil

Florianópolis, 12 de dezembro de 2022.

Documento assinado digitalmente
Monica Maria Mendes Luna
Data: 19/12/2022 09:11:03-0300
CPF: ***.104.993-**
Verifique as assinaturas em https://v.ufsc.br

Prof.ª Mônica Maria Mendes Luna, Dr.
Coordenadora do Curso

Banca Examinadora:

Documento assinado digitalmente
Guilherme Ernani Vieira
Data: 16/12/2022 21:29:27-0300
CPF: ***.311.759-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Guilherme Ernani Vieira, Dr.
Orientador
Universidade Federal de Santa Catarina (UFSC)

Documento assinado digitalmente
Viviane Vasconcellos Ferreira Grubisic
Data: 16/12/2022 16:31:56-0300
CPF: ***.671.729-**
Verifique as assinaturas em https://v.ufsc.br

Prof.ª Viviane Vasconcellos Ferreira Grubisic, Dr.(a)
Avaliadora
Universidade Federal de Santa Catarina (UFSC)

Documento assinado digitalmente
RENATO CAETANO DA SILVA NETO
Data: 16/12/2022 16:05:38-0300
Verifique em https://verificador.iti.br

Eng.º Renato Caetano Neto da Silva
Avaliador
Lean Scheduling Brazil (LSB)

Este trabalho é dedicado à minha maravilhosa família.

AGRADECIMENTOS

Agradeço aos meus pais, Izabel e Adão, aos meus irmão Cleverson, Jeferson, Fabricio e Fernando. Agradeço também minha querida irmã Monique. Por todo o carinho e cuidado que sempre tiveram comigo.

Agradeço todos os meus amigos que, de alguma maneira, estiveram envolvidos tanto na graduação, quanto neste trabalho. Um agradecimento especial ao Rurik, meu grande parceiro de toda a graduação.

Aos meus amigos fora da universidade, Natan e João, por todo apoio e distração quando necessária, lembro de cada um de vocês e sempre levarei a amizade comigo. A todos que cruzaram minha vida e deixaram boas pegadas.

Agradeço a UFSC e em especial o professor Guilherme Ernani Vieira que muito bem me instruiu neste trabalho, foi um prazer trabalhar com uma pessoa que é referência para mim não só como professor, mas como alguém que compartilha dos mesmos interesses profissionais.

Agradeço imensamente a Lean Scheduling Brazil (LSB), minha segunda família que sempre acreditou nos meus sonhos e investiu esforço junto comigo neste trabalho e na capacitação pessoal. Serei eternamente grato por todos os momentos que tivemos e anseio pelos próximos anos de trabalho nesta área tão especial das indústrias.

Gostaria de agradecer a todos os envolvidos diretamente na minha vida e dizer que se hoje estou aqui, foi porque estive com pessoas maravilhosas durante todos esses anos.

RESUMO

O planejamento e controle de produção é uma tarefa essencial em qualquer indústria que possua um sistema produtivo. As indústrias se veem na necessidade de planejar o atendimento da sua demanda, mas realizar essa tarefa com assertividade pode envolver uma série de variáveis complexas. O Planejamento Mestre de Produção é um processo que vem como um aliado para gerar um plano interessante e auxiliar no processo de controle das variáveis complexas. Este processo pode envolver a consideração de capacidades finitas de recursos de produção no momento de construção do plano mestre de produção e este é um dos pontos mais complexos de controle nas atividades. Considerar os recursos com suas reais capacidades torna o planejamento complexo e dificilmente pode ser feito manualmente ou com ferramentas simples de planejamento. Assim os programas de planejamento e programação avançados entram como um ponto de apoio na solução destes casos, no entanto, o problema continua existindo, pois os programas avançados não conseguem atender todas as indústrias com as heurísticas nativas. Para garantir que ao menos uma ferramenta avançada consiga entregar uma possibilidade de atender e gerar um Planejamento Mestre de Produção executável, o objetivo deste trabalho é implementar uma heurística de construção do Planejamento Mestre de Produção considerando as capacidades finitas dentro de uma ferramenta avançada de planejamento. Dois métodos foram considerados para realizar o objetivo geral. Uma revisão da literatura, primeiro método, é feita para garantir que os conceitos sobre o tema sejam revisados e sua complexidade seja realmente entendida. O segundo método vem com a proposta de implementação e disponibilização através de uma pesquisa-ação, onde uma heurística já existente é colocada dentro do Opcenter, ferramenta avançada em questão. Após a implementação da heurística, o Opcenter passou a ter a possibilidade de considerar variáveis que não vem por padrão na ferramenta, garantindo que a flexibilização tenha sido possível e permite também, que novas implementações no código possam ser feitas para adaptar ainda mais os resultados para indústrias desejadas.

Palavras-chave: Planejamento Mestre de Produção. Heurística. Advanced Planning and Scheduling. Opcenter.

ABSTRACT

Production planning and control is an essential task in any industry that has a production system. Industries find themselves in the need to plan to meet their demand but carrying out this task assertively can involve a series of complex variables. Master Production Scheduling is a process that comes as an ally to generate an interesting plan and help in the process of controlling complex variables. This process may involve the consideration of finite capabilities of production resources when building the master production plan and this is one of the most complex control points in activities. Considering resources with their real rate makes the planning to be complex and can hardly be done manually or with simple planning tools. So, the advanced planning and scheduling systems come as a support point in the solution of these cases, however, the problem still exists, because the advanced systems may not serve all industries with native heuristics. To ensure that at least one advanced tool would deliver a possibility to meet and generate an executable Master Production Scheduling, the objective of this work is to implement an heuristic for the construction of the Master Production Scheduling considering the finite capacities within an advanced planning tool. Two methods were considered to accomplish the general objective. A literature review, the first method, is done to ensure that the concepts on the subject are reviewed and its complexity is really understood. The second method proposes implementation and availability through action research, where an existing heuristic is placed within Opcenter, the advanced tool in question. After implementing the heuristic, Opcenter now has the possibility of considering variables that are not included by default in the tool, ensuring that flexibility has been possible and allow that new implementations in the code can be made to further adapt the results to desired industries.
Keywords: Master Production Scheduling. Heuristic. Advanced Planning and Scheduling. Opcenter.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABEPRO – Associação Brasileira de Engenharia de Produção

APICS - *Association for Supply Chain Management*

APS – *Advanced Planning and Scheduling*

CRP – *Capacity Requirements Planning*

DRP – *Distribution Resources Planning*

ERP – *Enterprise Resource Planning*

MPS – *Master Production Scheduling*

PMP – Planejamento Mestre de Produção

PPCP – Planejamento, Programação e Controle de Produção

MRP – *Materials Requirements Planning*

OP – Ordem de Produção

RCCP – *Rough Cut Capacity Planning*

S&OP – *Sales and Operations Planning*

SFC – *Shop Floor Control*

TDF – *Table Definition File*

SUMÁRIO

1 INTRODUÇÃO

Neste primeiro capítulo é realizada uma contextualização sobre o tema e o problema de pesquisa. O objetivo geral do trabalho é justificado dentro de um tópico exclusivo que busca demonstrar a relevância da solução proposta ao final. Uma organização do trabalho é apresentada ao final da introdução para

1.1 ABORDAGEM DO TEMA

O planejamento das operações das indústrias sempre foi uma atividade importante para o sucesso das indústrias (CORRÊA *et al.*, 2017). O planejamento da produção é uma atividade que até o último século era realizado com o conhecimento e experiência dos envolvidos no processo. O uso de tecnologia e computadores era limitado para resolver as complexidades associadas ao planejamento (MAUERGAUZ, 2016). Embora, o processo de planejamento seja complexo, a busca pela excelência de manufatura é presente nas indústrias, pois precisam definir um plano estratégico de atendimento da demanda que guie toda a operação da indústria com base nos objetivos (COSTA, 2016).

Uma das maneiras de buscar a excelência nas tomadas de decisões, é planejar hoje, o que se deseja ter resultado dentro de médios e longos prazos. Segundo Corrêa et al. (2017), o tempo decorrido entre a ação tomada hoje e o resultado obtido a partir de um prazo maior, é conhecido como inércia da decisão. Em virtude da inércia inerente às decisões tomadas, torna-se necessário o planejamento de médio e a longo prazo. Isto pode fornecer, por exemplo, visão de atendimento de demanda, projeção de ruptura de estoque e consumo da capacidade de produção.

O processo de planejamento de uma indústria pode ser extremamente complexo dependendo do ambiente em que se encontra inserida, no entanto, com o avanço da tecnologia, é possível utilizar de ferramentas digitais para facilitar o planejamento e controlar melhor as complexidades envolvidas, como produtos, recursos e variáveis operacionais (SLACK *et al.*, 1996; GAITHER e FRAZIER, 2002; MAUERGAUZ, 2016).

Associando a necessidade de planejamento de processos, tempos e recursos para melhores decisões a médio prazo, existe uma etapa dentro da hierarquia do Planejamento, Programação e Controle de Produção (PPCP) conhecida como

Planejamento Mestre de Produção (PMP) ou *Master Production Scheduling* (MPS) – embora o termo esteja disponível em português, neste trabalho utilizar-se-á a sigla MPS. O MPS é responsável por receber os dados de, por exemplo, pedidos de clientes, previsões de vendas, estoques atuais e capacidade de produção e transformar essas informações em um plano de produção que pode conter os seguintes resultados (GAITHER e FRAZIER, 2002; TUBINO, 2017):

- Quantidade necessária a produzir para atender a demanda;
- Estoque projetado baseado na produção estimada;
- Momento de produção considerando *lead times* para atender a demanda;
- Momentos de recebimento de materiais considerando *lead times*.

O breve resumo das possibilidades de saídas de um PMP demonstra que existem informações importantes a serem analisadas pelas empresas para o objetivo de garantir o atendimento da demanda atual e futura. Uma observação muito importante que deve ser lembrada desde o princípio deste projeto é a qualidade dos dados que são utilizados para gerar a informação, pois, uma vez que dado ruim é colocado como *input*, é dado ruim que se reflete como *output*. E a qualidade de saída reflete o quão bem uma organização consegue utilizar os resultados para desempenhar no dia a dia (CORRÊA, 2017).

Por fim, com um processo de planejamento bem consolidado dentro das indústrias, é possível desenvolver habilidades competitivas interessantes para o desempenho diário, pois, as empresas que se planejam com qualidade certamente estão mais suscetíveis ao sucesso do que ao fracasso (VIEIRA; FAVARETTO, 2006).

No mercado de ferramentas digitais de PMP, existem marcas já consolidadas que trazem soluções nesse sentido, no entanto, dos produtos oferecidos atualmente, a totalidade deles envolve uma série de esforços de projetos e horas de consultoria. Além disso, a customização dessas ferramentas nem sempre é possível de se realizar ou, quando sim, está associada a custos de configuração e pode não atender o que é desejado por uma indústria específica (ZAGO, 2013; TARNOWSKI, 2022).

As ferramentas digitais para auxiliar os complexos processos de PPCP e consequentemente o de MPS são conhecidas como sistemas *Advanced Planning and Scheduling* (APS). Os sistemas APS são ferramentas computacionais configuradas com modelos matemáticos complexos para encontra a solução de problemas reais. Os

sistemas são conhecidos como avançados não só pelo modelo matemático, mas também pela configuração de restrições complexas e métodos dinâmicos de tomadas de decisão, como por exemplo, analisar um conjunto de cenários diferentes de planejamento (MAUERGAUZ, 2016).

1.2 JUSTIFICATIVA

Os processos de planejamento precisam ser considerados finitos para a maioria das indústrias pois uma visualização incorreta de atendimento de demanda pode comprometer as relações comerciais entre fornecido e fornecedor. Pode também guiar as operações da produção, que são feitas em cima do planejamento, para um caminho problemático sob o ponto de vista de financeiro, pois não se sabe com certeza qual é o melhor cenário para os objetivos da empresa.

Por mais que uma série de programas e soluções estejam disponíveis no mercado, nem todas atendem a necessidade das indústrias com uma boa aderência, por conta disso se faz necessário o desenvolvimento de heurísticas flexíveis para as empresas que desejarem padronizar o seu processo através da computação.

A ferramenta APS para implementação foi o Opcenter pois é a solução mais vendida e usada no mundo das indústrias. Mesmo sendo a mais conhecida, as heurísticas nativas podem não atender algumas indústria. Por este motivo a heurística passa a ser flexível dentro do Opcenter. Outro motivo pelo qual a implementação via código fonte é interessante é para deter os métodos e funções desenvolvidas sob o controle do desenvolvedor ou consultor, situação que não é verdade quando se trabalha com as heurísticas nativas do Opcenter, pois, por questões comerciais, os códigos fontes não são abertos ao público ou até seus representantes.

Buscando atender um nicho de mercado em que se pode reduzir custos associados e fazer a construção, com a possibilidade de flexibilização, do PMP, o objetivo geral deste trabalho é implementar uma heurística para a criação de MPS considerando recursos com capacidade finita que as indústrias consigam utilizá-la para gerar resultados mais facilmente com um certo poder de flexibilidade e assertividade, além de permitir a visualização de informações da organização no formato digital.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Descrito anteriormente, o objetivo geral deste trabalho é implementar uma heurística de construção do Planejamento Mestre de Produção que considere os recursos com capacidade finita e busque atender os níveis de estoque desejado dentro de uma ferramenta APS.

1.3.2 Objetivos Específicos

A fim de atender o objetivo geral, os objetivos específicos do trabalho são:

1. Revisar os conceitos de PPCP associados ao cálculo de Planejamento Mestre de Produção.
2. Demonstrar a importância do cálculo do MPS e sua participação na hierarquia do PPCP.
3. Implementar uma heurística de criação de MPS baseada em Vieira e Favaretto (2006) considerando capacidade finita.
4. Apresentar e comparar os resultados gerais da heurística implementada com as heurísticas nativas da ferramenta APS.

1.3.3 Delimitações da pesquisa

Para garantir o foco no desenvolvimento do trabalho, busca-se limitar os estudos e desenvolvimentos envolvidos dentro dos seguintes parâmetros:

- O estudo não compara outras heurísticas disponíveis na literatura.
- A heurística implementada não debate a maneira de realizar o relacionamento do MPS com o *Materials Requirements Planning*.
- As inteligências de códigos de programação por traz da construção do MPS não são aprofundadas dentro da literatura.
- O desenvolvimento da heurística não terá uma inteligência de geração de ordens de produção.
- A heurística não busca atender indústrias de todos os ramos ou indústrias específicas, apenas busca permitir a criação de um primeiro plano com capacidade finita.

1.3.4 Organização do trabalho

O presente trabalho está organizado em uma introdução sobre o tema e o problema a ser tratado, junto com a introdução, estão os objetivos específicos e o geral, além da limitação de escopo trabalhado.

Após a introdução, uma revisão da literatura é realizada buscando entender a complexidade associada ao processo de MPS, a importância nas indústrias e como o processo pode ser facilitado através de tecnologia e programação para sistemas de gestão da produção.

Na sequência da revisão da literatura, o capítulo de metodologia é apresentado com o fluxo de tarefas realizadas para implementar a heurística de MPS na ferramenta APS (Opcenter Planning).

Já no desenvolvimento é apresentada uma validação da heurística implementada de Vieira e Favaretto (2006), garantindo que o fluxo proposto foi realmente seguido através da programação desenvolvida. Em seguida, uma comparação entre as heurísticas nativas do Opcenter Planning *versus* a heurística implementada é realizada para analisar dois indicadores de performance: estoque finais a cada período e utilização da capacidade dos recursos de produção.

A conclusão vem por último, garantindo que os objetivos do trabalho foram atendidos e deixa uma visão sobre o tema e seus próximos passos.

2 REVISÃO DA LITERATURA

Nesta seção são apresentados os conceitos de PPCP com sua hierarquia para auxiliar o entendimento da complexidade associada à criação do MPS. Os conceitos de tecnologia, junto com procedimentos disponíveis na literatura e ferramentas disponíveis no mercado, também são apresentados para demonstrar como o planejamento de produção pode ser realizado com ferramentas auxiliares.

Importante definir que toda sigla utilizada como MPS remete ao mesmo sentido de PMP e vice-versa. O acrônimo utilizado para o Planejamento Mestre de Produção é o MPS.

2.1 O PLANEJAMENTO, PROGRAMAÇÃO E CONTROLE DA PRODUÇÃO E SEUS CONCEITOS

É comum que as indústrias sejam conhecidas como um sistema integrado de entrada, transformação e saída de produtos ou serviços, este é popularmente chamado de sistema produtivo ou de produção (TUBINO, 2017). Dado que o sistema de produção é responsável por transformar entradas - matéria prima, recursos, tecnologia, dinheiro e outros recursos - em saídas como produtos e/ou serviços, é necessário saber responder o que, quando, quanto, e como produzir e comprar para criar um plano de apoio para tomada de decisões que seja mais assertivo para cada empresa (GAITHER *et al.*, 2002; CORRÊA *et al.*, 2017).

Para que um plano de ações seja efetivamente cumprido dentro de uma indústria, torna-se essencial pensar em prazos para realizar cada tarefa necessária dentro do planejamento (GAITHER *et al.*, 2002; TUBINO, 2017). Com isso, institui-se um dos conceitos mais conhecidos dentro do planejamento de qualquer atividade, o horizonte de planejamento ou de programação. Embora o horizonte de planejamento seja similar ao horizonte de programação, o termo planejamento tende a compreender o longo e médio prazo, enquanto, o horizonte de programação visa compreender o horizonte de curto prazo (GAITHER *et al.*, 2002; CORRÊA *et al.*, 2017).

Para a definição do horizonte de planejamento, embora não exista um procedimento padrão, pois pode variar de acordo com a realidade de cada indústria, é possível aproximá-lo a partir da análise qualitativa do tempo, ou seja, o quanto as

informações do futuro agregam nas decisões do momento atual. Além disso, é necessário pensar sobre o custo associado à obtenção de informação nos horizontes de longo prazo, que aumenta na medida em que passa o tempo (CORRÊA *et al*, 2017).

Corrêa *et al.* (2017) define um conceito associado ao planejamento chamado de inércia da decisão, que retrata a diferença de tempo dentre a tomada de decisão e o momento em que os resultados passam a surtir efeitos. Ainda, explica que o conceito de planejamento é dado como um processo hierárquico que segue um fluxo de informações do lógico, para assim, sair das decisões de maior inércia para as de menor inércia.

Fica evidente que um sistema de produção necessita ter uma boa visão do futuro para gerar um bom planejamento de ações e esta visão é muitas vezes obtida por processos de previsões combinados entre informações qualitativas e quantitativas (CORRÊA *et al.*, 2017).

2.1.1 O PPCP e seu fluxo de informações hierárquico

Antes de entrar nos conceitos de hierarquia, procura-se entender um pouco mais sobre o PPCP e o que significa cada uma das partes que compõem o acrônimo (sigla).

O Planejamento da produção, primeiro P da sigla, está associado ao conceito definido anteriormente de inércia das decisões. Corrêa *et al.* (2017) propõe que se fosse possível realizar alterações nas configurações da fábrica, como aumentar a capacidade de produção, e ter os resultados no mesmo momento, não seria necessário se planejar. Sabe-se que isto não é verdade e que situações de entregas de materiais e disponibilidade de recursos levam tempo e precisam ser pensadas para se ter dentro do momento necessário.

Dentro do mesmo contexto, vale ressaltar que um processo eficiente de planejamento em uma indústria visa compreender como o cenário atual da organização pode auxiliar no processo de planejamento de maneira adequada. Com isso, uma empresa pode construir um plano que tem o objetivo de produzir para atender a demanda com recursos disponíveis no momento atual e no futuro considerando tempos de entrega de fornecedores e quaisquer outras variáveis envolvidas no processo (CORRÊA *et al.*, 2017). O processo de planejamento pode também ser parte estratégica da empresa, onde irá definir missões globais, planos específicos para áreas difundidas

da organização (TUBINO, 2017), como por exemplo, um plano mestre para atingir um faturamento esperado pela controladoria.

Um dos pontos essenciais no planejamento é o entendimento prático das capacidades disponíveis da fábrica, fornecendo uma visão da capacidade de produção a médio prazo (VOLLMANN *et al.*, 2005).

Um exemplo de um problema de planejamento para uma indústria pode ser o seguinte: para que a capacidade de uma área de produção seja o dobro da atual, quando é necessário adquirir máquinas e contratar novos recursos para que isso seja refletido na organização dentro do horizonte de um ano? (VIEIRA; FAVARETTO, 2006).

O próximo P do PPCP é a programação da produção que depende diretamente dos conceitos de planejamento demonstrados anteriormente. Tendo como base o plano mestre de produção, a programação encarrega-se de cumprir o plano dentro de um curto prazo (SLACK *et al.*, 1996). Nesse sentido, este nível operacional é responsável por definir quanto e quando comprar, fabricar ou montar cada item que satisfaça o plano mestre de produção (TUBINO, 2017).

O nível de detalhamento da programação da produção varia de acordo com o ambiente de cada indústria. Para uma indústria com alto volume de produção e com uma baixa variabilidade de produtos, talvez não seja tão importante se ater aos detalhes da programação fina no chão de fábrica – hora de entrada e saída de material, mas buscar o detalhamento por dia ou semana de entrada e saída de material (TUBINO, 2017).

A atividade de programação da produção é uma tarefa de operação complexa dentro das indústrias e programar a produção considerando o melhor uso dos recursos é um dos motivos da complexidade (SLACK *et al.*, 1996). Não somente os recursos causam esta grande complexidade, mas outras variáveis como o calendário fabril, tempos de *setup*, chegada de materiais, capacidade produtiva e outras variáveis que incrementam ainda mais a dificuldade no momento de programar a produção de maneira inteligente (VOLLMANN *et al.*, 2005; VIEIRA; FAVARETTO, 2006).

O controle da produção, as últimas duas letras do PPCP, visa garantir que tudo o que foi planejado e programado seja cumprido da forma mais fiel possível (CORRÊA *et al.*, 2017). É extremamente vital para as indústrias garantir que o nível de aderência ao planejamento seja alto, caso contrário, o que foi proposto durante todos os horizontes

mais longos não terá o resultado esperado, podendo comprometer estratégias competitivas, desempenho de chão de fábrica e outros cenários comuns dentro das indústrias como falta de qualidade na entrega e prazos não atendidos (VOLLMANN *et al.*, 2005).

O processo de controle de chão de fábrica está associado ao horizontes de curto e/ou curtíssimo prazo, pois trata-se do momento em que o produto será produzido ou já foi produzido. Processos de controle da produção associados a um horizonte de longo prazo geralmente estão interessados em garantir a aderência ao modelo planejado, e se necessário, modificar o plano de acordo com as novas necessidades (SLACK *et al.*, 1996). O controle da produção fornece informações importantes para os processos de planejamento e programação, pois, utilizando de um horizonte rolante de tempo, os planos propostos podem ser atualizados e uma nova carga de atividades pode ser acrescentada (VOLLMANN *et al.*, 2005).

Após entender todos os conceitos anteriores, é possível perceber que o PPCP que não é um processo por si só, mas, um conjunto de atividades que se conectam entre si e variam ao longo do horizonte de tempo. O PPCP costuma ser uma área focada na gestão da produção de diversas indústrias e possui variados processos alocados no seu dia a dia, as atividades do PPCP se comunicam através de uma hierarquia conforme demonstrada na Tabela 1. Os processos envolvidos geralmente são complexos dentro das indústrias, mas possuem ferramentas disponíveis no mercado para o auxílio das atividades (CORRÊA *et al.*, 2017).

Cada parte que compõe o PPCP descrita anteriormente se conecta através de uma hierarquia que se conecta ao longo dos horizontes de planejamento da produção. Para cada processo, seja planejamento, programação ou controle, existe uma atividade de planejamento associada dentro hierarquia e para cada necessidade de análise dos recursos, existe um processo de planejamento de capacidade de produção. A Tabela 1 demonstra hierarquia proposta do PPCP (CORRÊA *et al.*, 2017; TUBINO, 2017) junto com os processos de MPS.

Tabela 1 – Hierarquia do PPCP

Planejamento de Capacidade	Planejamento de Produção	Horizonte
	S&OP	Longo/ Médio Prazo
RCCP	MPS (PMP)	Médio/Curto Prazo
CRP	MRP	Médio/Curto Prazo
APS	*Scheduling*	Curto Prazo
	SFC	Curtíssimo Prazo

Fonte: Autor (2022).

2.1.2 Sales and Operations Planning

O Planejamento de Vendas e Operações (*S&OP*), também conhecido como uma parte do planejamento estratégico (PE) de uma empresa, é um processo de planejamento da produção, e como discutido anteriormente, leva em conta o horizonte de longo/médio prazo e a configuração atual da organização e tem como resultado um plano desagregado para cada setor de empresa (marketing, compras, vendas etc.), que visa atender objetivos definidos no plano estratégico da organização como um todo (CORRÊA, 2017). A APICS (2013) define o S&OP como um processo de desenvolvimento de um plano tático, que provê a habilidade para a alta gestão de estrategicamente dirigir o negócio atingindo vantagens competitivas de forma contínua, através da integração entre planos da organização e com a gestão da cadeia de suprimentos. O S&OP é o primeiro processo a ser realizado dentro do fluxo da hierarquia do PPCP e passa por revisões periódicas que são definidas de acordo com as necessidades de cada empresa (CORRÊA, 2017).

2.1.3 Master Production Scheduling

O processo de Planejamento Mestre de Produção (MPS) é responsável por elaborar um plano de produção de produto acabado ou *Stock Keeping Unit* (SKU) – do inglês, Unidade de Manutenção de Estoque – passando por todos os itens que possuem demanda, tanto para estoque, quanto para vendas, e varrendo todos os períodos necessários de planejamento (CORRÊA, 2017). O plano mestre de produção representa o que a organização deseja produzir expresso em quantidade, datas e configurações de

produtos (APICS, 2013). O MPS recebe dados do S&OP como entrada para, por exemplo, produzir uma gama de produtos que atenda o faturamento desejado da companhia e os transforma em um plano de produção de SKUs que é explodido dentro do MRP. É muito comum que o MRP rode as necessidades de materiais e retorne algum ajuste necessário para o MPS, ou seja, eles se comunicam antes de descer para o próximo nível de programação da produção (*Scheduling*) que será apresentado na sequência. O MPS, portanto, é o dirigente para todo o detalhamento de produção fina necessária para atender os objetivos da organização (VOLLMANN *et al*., 2005).

2.1.4 Materials Requirements Planning

O processo de Planejamento das Necessidades de Materiais (MRP) tem como objetivo gerar um plano viável e detalhado de produção de componentes para produtos acabados e de necessidades de compras de matéria prima que não são obtidos internamente. O plano é gerado diretamente a partir do MPS e explode suas necessidades uma a uma, considerando configurações de produção interna e *lead times* de fornecedores (CORRÊA, 2017).

2.1.5 Scheduling

Até o momento, todos os processos anteriores são essencialmente atividades de planejamento. O processo de *Scheduling*, ou seja, de programação do que foi planejado ocorre após o de MRP e MPS serem realizado (CORRÊA, 2017). É o processo em que decide o momento de início e fim de produção de cada ordem de produção (OP) que foi emitida com base nos planos anteriores (SLACK *et al.*, 1996).

2.1.6 Shop Floor Control

O módulo de Controle de Chão de Fábrica (SFC) é responsável por controlar a programação da produção no chão de fábrica, para que seja cumprida da maneira mais aderente ao planejamento e programação da produção possível. É a execução que faz a ligação entre o plano de produção e o que de fato está produzindo-se na organização (SLACK *et al.*, 1996). Desse modo, representa o momento de linha de frente na hora de produzir, sendo o processo responsável por fazer que tudo aconteça como deve acontecer, garantindo materiais, recursos e todo o equipamento necessário para iniciar uma ordem (APICS, 2013).

2.1.7 Rough Cut Capacity Planning

O modulo de Planejamento Grosseiro de Capacidade (RCCP) visa verificar a capacidade de recursos críticos na indústria, a fim de garantir que num nível de informações mais agregado, gargalos de produção sejam identificados já dentro do MPS (CORRÊA, 2017).

2.1.8 Capacity Requirements Planning

O Planejamento das Necessidades de Capacidade é o processo que garante que as necessidades de MRP geradas a partir dos planos do MPS estão dentro das capacidades disponíveis no chão de fábrica. Também, verifica estouros de capacidade (quando passa o limite disponível no período) ou ociosidades nos recursos de produção e sugere alterações de datas de produção e quantidades de produtos no nível do MPS (CORRÊA, 2017).

2.1.9 Advanced Planning and Scheduling

O termo *Advanced Planning and Scheduling* (APS) – no português, Sistemas de Planejamento e Programação Avançado – foi vinculado com o *Scheduling* e é um ambiente onde um conjunto de regras e heurísticas são implementadas para gerar uma sequência de ordens para chão de fábrica considerando a real capacidade de produção no nível de detalhamento de produto mais desagregado. Os sistemas APS são softwares bastante utilizados dentro do PPCP das indústrias em conjunto aos sistemas de *Enterprise Resources Planning* (ERP), que são sistemas de gestão de recursos da organização (TUBINO, 2017).

Demonstrado anteriormente na Tabela 1, é possível encontrar quatro níveis de planejamentos diferentes que se interconectam ao longo do planejamento e execução da produção de uma indústria. A seguir, a definição dos níveis explicitados (CORRÊA, 2017):

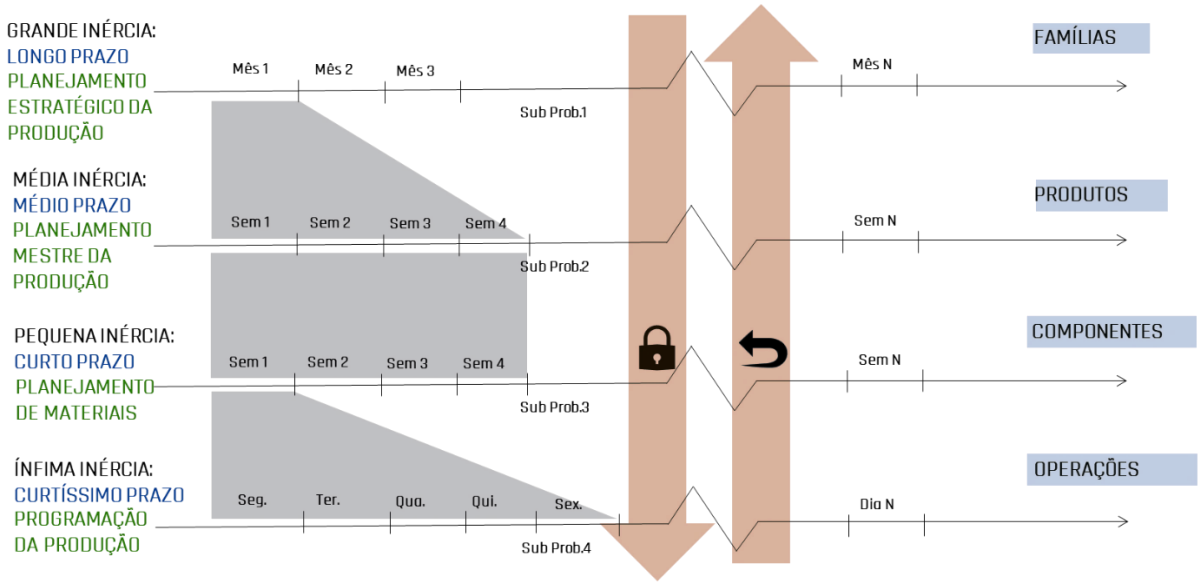
- **Longo prazo**: períodos de anos, este nível compreende o S&OP, é o responsável pelo planejamento de atividades que possuem grande inércia de decisão, ou seja, que levam tempo para que os resultados sejam obtidos.

Utiliza-se muito de métodos de previsão qualitativos e baseados em históricos de dados.

- **Médio prazo**: período de meses, onde é necessário planejar a produção de produtos acabados, validar se haverá gargalo, prever necessidades de materiais e trabalhar com níveis de planejamento mais agregados.
- **Curto prazo**: período de semanas, pretende congelar as atividades que foram previstas em períodos anteriores de planejamento, é a etapa de validação de materiais, capacidades e níveis de agregação mais detalhados que os anteriores.
- **Curtíssimo prazo**: período de dias, trabalha com as atividades do momento mais atual do chão de fábrica, evita as alterações de última hora que prejudicam a performance das operações.

Corrêa faz uma comparação entre os níveis hierárquicos do PPCP e os níveis de agregação dos produtos como é disposto na Figura 1. Cada processo da hierarquia do PPCP aparece do lado esquerdo da Figura 1 trazendo as inércias associadas a cada um deles e, do lado direito, traz o nível de agregação de informação de produtos para horizonte de planejamento. Ou seja, nos níveis de planejamento de longo prazo, os horizontes de tempo indicados para fins de processo são meses ou anos e buscam detalhar as informações de produção no nível de famílias de produtos.

Figura 1 – Níveis de agregação de produtos



Fonte: Adaptado de Corrêa (2017).

Os níveis de agregação de produtos variam entre:

- Famílias: São os produtos agregados em semelhanças de atributos, por exemplo: canetas.
- Produtos: É o produto, com suas características finais, por exemplo: caneta preta.
- Componentes: São os componentes necessários para a formação da caneta: ponta preta e corpo.
- Operações: Após todos os componentes e itens acabados (SKUs) serem planejados, o próximo passo é realizar a programação dos momentos de produção destes SKUs: Produzir a ponta preta primeiro, na sequência o corpo da caneta e montar ao final.

Na medida em que os tempos são agregados em meses ou anos, é difícil trabalhar com o alto detalhamento dos produtos, pois, não é essencial saber quanto de cada componente irá ser necessário produzir em determinado período, mas sim, o quanto de uma família de produto é preciso entregar em determinada data (CORRÊA, 2017).

Compreendida a importância do PPCP, junto com seus processos e hierarquia definida, é possível passar para o detalhamento do MPS e suas complexidades características.

2.2 O PLANEJAMENTO MESTRE DE PRODUÇÃO

Esta seção busca entender como o Plano Mestre de Produção é realizado dentro das organizações, qual a sua complexidade e as configurações considerando capacidade finita ou infinita de produção.

2.2.1 Planejamento Mestre de Produção e a sua importância nas indústrias

O planejamento mestre de produção é um equilíbrio entre a demanda de mercado e os recursos internos de uma companhia e tem o objetivo de gerar volumes adequados de produção de SKUs para a produção de curto prazo (CORRÊA, 2017).

Ainda, o PMP faz a conexão entre o planejamento estratégico de longo prazo e a operação do chão de fábrica. O plano mestre de produção realiza esta comunicação entre os dois elos realizando o desmembramento do planejamento de longo prazo em

um plano de produção mais detalhado, transformando as famílias de produtos em produtos acabados (TUBINO, 2017).

A 14ª (décima-quarta) versão do *The Association for Operations Management Dictionary* (2013) define o MPS como:

> Representa o que a empresa planeja produzir expressado em termos de configurações específicas, quantidades e datas. O Plano Mestre de Produção não é uma previsão de vendas que representa uma simples declaração da demanda. O Plano Mestre de Produção deve levar em conta a previsão, o plano de produção e outras considerações importantes, como o backlog, disponibilidade de material, disponibilidade de capacidade e gerenciamento de políticas e objetivos. (APICS, 2013, p. 106).

Como um plano mestre de produção é construído depende da filosofia adotada por cada companhia para realizar a gestão da produção. Os principais critérios selecionados, na criação do MPS, durante as últimas décadas estão diretamente ligados ao aumento de produtividade e, consequentemente, aumento de faturamento. Existem várias filosofias associadas no *modus operandi* das indústrias, destacam-se duas: a Teoria das Restrições (*Theory of Constraints, TOC*) e a *Just in Time* (*JIT*). De acordo com Viera e Favaretto (2006), o processo de criação de MPS que segue a teoria das restrições deve ser realizado sempre no nível de produto acabado e considerando recursos gargalos como capacidade de produção, para aí então, gerar um plano sobre os componentes dos produtos acabados.

O MPS exerce algumas funções importante para a performance e competitividade das indústrias. O Planejamento Mestre de Produção traz visões de como a demanda será atendida, qual a capacidade alocada devida a produção para atender a demanda, gestão de estoque a cada período de planejamento e integração das tomadas de decisões na hierarquia do PPCP de uma indústria, garantido uma aderência aos planos detalhados no S&OP e confiabilidade no próximo processo de MRP. Por outro lado, um plano de produção criado não garante nenhum sucesso, pois caso seja feito sem fundamentos, pode não trazer os resultados desejados e incorrer em outros riscos para a organização (CORRÊA, 2016).

Como a demanda é o que guia os resultados do MPS, seja ele para indústrias que trabalham sob o modelo MTS, MTO ou quaisquer outros combinados ou não, é importante defender a importância da gestão dela. O conceito de demanda, por si só, pode ser enquadrado como sendo a necessidade de um produto ou um componente intermediário, demandado por um terceiro ou por necessidade interna (PROUD, 2012).

A gestão da demanda é o momento em que a indústria toma conhecimento de todas as demandas por produtos e/ou serviços e prioriza cada uma delas dentro da execução da produção e uso de recursos (APICS). Acrescentando, a gestão da demanda envolve o esforço de cinco focos de uma indústria: previsão de demanda, promessa de prazos, priorização e alocação de produção, comunicação com o mercado e influência sobre o mercado (CORRÊA, 2017).

A gestão adequada da demanda é de interesse vital para o negócio de uma empresa, pois, acaba por direcionar o rumo de atividades em cima das necessidades de atendimento (TUBINO, 2017). Além disso, existem algumas razões pelas quais a demanda deve ser administrada:

- As empresas podem ter clientes que chegam a negociar quantidades e momentos apurados de entrega para atendê-los.
- A demanda de algumas empresas pode ser substancialmente alterada de acordo com marketing, promoções e alguns outros esforços sobre o consumidor.
- A empresa pode, por esforço pessoal, gerar um comportamento indutor de vendas dentro da área de vendas e representantes através de sistemas de bonificação, por exemplo.

Estes são alguns motivos que reforçam a necessidade de gerir a demanda e pode ser melhor realizado através de um bom dimensionamento do planejamento mestre de produção, o qual os dois estão fortemente ligados. Sem demanda, um MPS poderia perder o sentido de existência, mas, sem um MPS bem construído, o atendimento da demanda fica sob risco de alguns profissionais dentro da indústria que costumam trabalhar no conhecido *feeling*, ou seja, trabalham se baseando na própria experiência (CORRÊA *et al.*, 2017).

Ao final do fluxo de criação do MPS, tem-se um plano que formaliza todas as decisões tomadas quanto às necessidades de produtos para cada período de demanda analisada no processo. A partir deste plano é possível fazer a leitura de algumas informações (VOLLMANN, 2005; TUBINO, 2017; CORRÊA *et al.*, 2017):

- Projeção de estoque futura;
- Visibilidade de gargalos de produção;
- Necessidades de horas extras, subcontratação e turnos extras;
- Gestão da demanda, suprimentos e *lead times*;
- Recusa de pedidos que não podem ser entregues nas datas solicitadas;

- Disponibilidade de capacidade para aceitar pedidos urgentes;

2.2.2 A complexidade de construção do MPS

O desenvolvimento do MPS é uma tarefa complexa, especialmente quando consideramos que a capacidade real das indústrias é limitada. O cálculo do MPS disponível em literaturas clássicas não considera a capacidade finita de produção, ou seja, é como se tudo que fosse planejado, pudesse ser executado no momento ideal. É possível que em cenários específicos de algumas indústrias, com poucas restrições de capacidade, recursos de produção ociosos e demandas simples, o processo de criação de um plano mestre de produção possa ser uma tarefa fácil, porém, esse não é o cenário da grande maioria (VIEIRA; FAVARETTO, 2006).

O processo se torna complexo, pois, geralmente as indústrias se veem numa situação de alta utilização de recursos gargalos de produção, dificuldade de gestão de mão de obra capacitada e ferramental limitado. Nestes casos, por exemplo, quando um recurso de produção tem um problema de quebra de máquina, a situação pode ser crítica na tentativa corrigir os planos de produção (VIERA; FAVARETTO, 2006).

Além de considerar as capacidades fabris, é necessário olhar para a cadeia de suprimentos como um todo, em que várias empresas estão interconectadas, fornecendo e comprando materiais entre si (CORRÊA, 2017). Considerando que o cenário de muitas indústrias tem problemas de capacidade, esta situação acaba refletindo na execução da produção e consequentemente em prazos de entregas para alguns clientes e prazos de recebimento para fornecedores (VIERA; FAVARETTO, 2006).

Mais complexidades dentro do processo podem aparecer em outros formatos, como a flexibilidade de roteirização de recursos, ou seja, um produto pode ser produzido em diversos recursos em várias etapas de produção diferentes, então, algumas perguntas começam a surgir. Qual o melhor cenário de produção para a empresa? Qual o melhor planejamento para os recursos? Qual é o plano de produção que traz maior lucro? Os envolvidos dentro do processo de criação de um plano mestre de produção certamente vivem a complexidade de construir diversos de planos mestres de produção para a tomada de decisão (VIERA; FAVARETTO, 2006), no entanto, o MPS, dentro de algumas indústrias, pode levar tempo considerável para ser construído e por este motivo nem sempre é possível criar cenários e compará-los a fim de atingir um objetivo específico.

Para exemplificar, Slack *et al.* (1996), Vieira e Favaretto (2006) propõem uma visualização de cenários: mais de três milhões de combinações são possíveis para a ordenação de 10 ordens de produção em uma única linha de produção. A complexidade aumenta na medida em que novos parâmetros são acrescentados, como é o resultado do número intangível de combinações possíveis para 200 produtos diferentes em 40 linhas de produção.

Diante do exposto acima, portanto, a complexidade aumenta a cada parâmetro novo no processo. Os critérios que são utilizados na construção do MPS são ditados por cada indústria, tornando ainda mais difícil a padronização de um processo de criação de MPS que atenda todos os ramos possíveis. Nos objetivos de criar um processo mais robusto, a capacidade de produção das indústrias pode ser considerada, e é importante que seja, para o cálculo do MPS. Para efeitos de criação da MPS, os conceitos mais considerados para entrada de dado dentro do cálculo inicial são (VIERA; FAVARETTO, 2006):

1. Horizonte de planejamento: Geralmente varia entre semanas e meses;
2. Recursos e Produtos: Itens finais ou SKUs (Unidades de Manutenção de Estoque);
3. Necessidades grosseiras: Demanda do período, seja pedido em carteira ou previsão de demanda;
4. Volume terceirizado: Quantidade de um item que vai para um terceiro fabricar;
5. Tamanho do lote padrão: Quantidade padrão de abertura de produção de um item;
6. Tamanho de lote mínimo: Quantidade mínima para abrir uma ordem de produção para determinado item;
7. Estoque de segurança: Estoque mínimo desejado para evitar rupturas de estoque e garantir nível de serviço;
8. Estoque máximo: Estoque máximo permitido para cada produto;
9. Estoque inicial: Estoque inicial, para cada período, de cada produto;
10. Taxa de produção: Tempos de produção dos produtos nos recursos de produção;

11. Tempos de setup: Tempo de setup entre trocas de características de produtos nas máquinas de produção;
12. *Backlog*: O que não foi possível produzir nos últimos períodos, que deve ser produzido nos próximos;
13. Capacidade: Capacidade disponível para produção dos recursos;
14. Horas extras: O máximo de horas extras permitidas por período sobre a capacidade.

Para fins de saída de dado os parâmetros são:

1. Estoque inicial: A quantidade de um SKU disponível no estoque dentro de cada período planejado.
2. Estoque final: Estoque final de um produto em determinado período.
3. Necessidades Líquidas: O que deve, de fato, ser produzido no chão de fábrica no período.
4. MPS por recurso: Quantidade a ser produzida de um produto em um recurso dentro de um período.
5. MPS total nos recursos: Quantidade somada por recurso que irá produzir um produto.
6. Capacidade utilizada: Capacidade utilizada para aquele período em virtude do MPS.
7. Necessidades atendidas: Quanto das necessidades grosseiras foram atendidas.
8. Necessidades não atendidas: Quanto das necessidades grosseiras não foram atendidas.
9. Nível de serviço: Representa a porcentagem sobre o quanto era esperado atender das necessidades grosseiras e o que foi atendido de fato.

2.2.3 Diferenças entre cálculo de capacidade finita *versus* capacidade infinita

Para entender os conceitos de finito ou infinito das indústrias, é necessário entender como funciona o planejamento dos recursos. Como vem sendo explicado, o planejamento de produção é efetuado durante vários horizontes, detalhando ainda mais as informações na medida em que o tempo vai para o presente. Com a intenção de gerar um plano consistente de produção, é necessário determinar quanto tempo uma ordem da produção de determinado produto e volume associado vai levar para ser produzido em

um recurso. Para conciliar o volume e o tempo, três atividades são levadas em consideração (SLACK *et al.*, 1996):

- Planejamento do recurso – determinação do volume com o qual uma operação produtiva pode lidar. Slack *et al.* (2006) propõe o termo de carregamento, porém neste contexto cabe a adaptação para planejamento do recurso;
- Sequência – determinação da prioridade de tarefas a serem desempenhadas;
- Programação – decisão do tempo de início e fim para cada atividade.

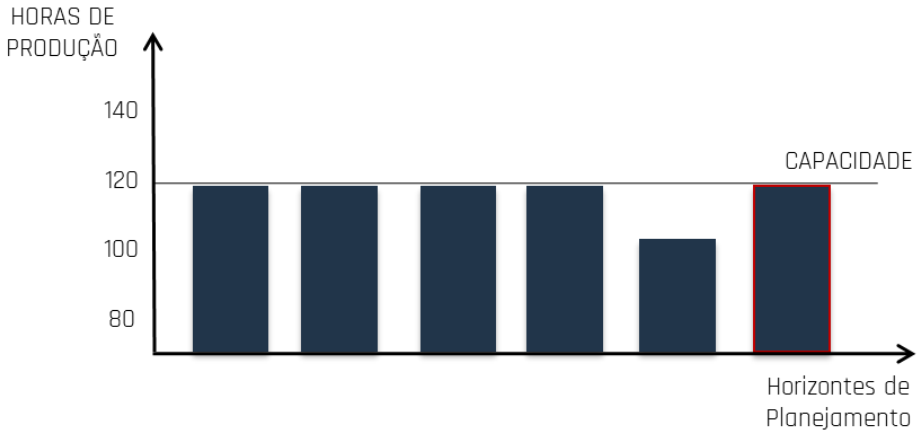
Para entender o comportamento finito ou infinito neste momento, apenas o aprofundamento do planejamento do recurso é necessário.

O planejamento do recurso é o quanto, na unidade de tempo, em volume de trabalho está alocado a um determinado recurso. Por exemplo, uma máquina está disponível por 168 horas durante uma semana, porém, não necessariamente pode receber um planejamento de produção de 168 horas, pois, não opera por todo esse tempo devido à calendários e afins. O tempo total disponível para operação é feito pela área de engenharia da organização e reflete o quanto ela pode produzir em volume de horas dentro de um período. Com isso, existem dois tipos de planejamento, os finitos e os infinitos (SLACK *et al.*, 1996).

2.2.3.1 Planejamento finito

O planejamento finito é uma abordagem que aloca trabalho para um recurso até um limite pré-estabelecido e o trabalho acima desta capacidade não é aceito. Por exemplo: Um forno que produz duas pizzas por hora, uma por vez, não pode ter um planejamento com três pizzas por hora. Em outras palavras, o volume de produção estimado deve obedecer ao quanto os recursos disponíveis na fábrica podem produzir (SLACK *et al.*, 1996). É possível visualizar o planejamento com a capacidade limite de um recurso através da Figura 2:

Figura 2 – Planejamento de recursos no modo finito

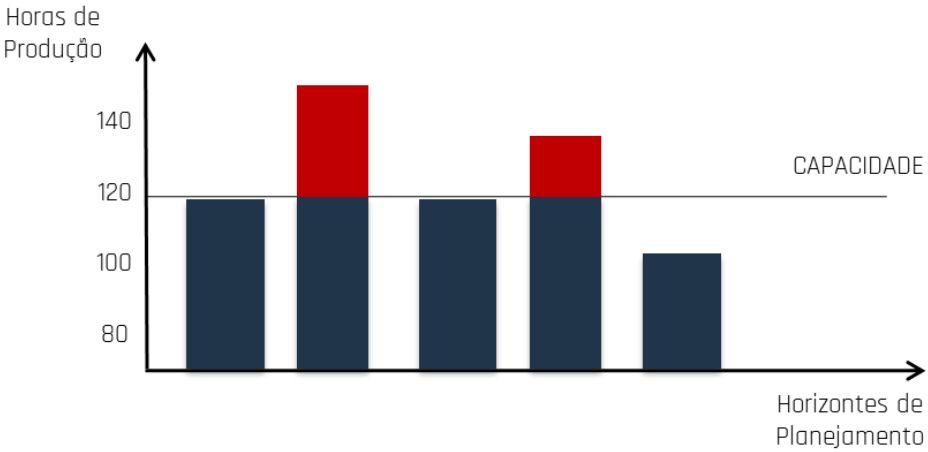


Fonte: Adaptado de Slack *et al.* (1996).

2.2.3.2 Planejamento infinito

O carregamento infinito, por sua vez, não limita a aceitação de alocação de trabalho, mas sim, busca corresponder ao que lhe foi planejado. Seguindo o exemplo do carregamento, o forno buscaria produzir 3 pizzas dentro de uma hora. Associar produção a um recurso para que este tente produzir o que lhe é esperado não necessariamente significa que será possível, e muitas vezes não é, mas sim que este será o objetivo do recurso. Ver Figura 3:

Figura 3 – Planejamento de recursos no modo infinito



Fonte: Adaptado de Slack *et al.* (1996).

Com os dois tipos de planejamento de recursos possíveis, um plano mestre de produção que considera capacidade limitada, guiará os fluxos de processos do PPCP a uma aderência maior no momento em que for necessário fazer a programação de produção, pois, as ordens de produção foram previamente analisadas e geradas em cima da capacidade real de produção.

Por outro lado, os planejamentos mestres de produções gerados com capacidades infinitas não são totalmente descartados. Existe interesse por parte das indústrias em realizar MPS com capacidade infinita para identificar possíveis gargalos de produção a longo prazo e realizar um plano de compras de novos recursos para corrigir esta situação.

2.3 HEURÍSTICAS E META-HEURÍSTICAS

Embora este trabalho adote uma proposta heurística como uma possível solução para a complexidade do planejamento mestre de produção, existem outros métodos disponíveis na literatura que podem ser aplicados para alcançar resultados ótimos ou aproximados do ótimo.

Para realizar o cálculo do MPS, existem variáveis associadas ao resultado que formam uma função geral, que é caracterizada como um problema de otimização combinatória do tipo *NP-Hard* (PEREIRA, 2011). Os métodos que são apresentados compreendem uma gama de possíveis artifícios para serem utilizados na busca da solução para problemas de otimização.

Dentro dos métodos de soluções de aproximação existem duas áreas:

1. Soluções ótimas;
2. Soluções de aproximação.

2.3.1 Soluções ótimas

Quando um problema é relativamente fácil de ser solucionado ou envolve um pequeno número de variáveis e o cenário é controlado os métodos de soluções ótimas são os mais indicados para encontrar um ponto ótimo (NORIN, 2016). Existem alguns métodos dentro das soluções ótimas que buscam por meio de diversos procedimentos definidos encontrar uma saída ótima de acordo com os dados fornecidos. São alguns deles (NORIN, 2016):

- *Graph Theory;*
- *Integer Programming;*
- *Dynamic Programming;*
- *Branch and Bound.*

O detalhamento exato destes métodos apresentados foge do escopo deste trabalho, pois não se enquadram nos métodos mais adequados para a resolução de problemas relacionados ao MPS (VIERA e FAVARETTO, 2006).

2.3.2 Soluções aproximadas

Dentro das soluções aproximadas os métodos se subdividem em dois grupos:

1. Heurísticas;
2. Meta-heurísticas.

2.3.3 Heurísticas

Métodos heurísticos são procedimentos que buscam a solução de um problema apoiando as decisões em critérios lógicos racionais, ou computacionais, para percorrer um caminho dentre vários possíveis sem o objetivo de encontrar a solução ótima ou percorrer todas as variações de soluções. Embora, e em geral, uma heurística não encontre uma solução ótima, a diferença entre o ponto ótimo e a saída da heurística é pequena e isto justifica o uso em virtude dos métodos de soluções otimizadas devido ao alto tempo computacional associado. Soluções ótimas costumam ter custos mais elevados em termos computacionais, do que soluções heurísticas ou meta-heurísticas (FUCHIGAMI, 2005). As heurísticas podem ser classificadas em dois métodos e funcionam como estão descritos a seguir:

- **Métodos construtivos**: A sequência de procedimentos é obtida diretamente da ordenação dos dados de entrada baseada em parâmetros pré-estabelecidos para um objetivo final, ou ainda, uma sequência de atividades é escolhida através das prioridades pertencentes a elas mesmas, sem precisar ordenar os dados iniciais. Por fim, ao combinar as duas situações pode-se ordenar parcialmente o conjunto de dados e as sequências de atividades (FUCHIGAMI, 2005). Estes, são os métodos mais simples dentro das heurísticas, que buscam atender um conjunto de atividades necessárias para a realização do procedimento final. Embora apresentem boas soluções como saída, caracterizam-se como as que menos se aproximam do ponto ótimo (NORIN, 2016). O método finaliza ao iterar todos os dados, ordenados ou não, através das sequências definidas (PEREIRA, 2011).

- **Métodos melhorativos**: O método se dá a partir de uma solução inicial, geralmente gerada por outra heurística iterativa, que busca melhorar o estado atual através de métodos de perturbações no conjunto de dados disponibilizado. Neste caso, para determinar uma solução melhorativa, busca-se definir um parâmetro melhorativo. Destacam-se os métodos de busca em vizinhança local, considerados métodos simples (NORIN, 2016).

Os métodos melhorativos, que podem ser aplicados dentro de uma solução disponibilizada por um método construtivo, delimitam uma linha para as meta-heurísticas, que são métodos também melhorativos e partem de uma solução inicial, porém, são mais complexos e possuem estratégicas que exploram apropriadamente os resultados disponíveis (FUCHIGAMI, 2005).

2.3.4 Meta-heurísticas

Meta-heurísticas são métodos aproximados com o objetivo de resolver problemas de otimização combinatória, geralmente utilizadas quando heurísticas clássicas e mais simples não conseguem obter um resultado satisfatório. Meta-heurísticas podem ser um combinado entre diversos conceitos de heurísticas clássicas, inteligência artificial, redes neurais e outros encontrados na natureza. Na maioria das vezes, os métodos meta-heurísticos trazem resultados mais adequados aos problemas aplicados (PEREIRA, 2011).

Os problemas do tipo NP-*Hard* (Tempo polinomial não determinístico) com frequência compreendem métodos meta-heurísticos como solução, o que se justifica a aplicação dentro dos conceitos de MPS demonstrados até o momento. Além disso, a maioria das propostas disponíveis na literatura deste método têm sido no âmbito de resolver problemas relacionados ao PPCP (PEREIRA, 2011).

Dentro das várias propostas disponíveis na literatura, destacam-se o uso de algoritmos genéticos com uso de inteligência artificial na criação de MPS para cenários complexos, descrito no artigo de Vieira e Soares (2008) que busca a minimização dos seguintes parâmetros: níveis de estoque, demanda não atendida, estoque abaixo do mínimo desejado e quantidade de hora-homem extra necessária. Outro uso de meta-heurística também demonstrado na área de MPS é o artigo de Vieira e Ribas (2003), que

usa o método de recozimento simulado, onde busca-se também a minimização dos mesmos parâmetros anteriores.

É possível perceber então, que existem diversas técnicas disponíveis para auxiliar no processo de geração do MPS de uma indústria, no entanto, é necessário estudá-los e verificar a aderência a cada cenário e organização.

2.4 FERRAMENTA E TECNOLOGIA NO PLANEJAMENTO

Esta seção tem como objetivo demonstrar as ferramentas APS e como elas auxiliam nos processos de PPCP.

2.4.1 Tecnologia e programação de computadores

Neste tópico, o objetivo é demonstrar que a tecnologia, aliada ao tema anterior, pode dar uma solução para as complexidades envolvendo a construção do MPS.

Com o avanço da tecnologia disponível no mercado, os problemas do mundo físico vêm sendo cada vez mais sendo possíveis de serem modelados dentro de ambientes computacionais (TARNOWSKI, 2022).

Todos os sistemas de computação, dos mais básicos aos mais complexos, são desenhados para auxiliar a resolução de problemas do mundo real. Neste contexto, existe um conceito dentro das programações de tecnologias conhecido como: Programação Orientada a Objeto (POO) (CARVALHO, 2012).

Na programação orientada a objeto, um programa computacional pode ser definido como um conjunto de objetos e tarefas que tem um desenho para solucionar um problema desejado. Cada objeto então, pode ter suas funções e objetivos específicos que interagem com outros objetos de maneira controlada e configurável (CARVALHO, 2012).

Antes de introduzir o conceito de objeto, que foi citado acima como uma parte da POO, é necessário entender um passo anterior, o conceito de classe. Quando se escreve um problema em programação orientada, os objetos não são definidos um a um, como por exemplo, um carro azul e outro branco, a definição de carro é previamente estabelecida, para que objetos do tipo carro sejam criados com suas respectivas cores. A classe funciona como um modelo previamente especificado para um conjunto de objetos, no exemplo anterior, a classe é um carro, mas, em outros contextos pode ser

uma pessoa com suas propriedades de nome, idade ou qualquer outra definição necessária (CARVALHO, 2012).

Seguido do conceito de classe, um objeto é uma instância de uma classe, em linguagem normal, ser uma instância significa ser concretizado, ter vida. João, José e Maria, por exemplo, são considerados objetos instanciados de uma classe Pessoa, cujo possui propriedades de nome, data de nascimento e afins. Com esta definição, é possível utilizar de um banco de dados que guarde as informações dos objetos e criar uma *interface* gráfica de edição e leitura de dados. A *interface* permite que lógicas de programação sejam acionados por um usuário e realize os procedimentos configurados (CARVALHO, 2012).

Existe dentro do mercado comercial de soluções tecnológicas, uma série de programas e *softwares* para diversos setores, áreas e ramos. Para cada atividade que se deseja realizar, como um simples cálculo matemático através de uma calculadora, existe programação associada à tecnologia. Na medida em que a tecnologia avançou, alguns sistemas de administração da produção surgiram com intenção de auxiliar os processos do PPCP como um todo, inicialmente considerando conceitos do MRP, que cuida de materiais e decisões de produção e mais recentemente realizam planejamento dos recursos de manufatura também (GAITHER *et al.*, 2002; CORRÊA, 2017). Os sistemas de gestão da produção são conhecidos como *Enterprise Resource Planning* (*ERP*), ou Sistemas de Gestão Empresarial, e são divididos em vários módulos que atendem áreas direcionadas – marketing, recursos humanos, controle de produção e afins – de uma indústria.

2.4.2 Sistemas de gestão da produção

Dentro dos sistemas de ERP das indústrias costumam ter módulos de processos do PPCP, onde pode-se realizar a programação ou o planejamento, por exemplo. No entanto, é comum que durante o uso destes módulos, o que é planejado não saia sempre como o esperado. Então, os indivíduos que realizam estes processos via ERP, em uma tentativa de corrigir problemas que no ERP não visualizava, trazem para planilhas eletrônicas e por meio delas tentam realizar o mesmo processo. Porém, passam horas criando um cenário de programação ideal que pode mudar em um pequeno instante devido a qualquer variação nos parâmetros de entrada (LIDDEL, 2020).

Os ERPs disponíveis no mercado, geralmente falham no momento de gerar um sequenciamento ou um planejamento finito considerando as reais características da empresa. Muitas vezes, estes sistemas realizam todo o fluxo sem considerar a capacidade finita, inclusive. Para solucionar este problema, os sistemas APS vêm como uma ferramenta auxiliar nestes processos (LIDDEL, 2020). Portanto, não exclui a necessidade do ERP, pelo contrário, realiza troca de informações integradas com os sistemas de gestão da produção com o objetivo de gerar um sequenciamento que seja aderente às indústrias (LIDDEL, 2020).

Por fim, as ferramentas avançadas trazem consigo uma série de heurísticas padrões que visam atender os complexos processos do PPCP. Em uma tentativa de anteder a maior gama de ramos industriais, nem sempre isso é possível ou é feito com qualidade, dependendo diretamente de muita de configuração do APS e da sagacidade dos consultores. Algumas indústrias podem ficar desassistidas por essas heurísticas padrões por não ter aplicabilidade. Como o código fonte tem acesso exclusivo somente pelo desenvolvedor de um sistema quase sempre, não é possível abrir as heurísticas padrões disponibilizadas e ajustá-las para um melhor encaixe em determinada indústria. Para contornar esta situação as ferramentas atuais de mercado estão abrindo a possibilidade de construir código fonte e implementar dentro da aplicação através de uma *Application Programming Interface* (*API*) (MAUERGAUZ, 2016).

2.4.3 *Advanced Planning and Scheduling*

Os sistemas APS, também conhecidos como sistemas de programação avançada, são relativamente novos dentro da indústria e no Brasil. Foi pela primeira vez implantado no Brasil em 1998 por uma empresa de consultoria (TECMARAN), em parceria com a antiga Preactor, hoje assumida pela Siemens Digital Industries Software ©, que possui um portfólio de tecnologias para atender as indústrias no processo de transformação digital. Os sistemas APS são desenhados para realizar a programação e planejamento considerando a capacidade finita das indústrias (ZAGO, 2013).

Nos níveis de horizontes curtos, médios ou longos, os processos associados a eles foram demonstrados complexos nos tópicos anteriores. Por conta desta complexidade e de uma falta de estrutura interna, comum em muitas empresas, torna-se interessante a utilização da ferramenta APS (ZAGO, 2013). Os sistemas APS funcionam com lógicas avançadas de programação integradas no *software*. Tais lógicas

foram brevemente demonstradas no capítulo anterior, como algumas heurísticas avançadas de otimização, de aproximação e afins. Corrêa (2017) ainda define que sistemas avançados podem ter outras bases diferentes (TARNOWSKI, 2022):

1. Sistemas baseados em liberação de ordens e tarefas;
2. Sistemas especialistas puros;
3. Sistemas apoiados em redes neurais (inteligência artificial).

Não cabe ao escopo deste trabalho detalhar os outros sistemas mencionados por Corrêa *et al.* (2017).

Além de entender como funciona o APS, é importante ressaltar que não substitui a necessidade de um sistema de ERP dentro das indústrias. É sabido que o APS auxilia nas tomadas de decisões onde o ERP costuma ter dificuldades de encontrar soluções com grandes aderências (ZAGO, 2013; CORRÊA *et al.*, 2017).

Os sistemas de produção de capacidade finita, além de considerar a real disponibilidade de recursos da fábrica, também considera outros parâmetros extremamente necessários nos processos do PPCP, como calendários de chão de fábrica, datas de fornecimento de materiais, estoques atuais, recursos secundários (pessoas, ferramentas etc.) para realizar a programação do sequenciamento. Além do sequenciamento, o APS faz a gestão de materiais através de heurísticas inteligentes numa regra de alocação e fornece uma série de análises possíveis de serem feitas do sequenciamento (via *Gantt* e relatórios) e seus impactos diretos nos objetivos da empresa (SLACK *et al.*, 1996; VOLLMANN, 2005; CORRÊA *et al.*, 2017; TUBINO, 2017).

3 METODOLOGIA

Nesta seção, busca-se estabelecer os enquadramentos desta pesquisa, qual o método utilizado e sua área de pesquisa dentro das classificações da Associação Brasileira de Engenharia de Produção (ABEPRO).

3.1 ENQUADRAMENTO METODOLÓGICO

O presente trabalho se encontra com abordagem quantitativa, pois, são necessárias informações quantificáveis para efetuar cálculos matemáticos através de heurísticas (TURRIONI, 2011).

Por seguinte, a natureza é definida como natureza aplicada, cujo objetivo é implementar uma heurística de cálculo de planejamento mestre de produção, utilizando pesquisa pura que possa ser aplicada às indústrias que possuem dificuldades ao gerar o plano. Além disso, faz-se necessário defini-la como aplicada, pois, trata-se de um problema específico de planejamento (GIL, 2010).

Quanto ao propósito, define-se como uma pesquisa exploratória, buscando exemplos e informações acerca dos problemas que circundam o planejamento mestre de produção para considerar um possível aprimoramento do processo de MPS (GIL, 2010).

Sobre o procedimento metodológico adotou-se uma combinação entre revisão da literatura e uma parte do método de pesquisa-ação.

3.2 PROCEDIMENTO METODOLÓGICO

O presente trabalho tem como um dos métodos a revisão da literatura para realizar o objetivo específico que é de rever os conceitos de PCP aplicados ao PMP e demonstrar sua importância dentro das indústrias. Além disso, este método também traz informações auxiliares sobre como é possível realizar o objetivo geral do trabalho utilizando conhecimentos de tecnologias, programação e áreas afins. A revisão traz de maneira breve, a principal solução disponível no mercado com suas vantagens e desvantagens.

Como o objetivo geral do trabalho é implementar uma heurística dentro de uma ferramenta APS, foi necessário encontrar a que mais se adequa ao tema. Dentre todas as possíveis na literatura, a mais aderente ao tema e às condições de desenvolvimento disponíveis é uma que descreve uma das milhares de possibilidades de criação do MPS no artigo Vieira e Favaretto (2006). A heurística do artigo de Vieira e Favaretto (2006)

foi vastamente estudada e foi utilizada como uma parte da metodologia de pesquisa-ação.

O método de pesquisa-ação foi usufruído para desenvolver uma ação a fim de flexibilizar a geração do MPS para as indústrias. Para realizar a pesquisa-ação, o conjunto de dados disponível no artigo citado acima foi utilizado para comparação e criação da heurística através de código de programação dentro de uma funcionalidade disponível dentro do Opcenter *Advanced Planning*, ferramenta APS utilizada neste trabalho. A funcionalidade é conhecida como *Application Programming Interface* (API) – em português, Interface de Programação de Aplicação – e é uma interface de um programa que permite inserir uma nova heurística de cálculo de MPS e considerar os recursos como capacidade finita. Portanto, a API foi a responsável por permitir a flexibilização de um novo procedimento de planejamento. Por fim, a pesquisa-ação é finalizada com uma comparação entre uma heurística nativa da ferramenta em questão e a disponibilizada no artigo.

Em suma, a revisão bibliográfica é realizada para demonstrar e entender como é possível resolver o problema de flexibilização e criação de MPS dentro das indústrias, então, a pesquisa-ação entra como uma auxiliar para disponibilizar uma solução a um problema prático das indústrias (TURRIONI, 2011) e comparar com as soluções existentes.

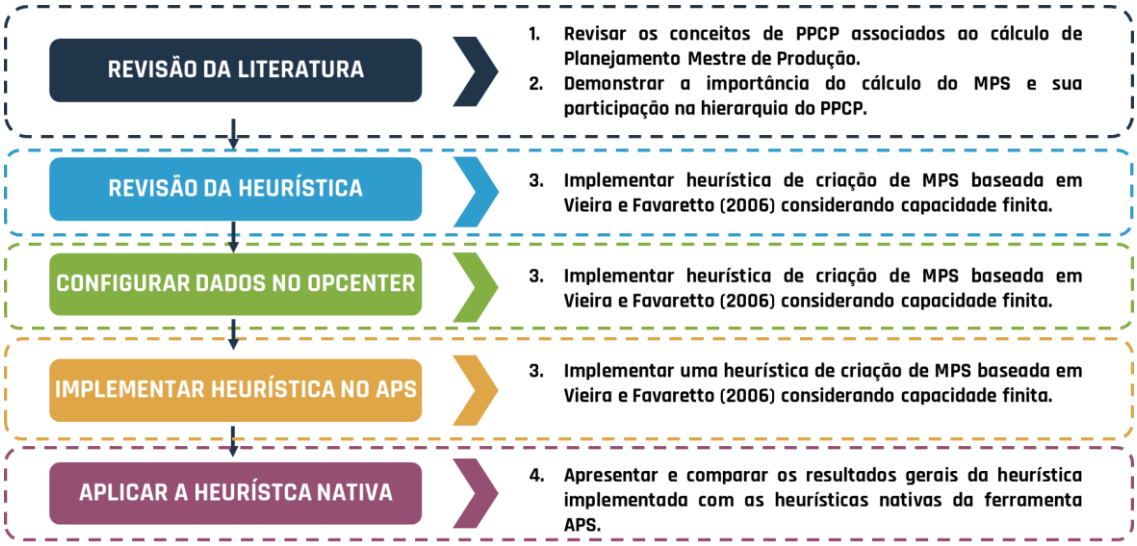
3.3 ÁREA DE PESQUISA

Quanto à área de pesquisa, o trabalho se adequa melhor à “Engenharia de Operações e Processos da Produção”, pois, trata-se de um processo que auxilia a entrega dos produtos primários da empresa. Dentro da área de Engenharia de Operações e Processos, a subárea de enquadramento é o Planejamento, Programação e Controle da Produção (PPCP).

3.4 ETAPAS METODOLÓGICAS

As etapas metodológicas aplicadas neste trabalho têm como meta a realização dos objetivos específicos do trabalho e consequentemente o objetivo geral. A Figura 4 estabelece a relação entre as etapas deste trabalho (lado esquerdo) e os objetivos específicos (lado direito).

Figura 4 – Etapas da metodologia aplicada no trabalho.



Fonte: Autor (2022).

Como se pode ver na Figura 4, a revisão da literatura está diretamente relacionada com o primeiro e o segundo objetivo específico do trabalho. A revisão da heurística de Vieira e Favaretto (2006) é uma etapa necessária para garantir que o fluxo proposto no artigo seja fielmente seguido no momento de implementar a heurística no Opcenter. O conjunto de dados utilizado vêm diretamente do artigo de Vieira e Favaretto (2006) e é aplicado nas configurações do Opcenter para que possa receber qualquer aplicação de lógicas heurísticas posteriormente.

Um dos principais trabalhos realizados neste trabalho é a implementação da heurística no Opcenter (ferramenta APS) e é responsável por garantir o objetivo específico de número 3 neste trabalho.

Após a implementação, o último objetivo é comparar os resultados obtidos com os resultados gerados através de uma das lógicas nativas do Opcenter.

3.4.1 Configuração do Opcenter

O Opcenter *Advanced Planning* é a ferramenta APS utilizada neste trabalho para fins de implementação e comparação de resultados posteriormente.

Em termos de configuração, o programa lê informações de um banco de dados, aloca esses dados em memória computacional e realiza procedimentos através do processador. Após os procedimentos do processador, os dados em memória são atualizados em tempo real e posteriormente são salvos novamente no banco de dados.

O nível de cadastro realizado com o conjunto de dados citado nos tópicos abaixo se encontra no banco de dados, pois é onde pode-se inserir informações para trabalhos posteriores.

Para realizar o cadastro de informações no banco de dados, o Opcenter permite a inserção manual de dado e dispõe, também, de uma funcionalidade conhecida como Preactor *Import Option* (PIO) que consegue trazer dados diretamente de um banco de dados nativo ou de arquivos em formatos *.csv*, por exemplo.

A realização do cadastro foi feita parcialmente manualmente através da interface gráfica do Opcenter e parcialmente através de arquivos em *.csv*.

3.4.2 Ambiente de desenvolvimento da programação

A IDE (*Integrated Development Environment*) utilizada no desenvolvimento da heurística foi o Visual Studio 2015 que pertence aos direitos da Microsoft Corporation. O Visual Studio 2015 foi utilizado pois é o ambiente proposto pela própria fornecedora (Siemens) do Opcenter para uso da API, outras IDEs, portanto, ficaram fora da possibilidade de uso. Além disso, a versão 2015 foi utilizada pois é a versão mais robusta para desenvolvimento dentro da API consensada, de maneira não acadêmica, pelos desenvolvedores do Opcenter.

3.4.3 Conjunto de dados utilizado

O conjunto de dados utilizado foi extraído do artigo de Vieira e Favaretto (2006) e os níveis de dados utilizados foram:

- **Itens**:
    a. Itens A, B, C e D.
- **Recursos:**
    a. Recursos L1, L2 e L3.
- **Horizontes de planejamento:**
    a. Três semanas e 2 meses.
- **Estoques:**
    a. Código, Quantidade e Data da Contagem como é disposto na Tabela 6.
- **Demandas:**

a. Tipo, Ordem, Código do Item, Data e Quantidade como é disposto na Tabela 7.

3.4.4 Implementação da heurística através da programação no Opcenter

A linguagem de desenvolvimento utilizada para a escrita lógica do novo cálculo de MPS foi o C# com o SDK (*Software Development Kit*) .NET versão 4.5.1.

A heurística foi implementada no Opcenter *Advanced Planning* (ferramenta APS) através de uma interface conhecida como API, que consegue implementar novos códigos fontes dentro da ferramenta. Para o uso da API, foi necessário utilizar do Visual Studio Code 2015 como IDE de desenvolvimento.

3.4.5 Validação da heurística implementada

Para realizar a validação dos resultados gerados pela implementação foi necessário ter, além da heurística implementada, o cadastro dos dados do artigo de Vieira e Favaretto (2006) no Opcenter. Os dados gerados pela heurística desenvolvida pelo artigo foram comparados com os dados gerados pela heurística implementada no Opcenter. Apenas um período de planejamento foi comparado para fins de validação e garantiu que todo o restante seguia o mesmo fluxo de operações descritos na heurística proposta.

3.4.6 Comparação do resultado nativo *versus* implementado

Para realizar a comparação de resultados, foi necessário escolher uma das 5 heurísticas disponíveis dentro do Opcenter. São elas (documentação Opcenter *Advanced Planning*, 2022):

- **Capacidade infinita:** O modo de planejamento de capacidade infinita criará um plano usando apenas o recurso de planejamento primário de um item. Os volumes MPS não são limitados, independentemente da capacidade disponível. Isso pode levar à sobrecarga da capacidade de um recurso primário para qualquer item. Qualquer recurso de planejamento secundário não será considerado neste modo de planejamento.
- **Restringir:** O modo de planejamento restrito, como **Capacidade infinita**, planejará apenas usando o recurso de planejamento principal de um item e

não utilizará nenhum recurso de planejamento secundário. Ele tentará "espremer" os volumes de MPS com base na capacidade do recurso.

- **Mover:** Ao usar o modo de planejamento Mover, o Opcenter procurará distribuir a produção excedida em recursos secundários, se possível. Se a necessidade total de um item for maior que a capacidade de um recurso, em um intervalo de tempo, o item será movido para outro recurso de planejamento que tenha capacidade disponível. Se necessário, os itens podem ser divididos em vários recursos para satisfazer a produção. Nenhuma produção será movida se os recursos secundários forem sobrecarregados.
- **Mover e então restringir:** Ao planejar usando este modo, o Opcenter procurará mover a produção em torno dos recursos disponíveis (dentro do mesmo grupo e horizonte) do item antes de restringir. A produção pode ocorrer em vários recursos para manter o valor de MPS conforme necessário. Se for necessária muita produção em todos os itens e recursos, o Opcenter PL procurará reduzir o valor de MPS usando a lógica de restrição descrita anteriormente.
- **Restringir e então mover:** Ao planejar usando este modo, o Opcenter procurará produzir itens, principalmente no recurso primário do item. Se não houver capacidade suficiente para fazer o volume necessário após a restrição, o Opcenter PL procurará usar a opção Mover. Caso o recurso secundário estiver cheio, ele passará para o próximo recurso. Se não houver recursos secundários para os quais a produção possa se deslocar, porque ela ficará sobrecarregada, nenhum ajuste será feito.

Para garantir uma coesão lógica entre o implementado e o nativo, utilizou-se **modo mover e então restringir** por se assemelhar mais à heurística implementada.

4 DESENVOLVIMENTO

Esta seção tem como objetivo esclarecer o desenvolvimento da heurística dentro da ferramenta APS, como ela funciona e compará-la com a heurística nativa do APS em questão.

4.1 LÓGICA DA HEURÍSTICA IMPLEMENTADA

A lógica descreve os critérios iniciais para uma construção inicial do MPS e envolve os seguintes parâmetros de entrada na heurística de Vieira e Favaretto (2006):

1. Demanda do período (Necessidades grosseiras)
2. Estoque inicial
3. Volume terceirizado
4. Tamanho de lote mínimo
5. Tamanho de lote padrão
6. Estoque máximo

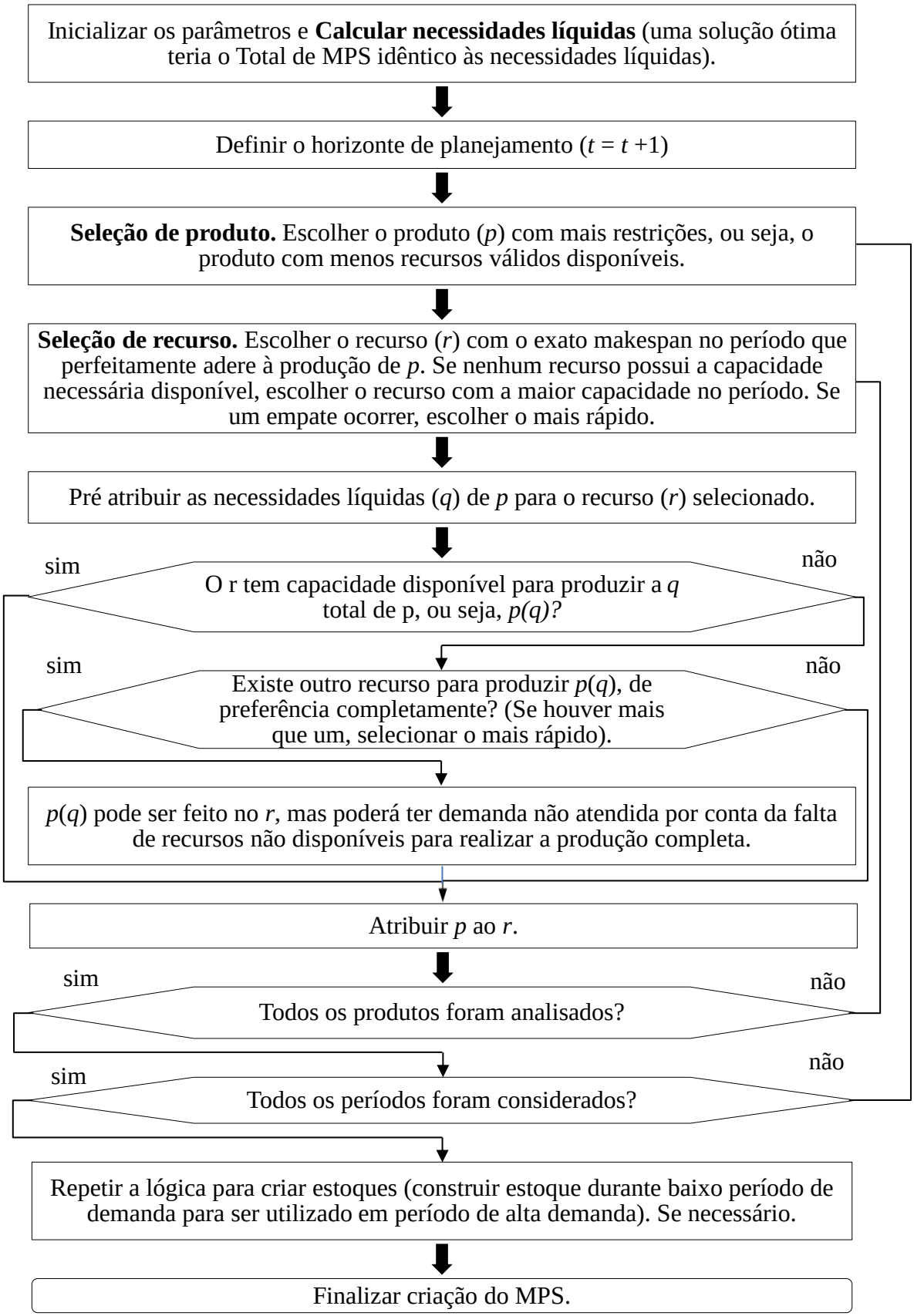
Estes dados anteriores são dados como entrada para o conceito de necessidades líquidas:

$$\boldsymbol{Necessidades\ líquidas} =$$
$$Mínimo\ (Múltiplo\ (Mínimo\ (Máximo\ (Demanda\ do\ período - (estoque\ inicial + volume\ terceirizado);\ 0);\ Tamanho\ de\ lote\ mínimo);\ Tamanho\ de\ lote\ padrão);\ Estoque\ máximo);$$

Após o cálculo de necessidades líquidas para inicializar a produção, é necessário seguir um fluxo que é descrito na Figura 5.

O processo se repete até que o último produto, dentro do último período tenha sido analisado e considerado para produção dentro dos recursos disponíveis. Existe também a possibilidade de refazer a lógica para produzir para estoque em períodos de baixa demanda e utilizá-lo em períodos em que a demanda é alta.

Figura 5 – Fluxo de atividades da heurística.



Fonte: Adaptado de Vieira e Favaretto (2006).

4.2 IMPLEMENTAÇÃO DA HEURÍSTICA ATRAVÉS DA PROGRAMAÇÃO

Este tópico contém todas as lógicas das funções desenvolvidas através da programação e como elas se relacionam. Também é exposto o conjunto de classes e objetos criados para desenvolvimento. Por fim, uma visão sobre os *hard-codes* realizados no desenvolvimento é declarada para fins de inflexibilidade via importação de dados.

4.2.1 Desenvolvimento da heurística

Dentro dos códigos de desenvolvimento, foi criada uma série de classes e objetos para receber e tratar informações do conjunto de dados disponível no artigo de Vieira e Favaretto (2006). As classes são:

1. Declarations: Uma classe necessária para criar declarações de variáveis utilizadas em todo o projeto de desenvolvimento;
2. Demand: É uma classe para receber as quantidades de demandas por itens dentro de cada período. As demandas para cada item num período são cruzadas com a classe ItemsResourceData para criar uma matriz de controle de capacidade posteriormente;
3. Item: Uma classe para receber as informações de SKUs;
4. ItemsResourceData: Uma classe que cruza os itens por recursos e suas taxas de produções específicas por recurso;
5. Resource: É uma classe que recebe a tabela de recursos disponível e suas capacidades;
6. Stock: É utilizada para receber informações de estoques atuais da fábrica;
7. NonAggregatedDemand: Necessária para receber as demandas grosseiras, seja pedido em carteira ou previsão de demanda;
8. DataExport: Uma classe para buscar os dados de saída da lógica e exportar para um arquivo em *.csv*.

Para que as classes pudessem ter um objetivo, uma série de métodos e funções que utilizam de objetos dessas classes foi desenvolvida. Dois conhecidos tipos de funções são utilizados neste desenvolvimento, os *getters* e os *setters*, uma busca dado e outra ajusta dado, respectivamente. A maioria das funções são *getters*, que costumam ser auxiliares no desenvolvimento da lógica principal:

- **getItems** ( ){...}: este *get* tem objetivo de buscar os itens que terão o cálculo de MPS realizado. Cria um objeto instanciado por Item em

formato de lista chamado de ItemsList. Somente entram nesta lista de itens os SKUs cadastrados;

- **getPlanningResources** ( ){...}: esta função tem o objetivo de criar uma lista com todos os recursos do conjunto de dados disponível e associar a capacidade disponível para cada período disposto na demanda (necessidades grosseiras). Esta função utiliza de um outro método getNonAggDemand (), descrita a seguir. Para cada recurso disponível, as capacidades consideradas por variação período são: capacidade mensal e semanal e disponibilidade de horas extras mensais e semanais. Este *get* utiliza da classe Resource para instanciar um objeto do tipo lista chamado de ResourcesList;
- **getNonAggDemand** ( ){...}: esta *get* busca todas as demandas de todos os períodos, sejam elas dadas por previsão de demanda ou pedidos em carteira. É onde o parâmetro de necessidades grosseiras do período é inserido numa objeto do tipo lista da classe Demand;
- **getItemsResourceData** ( ){...}: esta função, um pouco mais complexa, cria dados em formato de matriz, ou seja, para cada linha de um item, haverá *n* colunas com as informações de tempo de cada recurso especificado como válido de produção. Os dados em formato matriz são como os dispostos no formato da Tabela 2. Uma classe da ItemsResourceData foi utilizada para instanciar um objeto chamado ItemsResourceData.

Tabela 2 – Formato de dados de tempos de recursos por itens.

	Recurso		
Item	L1	L2	L3
A	15	20	25
B	20	20	25
C	15	30	20
D	-	10	15

Fonte: Adaptado de Vieira e Favaretto (2006).

Os *getters* apresentados até o momento são apenas de busca de dados originais e uma certa manipulação entre informações que permita o desenvolvimento em funções principais. Por outro lado, quando os cálculos de MPS mais profundos são realizados, é necessário criar outras funções de busca e atualização de dados devido às manipulações

ocorrendo de maneira iterativa. Os *getters* e *setters* desenvolvidos para a manipulação e busca de dado em memória são:

- **getResourceRate** (*string* itemCode, *string* resource){...}: esta função retorna dados de taxas de produção de acordo com cada item e recurso solicitado quando é invocada. Utiliza do objeto ItemsResourceData para realizar as buscas de taxas e retorna o valor assim que atende os parâmetros fornecidos;
- **getOriginalCapacity** (*string* resource, *DateTime* date, *string* day){...}: esta função tem como objetivo retornar a capacidade original no período para fins de comparação dentro de uma das funções principais, a CalculateMPS(). Depende diretamente de três parâmetros para retornar o valor da capacidade: o recurso, a data de demanda e o tipo de horizonte de planejamento atual (mês ou semana);
- **getItemsResourceCount** (*string* itemCode){...}: uma função necessária para buscar os produtos com maior restrição de produção de recursos. É utilizada no momento de ordenar os dados de entrada para o cálculo de criação de MPS, pois um dos passos necessários no fluxo da heurística é ordenar os produtos pela menor disponibilidade de recursos de produção. É utilizado diretamente pela função createGridControl();
- **getResourceAvaliableCapacity** (*string* resource, *DateTime* date, *string* day){…}: este *get* é a função responsável pela busca da capacidade disponível no momento de atribuição de MPS à um recurso específico. É utilizado diretamente pela função calculateMPS(). Os valores disponíveis de capacidade são alterados por funções iterativas de *setters* dentro do createGridControl();
- **setResourceStats** (*string* resource, *DateTime* date, *double* newCapacity, *string* day) {…}: esta função *set* está diretamente relacionada com a atualização de capacidade dos recursos dentro dos períodos. Esta função é acionada quando um recurso tem uma quantidade *q* de um produto *p* atribuído a algum recurso *r*, ou seja, o produto passa por uma série de condições dentro do fluxo de criação do MPS e quando tem seu recurso alocado, diminui a capacidade disponível por conta da própria utilização. É invocada pela função calculateMPS();

- **setDemandFit** (*string* itemCode, *DateTime* date, *int* fitsInFull, *int* fitsInPartial = 0){…}: esta função foi desenvolvida para garantir verificar se um recurso *r* produz uma quantidade *q* de um produto *p*, caso ele produza totalmente, o parâmetro fitsInFull é utilizado para guardar esta informação, caso seja produção parcial, o parâmetro fitsInPartial guarda o quanto de uma demanda *q* foi produzida para um produto *p* em um período determinado. É acionada pela função calculateMPS().

Ainda, a última função que foi criada com o objetivo de atender a heurística implementada foi a seguinte:

- **chooseResource** (string itemCode, DateTime date, string day) {…}: é a responsável por escolher o recurso *r* disponível no período que possa produzir a quantidade *q* demandada pelo produto *p*. Esta função retorna exatamente o nome do recurso que realiza mais MPS de um produto, ou seja, dentre todas as capacidades disponíveis e taxas diferentes de recursos por item, traz o que produz o maior volume de *p*. Utiliza diretamente a função getResourceAvailableCapacity().

Além das classes, objetos e funções demonstradas anteriormente, o núcleo da ferramenta é composto de uma função genérica que encapsula outras três funções vitais na criação do MPS conforme a heurística proposta. Leia-se o código abaixo como: Para cada período de demanda onde houver previsão de demanda ou pedido em carteira, calcule as necessidades líquidas de cada produto, crie uma matriz de controle de dados e realize a criação do MPS atualizando dados da matriz-controle. O código em linguagem de programação *C#* é:

```
public void runMPS()
    {
      var dates = DatesList.Select(x => x.DemandDate).Distinct();
      foreach (var date in dates)
      {
        calculateNetRequirements(date);
        createGridControl(date);
        calculateMPS(date);
      }
    }
```

Para a função calculateNetRequirements() ficou o processo inicial de abertura de necessidades líquidas. Vieira (2006) comenta que um MPS ideal teria suas quantidades iguais às necessidades líquidas, no entanto, este é um cenário onde muito provavelmente não seria possível produzir todo o MPS desejado por falta de capacidade de produção. Para calcular todas as necessidades líquidas, a função atualizou as quantidades de estoques (estoque em mãos) disponíveis de cada produto. Após esta atualização de estoque, o cálculo das necessidades é feito com base na seguinte fórmula:

$$\boldsymbol{Necessidades\ líquidas} = Mínimo\ (Múltiplo\ (Mínimo\ (Máximo\ (Demanda\ do\ período - (estoque\ inicial + volume\ terceirizado);\ 0);\ Tamanho\ de\ lote\ mínimo);\ Tamanho\ de\ lote\ padrão);\ Estoque\ máximo);$$

Portanto, para cada item que terá um MPS potencial, foi necessário buscar os valores de:

- Necessidades grosseiras;
- Estoque inicial;
- Volume terceirizado;
- Tamanho de lote mínimo;
- Tamanho de lote padrão;
- Estoque máximo;
- Estoque de segurança.

Uma série de operações lógicas, dispostas na Figura 6, foi desenvolvida com programação a fim de realizar o cálculo das necessidades líquidas (caculateNetRequirements) e no fim, atualizar os dados em memória computacional para a próxima função que considera a construção do MPS final com capacidade finita.

Figura 6 – Função de abertura de necessidades líquidas

```
// calcula as necessidades líquidas para iniciar o processo de criação de MPS
public int calculateNetRequirements(DateTime date)
{
    // Net Req = Min(Mult(Min(Max([gross - (initial inv + subcont)],0), min lot size), standard lot size), max inv level)
    //calcula os estoques para abrir o initial inventory
    sharedPreactor.Planner.CalculateStock();
    string lastCode = null;
    DateTime lastDate = sharedPreactor.ReadFieldDateTime(tblDemand, clnDemandDate, 1);
    int demandLength = sharedPreactor.RecordCount(tblDemand);
    for (int i = 1; i <= demandLength; i++)
    {
        string currentCode = sharedPreactor.ReadFieldString(tblDemand, clnDemandCode, i);
        DateTime currentDate = sharedPreactor.ReadFieldDateTime(tblDemand, clnDemandDate, i);
        if (currentCode != lastCode && currentDate == date)
        {
            double initialInventory = sharedPreactor.ReadFieldDouble(tblDemand, clnDemandOpeningStock, i);
            initialInventory = Math.Max(initialInventory, 0);
            if (initialInventory >= 0)
            {
                double subcontracted = sharedPreactor.ReadFieldDouble(tblDemand, clnDemandSubcontracted, i);
                double grossRequirements = sharedPreactor.ReadFieldDouble(tblDemand, clnDemandDemand, i);
                double minimumLotSize = sharedPreactor.ReadFieldDouble(tblDemand, clnDemandMinimumLotSize, i);
                double standardLotSize = sharedPreactor.ReadFieldDouble(tblDemand, clnDemandReorderMultiple, i);
                double maximumInventoryLevel = sharedPreactor.ReadFieldDouble(tblDemand, clndDemandMaximumInventoryLevel, i);
                double safetyInventory;
                string currentDay = sharedPreactor.ReadFieldString(tblDemand, clnDemandDay, i);
                if (currentDay == "Week")
                {
                    safetyInventory = sharedPreactor.ReadFieldDouble(tblDemand, clnDemandSafetyInventory, i);
                }
                else
                    safetyInventory = 2000;
                double var0 = safetyInventory + grossRequirements - initialInventory - subcontracted;
                double val1 = Math.Max(var0, 0);
                double val2 = Math.Max(val1, minimumLotSize);
                double val3 = Math.Ceiling(val2 / standardLotSize) * standardLotSize;
                double val4 = Math.Min(val3, maximumInventoryLevel);
                double netRequirements = val4;

                FormatFieldPair demandNetRequirements = new FormatFieldPair(sharedPreactor.GetFormatNumber(tblDemand), sharedPr
                sharedPreactor.WriteField(demandNetRequirements, i, netRequirements);
            }

            lastCode = currentCode;
        }

    }
    sharedPreactor.Planner.RefreshPlannerGrid();
    return 0;
}
```

Fonte: Autor (2022).

Na sequência do fluxo no código demonstrado, a createGridControl() cria uma matriz que cruza os produtos por demanda *vs* recursos de produção e a deixa pronta para ser utilizada pela calculateMPS(). É uma função simples que atualiza um objeto chamado DemandList, a matriz final, e é quem faz a ordenação dos dados com três parâmetros, como segue na ordem abaixo:

1. Ordenar por demanda, da menor data para a maior;
2. Ordenar pela contagem de recursos válidos, da menor quantidade para a maior;
3. Ordenar por necessidades líquidas, do maior para o menor.

Como última e mais importante parte da genMPS(), a calculateMPS() é responsável por receber a data de planejamento desejada e iniciar a atualização de informações dentro da matriz DemandList.

Esta função começa buscando o primeiro item ordenado na matriz e procura o recurso mais rápido disponível para pré-atribuir o MPS (necessidade líquida), se necessário for, criará um *backlog* para produzir parte das necessidades em recursos diferentes. Após a atribuição desta quantidade *q* a um recurso *r*, a função diminui a capacidade disponível de *r* no período fornecido como parâmetro de entrada ao chamar a função. Depois de diminui a capacidade e atribuir o recurso, atualiza também a informação de quanto há ainda para produzir de *q* (quando existir) e indica se a quantidade *q* de MPS coube em apenas um recurso ou foi atendido parcialmente por mais do que um recurso. Caso um recurso não consiga atender totalmente com a capacidade original, existe a possibilidade de utilizar a capacidade disponível em horas extras (HE) para atender o máximo das necessidades líquidas. Existe o uso direto da função chooseResource() para retornar o recurso mais indicado para atribuição de MPS. Por fim, caso não seja possível atribuir totalmente as necessidades líquidas, a função toma cuidado para que um lote de produção somente seja associado a um recurso se este for maior ou igual ao lote mínimo de produção do produto que está sendo analisado.

Dentro das funções dispostas anteriormente, existem várias outras que buscam cumprir o objetivo final da criação de MPS e seguir o fluxo da heurística proposta. Parte do código fonte da lógica implementada está disponível no APÊNDICE A, no entanto, pode ser solicitado no seguinte repositório on-line: https://github.com/leofcj/MPS-Heuristic.

Por fim, alguns *hard-codes* foram implementados visando atender o conjunto de dados da empresa fictícia do artigo de Vieira e Favaretto (2006). Os *hard-codes* se concentram apenas no momento de receber a quantidade disponível de capacidade por período, não sendo necessariamente um problema para a flexibilização deste *input* no futuro.

4.3 OPCENTER ADVANCED PLANNING

O Opcenter, ferramenta APS em questão de aplicação da heurística, é um dos produtos do portfólio da Siemens Digital Industries Software © que auxilia as indústrias no caminho da digitalização dos processos da indústria. A versão utilizada foi o Opcenter *Advanced Planning* 18.6.

O Opcenter é um dos mais conhecidos *softwares* de APS do mundo, anteriormente conhecido como Preactor, atendendo indústrias de diversos ramos e configurações de trabalho, desde um MTS a um ETO de manufatura complexa.

Os projetos de implantação do Opcenter geralmente envolvem uma série de especificações e estudos com o objetivo de entender a aderência ao *modus operandi* da indústria e seu potencial retorno. Além disso, conta com muitas horas de consultores que além de conhecer a ferramenta, são conhecedores também da cadeia de suprimentos e processos envolvidos.

Em termos de configuração de tecnologia, o Opcenter é um *software* que realiza uma comunicação direta com um banco de dados nativo em SQL Server e a partir da leitura das informações, estas são alocadas em memória RAM e tem suas heurísticas aplicadas via processador.

Para que a nova funcionalidade implementada tivesse aplicação e validação no Opcenter, foi necessário configurar toda uma modelagem com o conjunto de dados disponíveis no artigo de Vieira e Favaretto (2006), ou seja, foi necessário colocar informações no banco de dados da ferramenta. As configurações de dados envolveram as seguintes tabelas:

Tabela 3 – Recursos de planejamento

Nome	Habilitar Restrição?	Capacidade disponível para grupo inicial de ordens - %
L1	Sim	100.00
L2	Sim	100.00
L3	Sim	100.00

Fonte: Autor (2022).

Na Tabela 3 de recursos os campos significam: o nome do recurso, se terá capacidade finita e a capacidade disponível no primeiro horizonte de planejamento, respectivamente.

Tabela 4 - Grupo de recursos de planejamento

Grupo de Recursos	Recurso
Todos	L1
Todos	L2
Todos	L3
Item D	L2
Item D	L3

Fonte: Autor (2022).

Para cada item dentro do Opcenter, é necessário alocar um grupo de recursos válidos para produção, neste caso, na Tabela 4, um grupo "Todos" foi criado a fim de atender os itens A, B e C, significando que todos os recursos da modelagem estão disponíveis para estes produtos, enquanto o Item D teve um grupo específico para validar somente L2 e L3, pois o recurso L1 não pode produzi-lo neste conjunto de dados.

Tabela 5 – Itens

Item	Planejar?	Quantidade Mínima de Reordenamento	Estoque de Segurança	Grupo de Recursos
A	Sim	200	400	Todos
B	Sim	200	500	Todos
C	Sim	200	400	Todos
D	Sim	200	300	Item D

Fonte: Autor (2022).

Na tabela de Itens (Tabela 5) foi necessário cadastrar cada item com suas características de abertura de lotes e controle de estoque. A quantidade de estoque mínima de reordenamento, foi espelhada com o mesmo valor para a quantidade de reordenamento múltiplo (após o mínimo, de quanto em quanto pode-se incrementar o valor inicial de MPS). O estoque de segurança foi configurado para os itens em nível de horizonte de semanas e quando foi necessário tratar os estoques de segurança a nível de mês, os códigos via API realizaram um *hard-code* atribuindo o valor de 2000, assim como o valor no artigo de Vieira e Favaretto (2006). Vale também ressaltar que a modelagem padrão do Opcenter não contém os campos de níveis de estoques de segurança, máximos e mínimos, esta configuração foi realizada diretamente por uma funcionalidade conhecida como *Table Definition File* (*TDF*) e permite que novos campos sejam adicionados e considerados em toda a modelagem, inclusive no banco de

dados. As colunas novas adicionadas na tabela de Itens pelo TDF foram parametrizadas como se segue:

```
Safety Inventory,1,REAL,
    FORMAT (.2)
    SUBSTITUTE(-1->"Unspecified")
    DIALOG ONLY:
Maximum Inventory Level,1,REAL,
    FORMAT (.2)
    SUBSTITUTE(-1->"Unspecified")
    DIALOG ONLY:
```

Ou seja, o Opcenter não possui uma heurística nativa que atenda indústrias que utilizem de conceito de níveis de estoques.

Tabela 6 - Estoques

Tipo	Código Item	Descrição	Data da contagem	Quantidade
Contagem	A	Item A	19/09/2022	100
Contagem	B	Item B	19/09/2022	300
Contagem	C	Item C	19/09/2022	250
Contagem	D	Item D	19/09/2022	350

Fonte: Autor (2022).

Uma tabela de Estoques foi necessária para realizar os cálculos de necessidades líquidas e indicar o quanto existe para cada um dos produtos e em que data esta contagem foi efetuada.

Tabela 7 – Demanda não agregada

Tipo	Ordem Nº	Código Item	Data	Quantidade
Previsão	PV01	A	19/09/2022	400
Previsão	PV02	B	19/09/2022	400
...	...	...	...	...
Previsão	PV16	D	10/10/2022	1100
Previsão	PV17	A	11/11/2022	2000
Previsão	PV18	B	11/11/2022	2000
Previsão	PV19	C	11/11/2022	2000
Previsão	PV20	D	11/11/2022	2000

Fonte: Autor (2022).

A demanda não agregada, no Opcenter, representa toda a demanda que deve ser analisada para a criação do MPS. Recebe tanto os dados de pedidos em carteira (venda para um cliente) como de previsão de vendas. A data de demanda impacta diretamente nos horizontes de planejamento, pois é onde a ferramenta irá buscar dados para a criação dentro de um período específico.

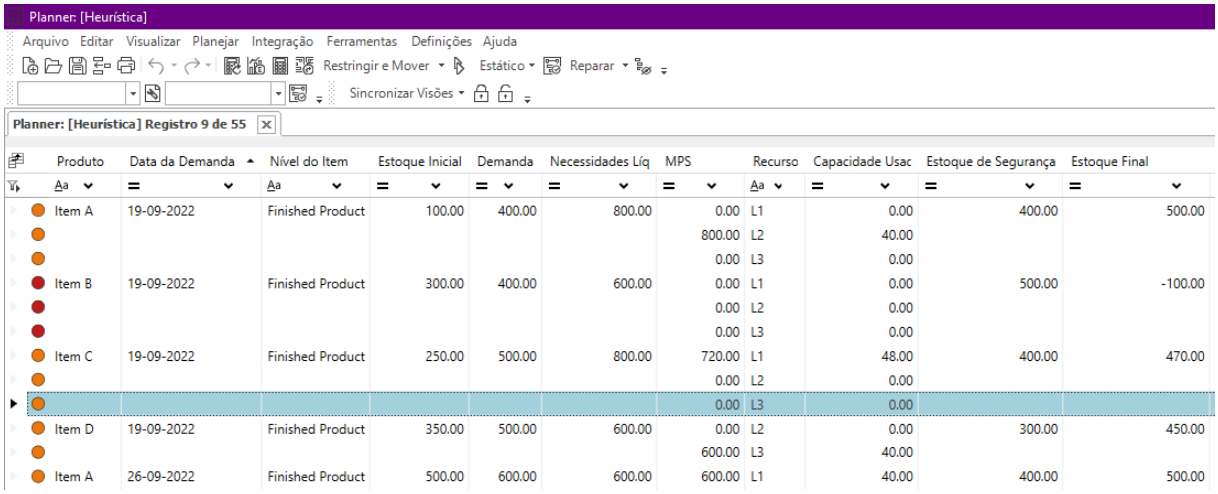
A tabela nativa de Demanda que retorna os resultados do MPS também necessitou de uma configuração específica, pois não recebe dados de entrada. É resultado de todos os dados configurados anteriormente em formato de um grid classificado por *KUDOSOCXGRID*. A tabela de Demanda do Opcenter é a mais importante visualmente, sendo responsável por trazer os resultados de MPS gerados pelas heurísticas nativas da ferramenta. Como comentado anteriormente, esta tabela também não possui os campos de níveis de estoque máximo, mínimo e de segurança, além disso, não possui o conceito de necessidades líquidas que a lógica de Vieira (2006) leva em consideração. Para a correta configuração da modelagem a fim de receber a heurística proposta, houve uma configuração no *Table Definition File* da Tabela de Demanda:

```
Safety Inventory,0,REAL,
    FORCE COLUMN BREAK
    DATABASE(Items(Safety Inventory))
    READ ONLY
    INHERIT FROM PARENT
    UPDATE REFERENCE(Code)
    DIALOG ONLY:
  Minimum Reorder Quantity,0,REAL,
    FORCE COLUMN BREAK
    DATABASE(Items(Minimum Reorder Quantity))
    READ ONLY
    INHERIT FROM PARENT
    UPDATE REFERENCE(Code)
    DIALOG ONLY:
  Reorder Multiple,0,REAL,
    FORCE COLUMN BREAK
    DATABASE(Items(Reorder Multiple))
    READ ONLY
    INHERIT FROM PARENT
```

```
    UPDATE REFERENCE(Code)
    DIALOG ONLY:
  Maximum Inventory Level,0,REAL,
    FORCE COLUMN BREAK
    DATABASE(Items(Maximum Inventory Level))
    READ ONLY
    INHERIT FROM PARENT
    UPDATE REFERENCE(Code)
    DIALOG ONLY:
  Subcontracted,0,REAL,
    FORCE COLUMN BREAK
    READ ONLY
    DIALOG ONLY:
  Net Requirements,0,REAL,
    FORCE COLUMN BREAK
    INHERIT FROM PARENT
    READ ONLY
    DIALOG ONLY:
```

É possível observar a tabela de Demanda, dentro do Opcenter, com alguns resultados da heurística implementada na Figura 7.

Figura 7 – Tabela de demanda no Opcenter



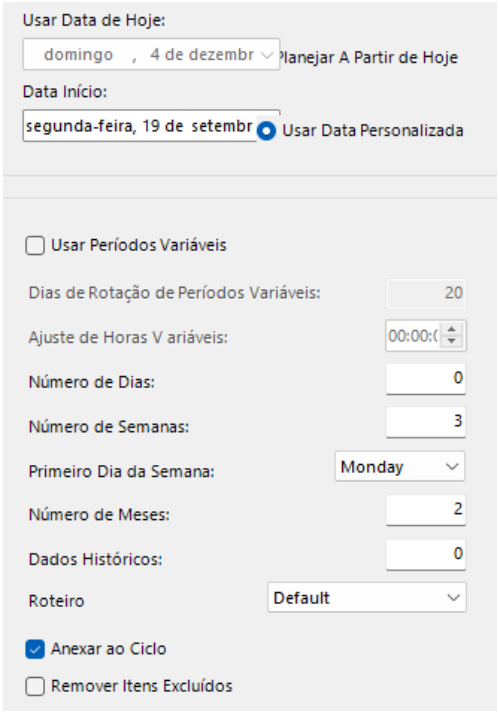
Produto	Data da Demanda	Nível do Item	Estoque Inicial	Demanda	Necessidades Líq	MPS	Recurso	Capacidade Usac	Estoque de Segurança	Estoque Final
Item A	19-09-2022	Finished Product	100.00	400.00	800.00	0.00	L1	0.00	400.00	500.00
						800.00	L2	40.00		
						0.00	L3	0.00		
Item B	19-09-2022	Finished Product	300.00	400.00	600.00	0.00	L1	0.00	500.00	-100.00
						0.00	L2	0.00		
						0.00	L3	0.00		
Item C	19-09-2022	Finished Product	250.00	500.00	800.00	720.00	L1	48.00	400.00	470.00
						0.00	L2	0.00		
						0.00	L3	0.00		
Item D	19-09-2022	Finished Product	350.00	500.00	600.00	0.00	L2	0.00	300.00	450.00
						600.00	L3	40.00		
Item A	26-09-2022	Finished Product	500.00	600.00	600.00	600.00	L1	40.00	400.00	500.00

Fonte: Autor (2022).

Percebe-se que os itens A, B, C e D são apresentados na Figura 7 conforme o cadastro de recursos, estoques, datas de demandas e quantidade demandada do período.

As configurações de horizontes de planejamento foram realizadas diretamente pelo configurador de grid da ferramenta conforme Figura 8.

Figura 8 – Configuração do horizonte de planejamento



Fonte: Autor (2022).

Durante as configurações, uma série de funcionalidades nativas do Opcenter não foram utilizadas pela ausência de necessidade e não cabe dentro do escopo do trabalho abrir o detalhamento destas possibilidades.

É importante ressaltar que a ordem seguida e demonstrada no cadastro das informações é extremamente importante na configuração, pois a ferramenta cruza dados entre tabelas de maneira relacional. Com todas as configurações no Opcenter realizadas, é possível aplicar a heurística e fazer comparações com as lógicas nativas já existentes.

4.4 VALIDAÇÃO DO FUNCIONAMENTO DA HEURÍSTICA

Para garantir que a heurística implementada seguiu realmente os passos descritos no artigo, a configuração do Opcenter foi realizada com o mesmo conjunto de dados e aplicou-se a nova lógica. Como fins de comparação, utilizou-se a primeira semana do horizonte de planejamento, neste caso, semana 2. Os resultados sobre o conjunto de dados da Semana 2 fornecidos pelo artigo de Vieira e Favaretto (2006) estão na Figura 9.

Figura 9 – Resultados traduzidos originais da heurística no artigo.

	Semana 2				
	A	B	C	D	Capacidade utilizada
Em estoque	100	300	250	350	
Estoque inicial (período)	100	300	250	350	
Necessidades grosseiras	400	400	500	500	
Lote padrão	200	200	200	200	
Estoque de segurança	400	500	400	300	
Necessidades líquidas	800	600	800	600	
MPS					
L1			720		48
L2	800				40
L3				600	40
Total	800	0	720	600	

Fonte: Adaptado de Vieira e Favaretto (2006).

No outro lado da comparação, os resultados gerados pela heurística dentro do Opcenter podem ser vistos na Tabela 8.

É possível perceber que os resultados da Semana 2 gerados tanto pela heurística do artigo, como pela do Opcenter são idênticos, no entanto, nas semanas posteriores houve um leve descolamento das informações devido à lógica de realizar cortes de produção nos períodos posteriores que não foi descrita no escopo do artigo, ou seja, quando não há capacidade suficiente, qual é a lógica utilizada para cortar o volume de MPS. Esta situação levou a uma leve falta de aderência nos dados finais da heurística, mas não prejudicam o fluxo desenhado original e podem ser comparados com o APÊNDICE B e os dados no artigo original.

Tabela 8 – Resultados da heurística no Opcenter

	Semana 2				
	A	B	C	D	Capacidade utilizada
Em estoque	100	300	250	350	
Estoque inicial (período)	100	300	250	350	
Necessidades grosseiras	400	400	500	500	
Lote padrão	200	200	200	200	
Estoque de segurança	400	500	400	300	
Necessidades líquidas	800	600	800	600	
MPS					
L1			720		48
L2	800				40
L3				600	40
Total	800	0	720	600	

Fonte: Autor (2022).

4.5 COMPARAÇÃO COM UMA HEURÍSTICA NATIVA DA FERRAMENTA APS

Na busca de comparar a qualidade da solução, é possível fazer uma análise contra algumas heurísticas do Opcenter. A ferramenta possui algumas heurísticas nativas que se comportam como foi descrito no capítulo das etapas metodológicas:

- **Capacidade infinita;**
- **Restringir;**
- **Mover;**
- **Mover e então restringir;**
- **Restringir e então mover.**

Antes de realizar a comparação, é importante ressaltar que as comparações podem não ser muito adequadas, pois o Opcenter trabalha com políticas de cobertura para calcular o MPS, enquanto a heurística desenvolvida utiliza de níveis estoques de segurança, mínimo e máximo.

Para realizar a comparação, o mesmo cenário do artigo (demonstrado no Anexo A) foi configurado de maneira aproximada – realizando cálculos grosseiros para encontrar os dias de cobertura – no Opcenter e utilizou-se o **modo mover e então restringir** pois é a heurística mais trabalhada dentro da ferramenta em indústrias e que

tem os melhores resultados com fins de comparação. Para os conceitos diferentes abordados sobre os estoques, a configuração foi realizada através de um cálculo grosseiro para os dias que consideram políticas de cobertura que pode ser visto na Tabela 9. Os dias de cobertura acabam por variar a quantidade de estoque mínimo de acordo com cada período, pois variam junto com a demanda de 3 períodos de planejamento a frente (regra desenhada pelo Opcenter), diferentemente dos níveis de estoques desejados que possui um valor meta fixado (regra inerente à heurística implementada).

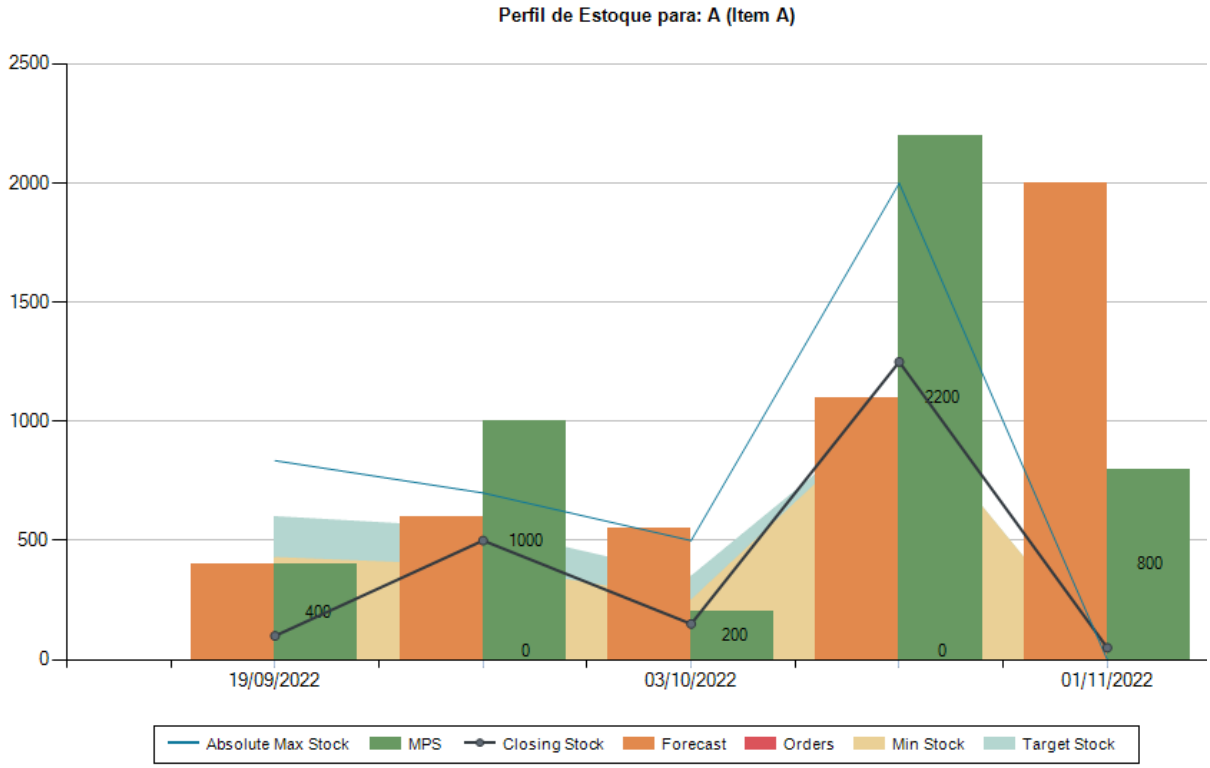
Tabela 9 – Configuração das políticas de cobertura

Item	Quantidade mínima de reordenamento	Quantidade de reordenamento	Dias de cobertura mínimo	Dias de cobertura alvo	Dias de cobertura máximo
A	200	200	5	7	10
B	200	200	5	7	10
C	200	200	5	7	10
D	200	200	5	7	10

Fonte: Autor (2022).

É possível concluir, da tabela 9, que o Opcenter tenta trabalhar dentro dos dias de cobertura buscando um valor alvo, colocando como objetivo um máximo e um mínimo de dias desejado. Mesmo que o Opcenter trabalhe com os dias mínimos de cobertura, nem sempre será possível atender o mínimo devido às limitações de capacidade, por exemplo. Este comportamento pode ser observado na Figura 10 quando, utilizando uma heurística nativa, o *Closing Stock* (Estoque final do período) fica abaixo do *Min Stock* (Estoque mínimo devido aos dias mínimos de cobertura).

Figura 10 – Perfil de estoque do Item A.



Fonte: Autor (2022).

Vale ressaltar, como análise da Figura 10, que o Opcenter, por trabalhar com dias de cobertura, não buscou atender os níveis de estoques mínimos no último período por não haver demanda futura, ou seja, não é necessário criar estoques para atender o que não está previsto ou vendido ainda.

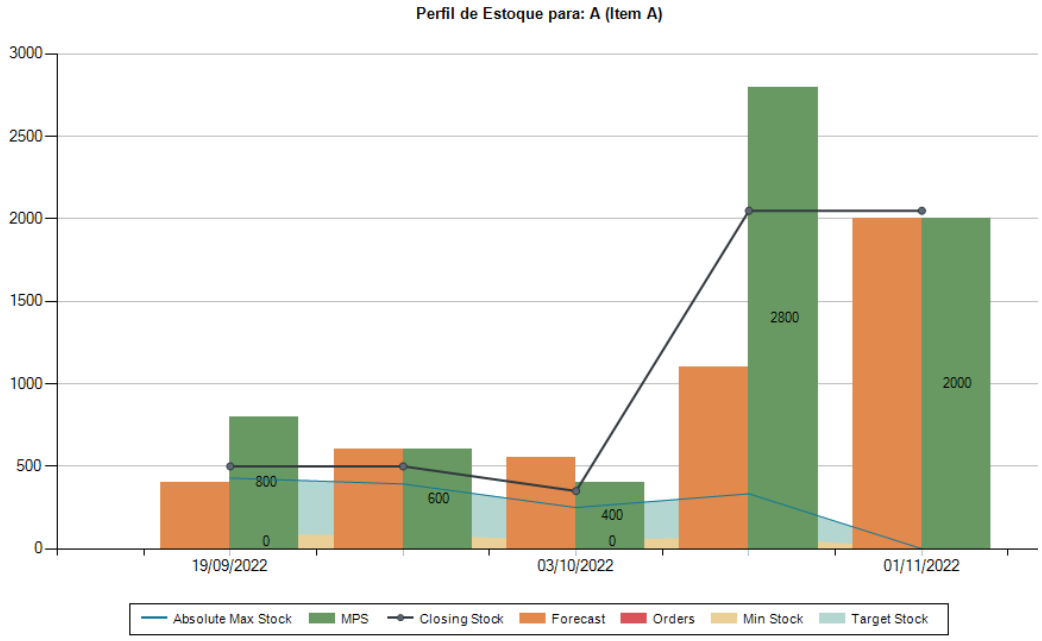
4.5.1 A comparação entre a heurística desenvolvida e a nativa do Opcenter

Dois grandes pontos serão analisados dentro deste tópico, o valor na saída de estoque final a cada período realizado e a capacidade utilizada disponível.

O primeiro ponto a ser analisado dentro das duas lógicas aplicadas é a qualidade no controle de estoques. Para este critério, apenas o item A será analisado por questões de repetitividade da comparação. Como demonstrado anteriormente na Figura 10, o resultado dado pela heurística do Opcenter não gerou nenhuma ruptura de estoque nos períodos analisados, porém teve fechamento de estoque abaixo do mínimo desejado e no último período não buscou atender possíveis demandas futuras. No lado da heurística implementada, é possível concluir da Figura 11 que os estoques de segurança pretendidos dentro das configurações são respeitados em quase todos os períodos, exceto na terceira semana, onde o estoque finaliza em 350, quando o desejado é 400

(Tabela 8) durante os períodos de planejamento semanais e 2000 nos períodos mensais. Além disso, a lógica desenvolvida buscou gerar um lote que atendesse os níveis de segurança do produto em questão. Percebe-se ainda que a linha de estoque final do período (*closing stock)* possui menor variação na heurística deste trabalho.

Figura 11 – Perfil de estoque do item A calculado pela heurística implementada



Fonte: Autor (2022).

É possível perceber que não houve ruptura de estoque no Item A em nenhum dos dois casos, no entanto, a heurística implementada gerou demanda não atendida na primeira e na segunda semana para os itens B e C (vide APÊNDICE B), respetivamente, enquanto a heurística nativa em momento nenhum teve o seu nível de serviço comprometido, gerando apenas estoques abaixo do desejado (Figura 12). Mesmo que as duas heurísticas tenham atendido de maneira diferente os níveis de estoques e dias de cobertura desejados, a qualidade de cada uma irá ser concretizada com os objetivos de uma indústria no momento de calcular o MPS.

Figura 12 – Níveis de estoque final do plano realizado com a heurística nativa

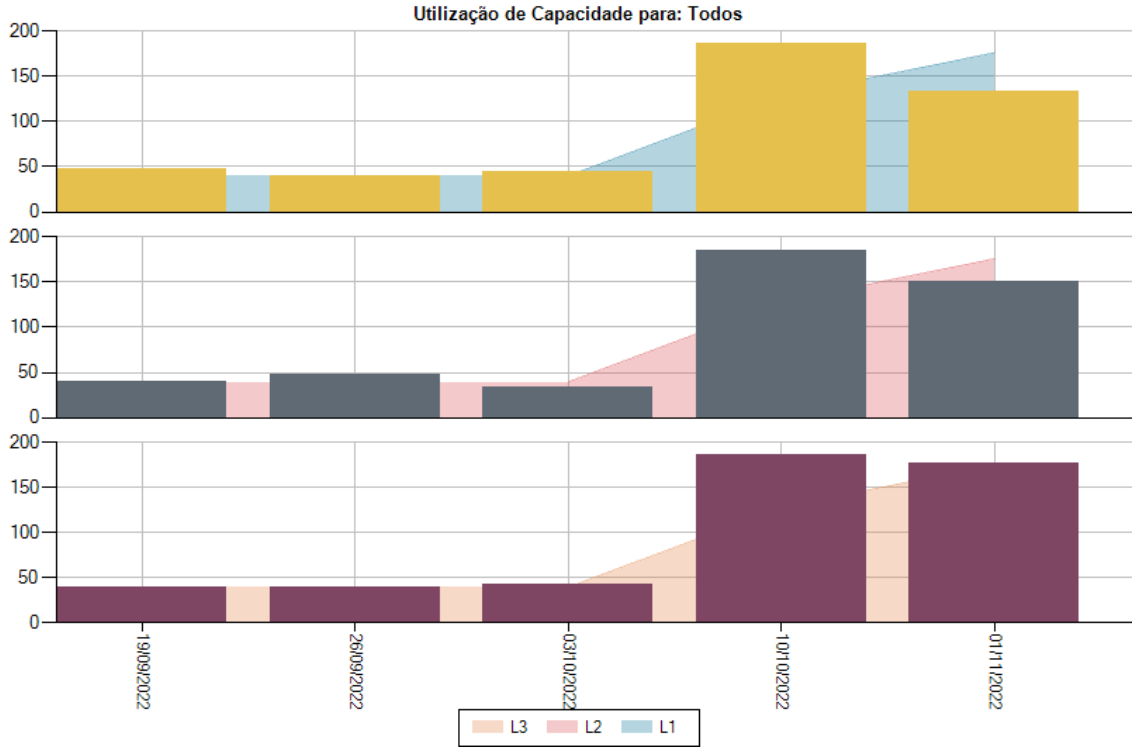
Produto	Data da Demanda	Estoque Inicial	Demanda	MPS	Recurso	Capacidade Usac	Estoque de Segurança	Estoque Final
Item C	01-11-2022	1200.00	2000.00	800.00	L1	53.33	400.00	0.00
Item D	01-11-2022	1200.00	2000.00	800.00	L2	80.00	300.00	0.00
Item B	26-09-2022	100.00	650.00	600.00	L1	30.00	500.00	50.00
Item D	26-09-2022	250.00	600.00	400.00	L2	40.00	300.00	50.00
Item A	01-11-2022	1250.00	2000.00	800.00	L1	53.33	400.00	50.00
Item A	19-09-2022	100.00	400.00	400.00	L1	26.67	400.00	100.00
Item B	19-09-2022	300.00	400.00	200.00	L1	10.00	500.00	100.00
Item C	26-09-2022	550.00	650.00	200.00	L1	13.33	400.00	100.00
Item C	03-10-2022	100.00	600.00	600.00	L1	40.00	400.00	100.00
Item D	03-10-2022	50.00	550.00	600.00	L2	60.00	300.00	100.00
Item A	03-10-2022	500.00	550.00	200.00	L1	13.33	400.00	150.00
Item B	01-11-2022	1150.00	2000.00	1000.00	L1	50.00	500.00	150.00
Item D	19-09-2022	350.00	500.00	400.00	L2	40.00	300.00	250.00
Item B	03-10-2022	50.00	600.00	0.00	L1	0.00	500.00	450.00
Item A	26-09-2022	100.00	600.00	0.00	L1	0.00	400.00	500.00
Item C	19-09-2022	250.00	500.00	0.00	L1	0.00	400.00	550.00
Item B	10-10-2022	450.00	1100.00	1800.00	L1	90.00	500.00	1150.00
Item C	10-10-2022	100.00	1100.00	1400.00	L1	93.33	400.00	1200.00
Item D	10-10-2022	100.00	1100.00	1800.00	L2	180.00	300.00	1200.00
Item A	10-10-2022	150.00	1100.00	0.00	L1	0.00	400.00	1250.00

Fonte: Autor (2022).

O Opcenter gerou cenários melhores do ponto de vista de estoques pois considerou dias de cobertura variáveis e apenas buscou gerar lotes a partir do mínimo necessário e atender a demanda, enquanto a heurística desenvolvida buscou gerar lotes para atender o nível de segurança somada da demanda do período. Embora, o Opcenter tenha dito desempenho melhor do que a heurística desenvolvida neste quesito, a ideia da implementação não era buscar ser concorrente na qualidade, mas sim permitir a flexibilidade como utilizar níveis de estoque.

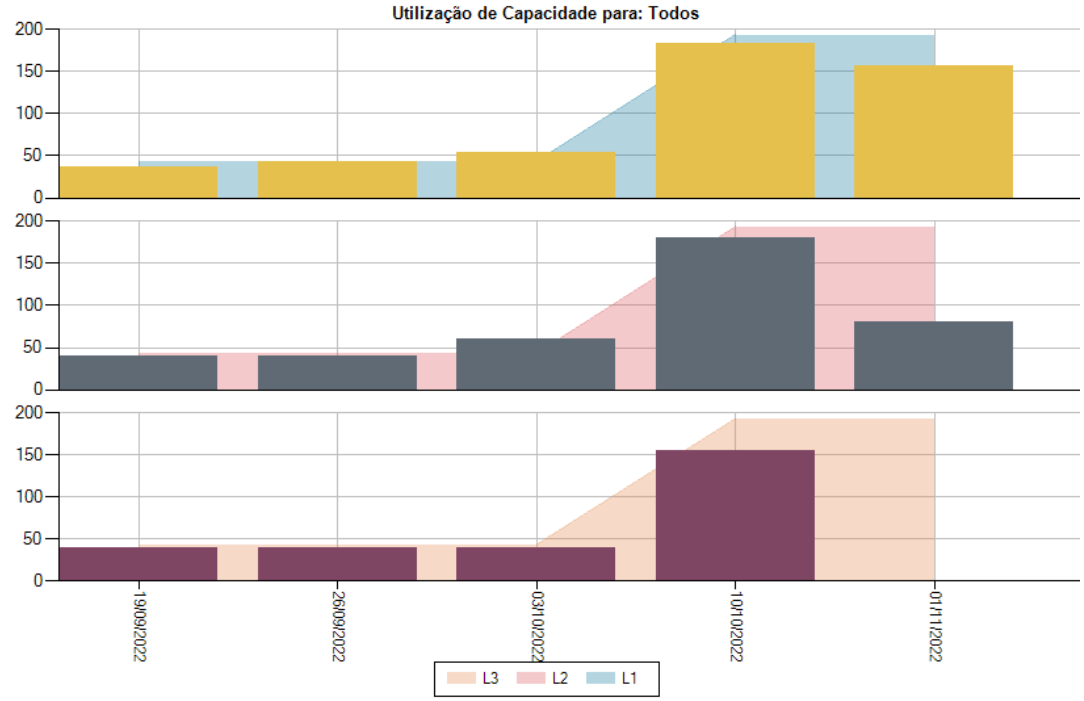
Para as análises de capacidade dos recursos, é possível comparar os dois cenários com as Figuras 13 e 14 e ter uma visão bruta sobre o uso geral de capacidade. Enquanto o Opcenter gerou um cenário mais otimista do ponto de vista de estoque, ao final dos últimos períodos, não utilizou a capacidade disponível (Figura 14), pelo lado da heurística aplicada (Figura 13) o uso da capacidade foi considerado plenamente em todos os níveis.

Figura 13 – Utilização de capacidade da heurística implementada



Fonte: Autor (2022).

Figura 14 – Utilização de capacidade da heurística nativa.



Fonte: Autor (2022).

Na Tabela 10 é possível perceber que os níveis de capacidade foram inteiramente respeitados dentro das limitações do conjunto de dados quando a saída de MPS é dada pela heurística implementada, por outro lado, quando o Opcenter realiza os cálculos considerando capacidade finita, acaba não respeitando os limites de 20% disponíveis de horas extras na modelagem, na semana 3, por exemplo, ficou com 133% de utilização em dois produtos.

A sobrecarga gerada nos recursos se pelo modelo de trabalho da heurística utilizada, o Opcenter não se encarrega de realizar cortes de produção sem parametrização realizada. O fluxo realizado é dado por quando há cortes devido à exceção de carga no recurso:

1. Criar MPS do produto no período.
2. Dentro do recurso de sobrecarga: Cortar lotes múltiplos de produção dinamicamente, ou seja, diminui ao menos um lote múltiplo do MPS original de um produto e passa para o próximo produto com o parâmetro de dias de cobertura alvo mais crítico.
3. Quando os cortes forem realizados até o lote mínimo e a capacidade ainda é excedida no recurso, o Opcenter não corta a produção do MPS sem uma pré-configuração.

Os parâmetros para realizar o corte de produção do produto no período não são disponibilizados pelo artigo do conjunto de dados Vieira e Favaretto (2006). Um exemplo de parametrização pode ser o corte de produção de um item C antes de um Item A de uma curva ABC.

A heurística implementada, por outro lado, possui uma configuração que olha pra disponibilidade do recurso e faz os cortes nos volumes de MPS quando acha necessário para adequar à capacidade disponível, impossibilitando que o usuário final indique o melhor cenário para cortes de produção. Concorda-se que a maneira que o Opcenter trabalha para gerar MPS, neste caso de sobrecarga de utilização, é mais interessante para as indústrias pois não é elegante que a heurística faça o corte sem um parâmetro ou uma permissão anterior. A maneira que a heurística implementada determina a não produção de um produto não fornece uma visibilidade sobre os produtos que precisaram ficar de fora do plano, mesmo em seus lotes mínimos de produção.

Por fim, o uso de capacidade dentro da heurística desenvolvida é mais respeitoso do ponto de vista de disponibilidade de condições, por outro lado o Opcenter entende que para que a capacidade realmente seja considerada, um parâmetro de corte deve ser definido para que este seja mais aderente às reais disponibilidades da fábrica.

Tabela 10 - Comparação de recursos

Período	Recurso	Capacidade usada	Utilização (%)	Fonte
Semana 2	L1	48	120%	HEURÍSTICA
Semana 2	L2	40	100%	HEURÍSTICA
Semana 2	L3	40	100%	HEURÍSTICA
Semana 3	L1	40	100%	HEURÍSTICA
Semana 3	L2	48	120%	HEURÍSTICA
Semana 3	L3	40	100%	HEURÍSTICA
Semana 4	L1	45	113%	HEURÍSTICA
Semana 4	L2	33,335	83%	HEURÍSTICA
Semana 4	L3	42,665	107%	HEURÍSTICA
Mês 2	L1	186,665	104%	HEURÍSTICA
Mês 2	L2	184	102%	HEURÍSTICA
Mês 2	L3	186,665	104%	HEURÍSTICA
Mês 3	L1	133,335	74%	HEURÍSTICA
Mês 3	L2	150	83%	HEURÍSTICA
Mês 3	L3	177,335	99%	HEURÍSTICA
Semana 2	L1	36,67	92%	OPCENTER
Semana 2	L2	36,67	92%	OPCENTER
Semana 2	L3	36,67	92%	OPCENTER
Semana 3	L1	43,33	108%	OPCENTER
Semana 3	L2	43,33	108%	OPCENTER
Semana 3	L3	43,33	108%	OPCENTER
Semana 4	L1	53,33	133%	OPCENTER
Semana 4	L2	53,33	133%	OPCENTER
Semana 4	L3	40,00	100%	OPCENTER
Mês 2	L1	183,33	108%	OPCENTER
Mês 2	L2	183,33	108%	OPCENTER
Mês 2	L3	183,33	108%	OPCENTER
Mês 3	L1	103,33	61%	OPCENTER
Mês 3	L2	103,33	61%	OPCENTER
Mês 3	L3	103,33	61%	OPCENTER

Fonte: Autor (2022).

4.6 VANTAGENS E DESVANTAGENS

Pelas naturezas de heurísticas utilizadas dentro das soluções do Opcenter e a que foi desenvolvida, os cálculos finais de MPS tendem a ser melhores pelo lado do Opcenter, pois, o Opcenter, pelo pouco que se sabe, utiliza conceitos de otimização combinados com heurísticas construtivas, além de ser a ferramenta mais vendida no mercado de APS, em contrapartida, a heurística desenvolvida que também tem seus lados positivos se enquadra como uma solução construtiva, sem nenhuma análise posterior ao resultado ou intenção de perturbar para gerar um melhoramento. Uma vez que já é uma ferramenta em uso no mercado, possui, dentre outras coisas, várias validações de fluxos e possibilidades de configurações via parametrização do *software*, os resultados nativos da ferramenta APS também tendem a ser melhores.

No entanto, pelas políticas de empresas de desenvolvimento de *software*, os códigos raramente ficam abertos para os usuários finais ou até mesmo representantes da empresa, isto acaba dificultando um entendimento claro do fluxo realizado e não possibilita de modo nenhum a flexibilização para indústrias via alteração no código fonte, ao contrário da solução desenvolvida, que pode, pelos próprios consultores que desejarem, flexibilizarem o código fonte e implementar novos métodos, melhorias ou até mesmo heurísticas de melhoramento para aprimorar o resultado.

Além disso, o Opcenter não tem, por configuração padrão, a possibilidade de criar MPS baseando-se em níveis de estoques de segurança, mínimos e máximos, foi necessário configurá-lo para que a API pudesse entender os campos na heurística desenvolvida, tendo como resultado, uma nova funcionalidade que não está inclusa nas versões disponibilizadas pela Siemens. O motivo pelo qual o Opcenter não realiza o MPS por cálculos baseados em estoque é porque utiliza conceitos do *Demand Driven Material Requirements Planning* (*DDMRP*), ou seja, trabalham com políticas de coberturas de estoque, em vez de basear-se em estoques mínimos. Existem lados positivos e negativos para as indústrias que buscam trabalhar com políticas de estoque ou não, porém não cabe no escopo do trabalho aprofundá-los.

Por fim, é possível também, dentro da nova solução, configurar maneiras diferentes para a criação do MPS através da flexibilização de código, garantido, por exemplo, uma adesão maior de indústrias no processo de transformação digital, também conhecido como a revolução da indústria 4.0.

5 CONCLUSÃO

Com todos os conceitos do PPCP revisados foi possível entender que existem diversos níveis de planejamento dentro das indústrias, ainda, compreendeu-se que os processos envolvidos podem ser complexos e variam de acordo com cada empresa. Um dos importantes processos é o MPS, que tem como intenção criar um plano de produção para cada produto em cada período de demanda futura e como visto. Desse mesmo modo, não foge das suas complexidades devido às configurações de chão de fábrica, principalmente devido aos recursos com suas capacidades produtivas limitadas.

Uma maneira de resolver a complexidade do cálculo de MPS está nas soluções por heurísticas, podendo elas serem ótimas ou não, dependendo da quantidade de parâmetros envolvidos. Com as heurísticas construtivas é possível obter soluções que buscam se aproximar do ótimo sem necessitar de um tempo considerável de computação para retornar um plano. Para isso, uma heurística de criação de MPS, proposta por Guilherme Ernani Vieira foi estudada para implementada dentro de uma ferramenta APS e foi possível entender o quão complexo é construir um Planejamento Mestre de Produção de um simples conjunto de dados de uma empresa fictícia.

Com a heurística em mãos, foi possível desenvolvê-la utilizando tecnologia e linguagem de programação disponível nos níveis de avanços tecnológicos. Além disso, existem uma série de ferramentas no mercado de soluções para indústrias que buscam auxiliar o processo de MPS. No entanto, como informado, os códigos fontes de ferramentas, como as APS, não são abertos para os usuários da ferramenta, o que dificulta o processo de implantação e entendimento das lógicas executadas pelo programa quando há déficit de documentação adequada, um cenário muito comum em indústrias de *software*.

A implementação da heurística dentro do Opcenter ocorreu com sucesso e respeitou todos os parâmetros e fluxos de atividades demonstrados no artigo. Houveram diferenças nos resultados entre a heurísticas nativa utilizada no Opcenter e a desenvolvida, no entanto, não prejudicou o fluxo proposto da heurística. A diferença de resultados se deu exclusivamente pela maneira de realizar os cortes nos lotes de produção. O Opcenter, por possuir um vasto uso nas indústrias hoje em dia, tende a ter resultados melhores, tanto por isso, mas também por possuir heurísticas combinadas

entre construtivas, melhorativas e meta-heurísticas. Porém, mesmo que o Opcenter tenha toda essa combinação, nenhum consultor deste *software* possui abertura para entender o correto fluxo de atividades e como a combinação ocorre. Com o fato da não abertura do código fonte, o Opcenter deixa a desejar na flexibilização por código fonte, sendo possível realizar apenas via API, banco de dados e parametrizações com configurações nativas.

A solução desenvolvida via programação na API agora permite que novos métodos sejam implementados e configurados na modelagem com possibilidade de abrir o código fonte e entender como o fluxo de tarefas é realizado, dando grande poder à consultores que desejarem implantar esta heurística em uma futura indústria. Existem algumas limitações na solução desenvolvida por conta de *hard-codes* (parâmetros não flexíveis fora do código fonte), são eles: capacidade de recursos, horas extras e lote mínimo de produção. Estes foram deixados como inflexíveis para atender o conjunto de dados do artigo citado durante todo o trabalho, mas não significam ser um problema de flexibilização no futuro, na verdade existe maior simplicidade para quem domina os conceitos de C#.

5.1 SUGESTÃO PARA FUTUROS TRABALHOS

Por fim, considerando todos os objetivos concluídos deste trabalho, os próximos passos para a continuação dele serão dados por aprimoramentos no código fonte da heurística implementada, podendo inclusive utilizar do seu próprio banco de dados no futuro, em vez de utilizar o do Opcenter.

Embora a quantidade de consultores que possuam o *know-how* de programação em C# e a API do Opcenter seja um número limitado e difícil de encontrar (TARNOWSKI, 2022), este trabalho também tem como objetivo incubado a intenção de encorajar os futuros estudantes a se desenvolverem tanto na área de Engenharia de Produção, como também entender como a tecnologia pode auxiliar nos processos complexos envolvidos desta área.

REFERÊNCIAS

BARBOSA, Eneias Santos; SANTOS, Maria Souza; NETO LOPES Verônica Maria. A Importância Do PCP (Planejamento e Controle da Produção) para a competitividade em indústrias de Juazeiro da Bahia**. Id on line Rev.Mult. Psic**., Outubro/2019, vol.13, n.47, p. 89-108. ISSN: 1981-1179.

CARVALHO, Victorio Albani de; TEIXEIRA, Giovany Frossard. **Programação Orientada a Objetos**: curso técnico em informática. Colatina: E-Tec/Mec, 2012. 134 p. Disponível em: http://redeetec.mec.gov.br/images/stories/pdf/eixo_infor_comun/tec_inf/081112_progr_obj.pdf. Acesso em: 24 nov. 2022.

CORRÊA, Henrique Luiz; GIANESI, Irineu Gustavo Nogueira; CAON, Mauro. **Planejamento, Programação e Controle da Produção**: conceitos, uso e implantação: base para SAP, oracle applications e outros softwares integrados de gestão. 6ª Edição, São Paulo, Atlas, 2017.

COSTA, Júlio César da. **Planejamento, programação e controle de produção**. Londrina: Editora e Distribuidora Educacional S.A, 2016.

GODINHO FILHO, Moacir; CAMPANINI, Luciano; VITA, Romano Augusto S. Guerra. A interação MRPII - CPM: estudo de caso e proposta de um sistema híbrido. **Production**, [S.L.], v. 14, n. 1, p. 31-43, 2004. FapUNIFESP (SciELO). http://dx.doi.org/10.1590/s0103-65132004000100004.

FUCHIGAMI, Hélio Yochihiro. **MÉTODOS HEURÍSTICOS CONSTRUTIVOS PARA O PROBLEMADE PROGRAMAÇÃO DA PRODUÇÃO EM SISTEMAS FLOW SHOP HÍBRIDOS COM TEMPOS DE PREPARAÇÃO DAS MÁQUINAS ASSIMÉTRICOS E DEPENDENTES DA SEQÜÊNCIA**. 2005. 135 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Departamento de Engenharia de Produção, Universidade de São Paulo, São Carlos, 2005.

Gaither, Norman. Frazier, Greg. **Administração da Produção e Operações.** .8. ed. São Paulo, Pioneira Thomson Learning, 2002.

Liddel, Mike. **Why Your ERP is NOT a Production Scheduler.** Disponível em: <https://lean-scheduling.com/why-your-erp-is-not-a-production-scheduler/ >. Acesso: 25/11/2022.

MASTER PRODUCTION SCHEDULING. *In*: APICS, **Dictionary**. Chicago: APICS, 2013. Disponível em: https://www.ascm.org/learning-development/certifications-credentials/dictionary/. Acesso em 10/08/2022.

MAUERGAUZ, Yuri. **Advanced Planning and Scheduling in Manufacturing and Supply Chains**. Moscow: Springer International Publishing Switzerland, 2016.

Nordin, Nurul Nadia. Lee, Lai Soon. Heuristics and metaheuristics approaches for facility layout problems: a survey. **Pertanika Journal of Scholarly Research Reviews** p. 62-76, 2016.

PEREIRA, Ana Amélia de Souza. **METAHEURÍSTICAS PARA O PROBLEMA DE FLOWSHOP FLEXÍVEL COM PENALIDADES DE ADIANTAMENTO E ATRASO**. 2011. 57 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade Federal de Viçosa, Viçosa, 2011.

TARNOWSKI, Carlos Eduardo Romancini. **Avaliação de desempenho da heurística busca local dirigida na minimização de setup em empresa do segmento de plástico flexível**. 2022. 72f. TCC (Graduação) - Curso de Produção Civil, Universidade Federal de Santa Catarina, Florianópolis, 2022.

TUBINO, Dalvio Ferrari. **Planejamento e Controle da Produção**: teoria e prática. 3ª Edição, São Paulo, Atlas, 2017.

TURRIONI, João Batista *et al.* **Metodologia de pesquisa em engenharia de produção**: estratégias, métodos e técnicas para condução de pesquisas quantitativas e qualitativas. 3. ed. Rio de Janeiro: Elsevier Editora Ltda., 2011.

VIEIRA, Guilherme Ernani; FAVARETTO, Fabio. A new and practical heuristic for Master Production Scheduling creation. **International Journal ff Production Research**. Curitiba, p. 3607-3625. out. 2006.

VIEIRA, Guilherme Ernani; RIBAS, Paulo Cesar. A NEW OPTIMIZATION METHOD FOR PRODUCTION PLANNING PROBLEMS USING SIMULATED ANNEALING. In: CONGRESSO INTERNACIONAL DE ENGENHARIA MECÂNICA, 17., 2003, São Paulo. **Proceedings [...]** . São Paulo: ABCM, 2003. p. 4609-4622.

VIEIRA, Guilherme E.; SOARES, Marcio M.. A new multi-objective optimization method for master production scheduling problems based on genetic algorithm. **The International Journal of Advanced Manufacturing Technology**. [S.I.], p. 549-567. mar. 2008.

VOLLMANN, Thomas E. *et al.* **Manufacturing Planning and Control Systems for Supply Chain Management**. 5. ed. S.I: McGraw-Hill, 2005.

PROUD, John F. **Master Scheduling**. 3. ed. New Jersey: John Wiley & Sons, Inc., 2007.

SLACK, Nigel *et al.* **Administração da Produção**. São Paulo: Atlas S.A, 1996.

ZAGO, Cecília Farid. **IMPLANTAÇÃO DE SISTEMAS AVANÇADOS DE PLANEJAMENTO (APS)**: um estudo de caso na indústria de laticínios. 2013. 120 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Departamento de Engenharia de Produção, Escola Politécnica da Universidade de São Paulo, São Paulo, 2013.

APÊNDICE A – CÓDIGO DA HEURÍSTICA IMPLEMENTADA

Projeto de solução MPSHeuristic:

```
using System;
using System.Runtime.InteropServices;
using Preactor;
using Preactor.Interop.PreactorObject;
using System.Windows.Forms;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using static Declarations;
using Microsoft.Office.Interop.Excel;

namespace OpcenterMPSHeuristic
{
   [Guid("767aaf60-acee-4114-a2ac-3d48db672fc5")]
   [ComVisible(true)]
   public interface IMPSHeuristic
   {

      int genMPS(ref PreactorObj preactorComObject, ref object pespComObject);

   }

   [ComVisible(true)]
   [ClassInterface(ClassInterfaceType.None)]
   [Guid("8375df0c-6d42-41a6-b03d-08cdbcf8bd7d")]
   public class MPSHeuristic : IMPSHeuristic
   {
      IList<Item> ItemsList = new List<Item>();
      IList<Demand> DatesList = new List<Demand>();
      IList<Stock> StockList = new List<Stock>();
      IList<Resource> ResourceList = new List<Resource>();
      IList<Demand> DemandList = new List<Demand>();
      IList<Item> ItemsResourceCount = new List<Item>();
      IList<Demand> MPSList = new List<Demand>();
      IList<MPSExport> MPSExportList = new List<MPSExport>();
      IList<ItemsResource> ItemsResourceData = new List<ItemsResource>();

      public int genMPS(ref PreactorObj preactorComObject, ref object
pespComObject)
      {
         sharedPreactor = PreactorFactory.CreatePreactorObject(preactorComObject);

         ItemsList.Clear();
         DatesList.Clear();
         StockList.Clear();
```

```
            ResourceList.Clear();
            DemandList.Clear();
            ItemsResourceCount.Clear();
            MPSList.Clear();
            MPSExportList.Clear();
            ItemsResourceData.Clear();
            getItems();
            getPlanningResources();
            getItemsResourceData();
            runMPS();
            sharedPreactor.Planner.RefreshPlannerGrid();
            //exportData();
            return 0;
        }


        public void runMPS()
        {
            var dates = DatesList.Select(x => x.DemandDate).Distinct();

            foreach (var date in dates)
            {
                calculateNetRequirements(date);
                createGridControl(date);
                calculateMPS(date);

            }
        }

        // calcula as necessidades líquidas para iniciar o processo de criação de MPS
        public int calculateNetRequirements(DateTime date)
        {
            // Net Req = Min(Mult(Min(Max([gross - (initial inv + subcont)],0), min lot
size), standard lot size), max inv level)
            //calcula os estoques para abrir o initial inventory
            sharedPreactor.Planner.CalculateStock();
            string lastCode = null;
            DateTime lastDate = sharedPreactor.ReadFieldDateTime(tblDemand,
clnDemandDate, 1);
            int demandLength = sharedPreactor.RecordCount(tblDemand);
            for (int i = 1; i <= demandLength; i++)
            {
                string currentCode = sharedPreactor.ReadFieldString(tblDemand,
clnDemandCode, i);
                DateTime currentDate = sharedPreactor.ReadFieldDateTime(tblDemand,
clnDemandDate, i);
                if (currentCode != lastCode && currentDate == date)
                {
```

```
                double initialInventory = sharedPreactor.ReadFieldDouble(tblDemand,
clnDemandOpeningStock, i);
                initialInventory = Math.Max(initialInventory, 0);
                if (initialInventory >= 0)
                {
                   double subcontracted = sharedPreactor.ReadFieldDouble(tblDemand,
clnDemandSubcontracted, i);
                   double grossRequirements =
sharedPreactor.ReadFieldDouble(tblDemand, clnDemandDemand, i);
                   double minimumLotSize = sharedPreactor.ReadFieldDouble(tblDemand,
clnDemandMinimumLotSize, i);
                   double standardLotSize = sharedPreactor.ReadFieldDouble(tblDemand,
clnDemandReorderMultiple, i);
                   double maximumInventoryLevel =
sharedPreactor.ReadFieldDouble(tblDemand, clndDemandMaximumInventoryLevel, i);
                   double safetyInventory;
                   string currentDay = sharedPreactor.ReadFieldString(tblDemand,
clnDemandDay, i);
                   if (currentDay == "Week")
                   {
                      safetyInventory = sharedPreactor.ReadFieldDouble(tblDemand,
clnDemandSafetyInventory, i);
                   }
                   else
                      safetyInventory = 2000;
                   double var0 = safetyInventory + grossRequirements - initialInventory -
subcontracted;
                   double val1 = Math.Max(var0, 0);
                   double val2 = Math.Max(val1, minimumLotSize);
                   double val3 = Math.Ceiling(val2 / standardLotSize) * standardLotSize;
                   double val4 = Math.Min(val3, maximumInventoryLevel);
                   double netRequirements = val4;

                   FormatFieldPair demandNetRequirements = new
FormatFieldPair(sharedPreactor.GetFormatNumber(tblDemand),
sharedPreactor.GetFieldNumber(tblDemand, clnDemandNetRequirements));
                   sharedPreactor.WriteField(demandNetRequirements, i,
netRequirements);
                }

                lastCode = currentCode;
             }

          }
          sharedPreactor.Planner.RefreshPlannerGrid();
          return 0;
       }
```

```
public int getItemsResourceCount(string itemCode)
{
   int itemLength = sharedPreactor.RecordCount(tblItem);
   ItemsResourceCount.Clear();
   int count = 0;
   for (int i = 1; i <= itemLength; i ++)
   {
      MatrixDimensions size = sharedPreactor.MatrixFieldSize(tblItem,
clnItemsPlanningResourceData, i);
      Item Item = new Item();
      Item.ItemCode = sharedPreactor.ReadFieldString(tblItem, clnItemsItemCode,
i);
      if (itemCode == Item.ItemCode.ToString())
      {
         count = size.X;
      }
      Item.ResourceCount = size.X;
      ItemsResourceCount.Add(Item);

   }
   ItemsResourceCount = ItemsResourceCount.OrderBy(x =>
x.ResourceCount).ToList();

   return count;
}




public int getNonAggDemand()
{

   int DemandLength = sharedPreactor.RecordCount(tblDemand);
   try
   {
      for (int i = 1; i <= DemandLength; i++)
      {
         Demand Demand = new Demand();
         Demand.ItemCode = sharedPreactor.ReadFieldString(tblDemand,
clnDemandCode, i);
         Demand.DemandDate = sharedPreactor.ReadFieldDateTime(tblDemand,
clnDemandDate, i);
         Demand.GrossRequirements =
sharedPreactor.ReadFieldDouble(tblDemand, clnDemandDemand, i);
         DatesList.Add(Demand);
      }
```

```
            //MessageBox.Show("Sucesso!");
        }
        catch
        {
            MessageBox.Show("Erro!");
        }

        return 0;
    }



    public int getPlanningResources()
    {
        getNonAggDemand();

        int ResourcesLength = sharedPreactor.RecordCount(tblPlanningResources);

        var dates = DatesList.Select(x => x.DemandDate).Distinct();
        for (int i = 1; i <= ResourcesLength; i++)
        {
            foreach (var date in dates)
            {

                Resource Resource = new Resource();
                Resource.ResourceName =
sharedPreactor.ReadFieldString(tblPlanningResources, clnResourceName, i);
                Resource.AvailableCapacityPeriodInWeek = 40;
                Resource.AvailableCapacityPeriodInMonth = 170;
                Resource.OriginalCapacityPeriodInWeek = 40;
                Resource.OriginalCapacityPeriodInMonth = 170;
                Resource.OvertimePercent = 20;
                Resource.DatePeriod = date;
                ResourceList.Add(Resource);
            }

        }

        return 0;

    }


    public double getResourceAvaliableCapacity(string resource, DateTime date,
string day)
    {
        int resourcesLength = ResourceList.Count();
        double availableCapacity = 0;
        for (int i = 0; i < resourcesLength; i++)
```

```
        {
            if (ResourceList[i].ResourceName == resource &&
ResourceList[i].DatePeriod == date)
            {
                if (day == "Week")
                {
                    availableCapacity = ResourceList[i].AvailableCapacityPeriodInWeek;
                    break;
                }
                else
                {
                    availableCapacity = ResourceList[i].AvailableCapacityPeriodInMonth;
                    break;
                }

            }
        }
        return availableCapacity;
    }

    public void setResourceStats(string resource, DateTime date, double newCapacity,
string day)
    {
        int resourcesLength = ResourceList.Count();

        for (int i = 0; i < resourcesLength; i++)
        {
            if (ResourceList[i].ResourceName == resource &&
ResourceList[i].DatePeriod == date)
            {
                if (day == "Week")
                {
                    ResourceList[i].AvailableCapacityPeriodInWeek = newCapacity;
                    break;
                }
                else
                {
                    ResourceList[i].AvailableCapacityPeriodInMonth = newCapacity;
                    break;
                }

            }
        }

    }

    public void setDemandFit(string itemCode, DateTime date, int fitsInFull, int
fitsInPartial = 0)
```

```
    {
        int demandLength = DemandList.Count();
        for(int i = 0; i < demandLength; i++)
        {
            if(DemandList[i].ItemCode == itemCode && DemandList[i].DemandDate ==
date)
            {
                DemandList[i].ItemFitsInFull = fitsInFull;
                DemandList[i].ItemFitsInPartial = fitsInPartial;
            }
        }
    }


    public void getItemsResourceData()
    {
        int itemsLength = sharedPreactor.RecordCount(tblItem);

        for (int i = 1; i <= itemsLength; i++)
        {

            MatrixDimensions size = sharedPreactor.MatrixFieldSize(tblItem,
clnItemsPlanningResourceData, i);
            for (int j = 1; j <= size.X; j++)
            {
                ItemsResource ItemResources = new ItemsResource();
                ItemResources.ItemCode = sharedPreactor.ReadFieldString(tblItem,
clnItemsItemCode, i);
                ItemResources.ResourceCode = sharedPreactor.ReadFieldString(tblItem,
clnItemsPlanningResourceData, i, j);
                ItemResources.RateperHour = sharedPreactor.ReadFieldDouble(tblItem,
clnItemsResSpecificRateperHour, i, j);
                ItemsResourceData.Add(ItemResources);
            }
        }
    }
    public double getResourceRate(string itemCode, string resource)
    {
        int resourceDataLength = ItemsResourceData.Count();
        double rate = 0;
        for (int i = 0; i < resourceDataLength; i++)
        {
            if(ItemsResourceData[i].ItemCode == itemCode &&
ItemsResourceData[i].ResourceCode == resource)
            {
                rate = ItemsResourceData[i].RateperHour;
                break;
            }
```

```
        }
        return rate;
    }

    public double getOriginalCapacity(string resource, DateTime date, string day)
    {
        double originalCapaity;
        if (day == "Week")
        {
            originalCapaity = ResourceList.ToList().Find(r => (r.ResourceName ==
resource) && (r.DatePeriod.Equals(date))).OriginalCapacityPeriodInWeek;
        }

        else
            originalCapaity = ResourceList.ToList().Find(r => (r.ResourceName ==
resource) && (r.DatePeriod.Equals(date))).OriginalCapacityPeriodInMonth;

        return originalCapaity;
    }

    public int createGridControl(DateTime date)
    {
        //DemandList.Clear();
        int demandLength = sharedPreactor.RecordCount(tblDemand);
        for (int i = 1; i <= demandLength; i++)
        {
            Demand Demand = new Demand();
            Demand.NetRequirements = sharedPreactor.ReadFieldDouble(tblDemand,
clnDemandNetRequirements, i);
            Demand.DemandDate = sharedPreactor.ReadFieldDateTime(tblDemand,
clnDemandDate, i);
            if (Demand.NetRequirements >= 0 && Demand.DemandDate == date)
            {
                Demand.Number = sharedPreactor.ReadFieldInt(tblDemand,
clnDemandNumber, i);
                Demand.ItemCode = sharedPreactor.ReadFieldString(tblDemand,
clnDemandCode, i);
                Demand.DemandDate = sharedPreactor.ReadFieldDateTime(tblDemand,
clnDemandDate, i);
                Demand.Resource = sharedPreactor.ReadFieldString(tblDemand,
clnDemandPlanningResource, i);
                Demand.CapacityUsed = sharedPreactor.ReadFieldDouble(tblDemand,
clnDemandCapacityUsed, i);
                Demand.ResourceCount =
getItemsResourceCount(Demand.ItemCode.ToString());
                Demand.Day = sharedPreactor.ReadFieldString(tblDemand,
clnDemandDay, i);
                Demand.ItemFitsInFull = 0;
```

```
                Demand.ItemFitsInPartial = 0;
                DemandList.Add(Demand);
            }
            DemandList = DemandList.OrderBy(x => x.DemandDate).ThenBy(c =>
c.ResourceCount).ThenByDescending(n => n.NetRequirements).ToList();
        }
            return 0;
    }

    public string chooseResource(string itemCode, DateTime date, string day)
    {
        var resources = from resourceData in ItemsResourceData
                        where resourceData.ItemCode == itemCode
                        select resourceData;
        string choosenResource = "";
        double auxMPS = 0;
        foreach(var resource in resources)
        {
            string currentResource = resource.ResourceCode;
            double currentRate = resource.RateperHour;
            double currentAvailableCapacity =
getResourceAvaliableCapacity(currentResource, date, day);
            double possibleMPS = currentRate * currentAvailableCapacity;
            if (auxMPS <= possibleMPS)
            {
                auxMPS = possibleMPS;
                choosenResource = currentResource;
            }
        }

        return choosenResource;
    }


    //public int getCurrentStock()
    //{



    //    int StockLength = sharedPreactor.RecordCount(tblStock);



    //    for (int i = 1; i <= StockLength; i++)
    //    {
    //        Stock Stock = new Stock();
    //        //Stock.Type = preactor.ReadFieldString(tblStock, "Type", i);
    //        Stock.ItemCode = sharedPreactor.ReadFieldString(tblStock,
clnStockItemCode, i);
    //        Stock.ProdnDate = sharedPreactor.ReadFieldDateTime(tblStock,
clnStockProdnDate, i);
```

```
        //        Stock.Qty = sharedPreactor.ReadFieldDouble(tblStock, clnStockQuantity,
i);
        //        StockList.Add(Stock);
        //    }

        //    return 0;

        //}



        public int getItems()
        {
            int ItemsLength = sharedPreactor.RecordCount(tblItem);
            for (int i = 1; i <= ItemsLength; i++)
            {
                sharedPreactor.ReadFieldString(tblItem, clnItemsItemLevel, i);
                if (sharedPreactor.ReadFieldString(tblItem, clnItemsItemLevel, i) ==
"Finished Product")
                {
                    Item Item = new Item();
                    Item.ItemCode = sharedPreactor.ReadFieldString(tblItem,
clnItemsItemCode, i);
                    Item.ItemDesc = sharedPreactor.ReadFieldString(tblItem,
clnItemsItemDesc, i);
                    Item.ItemLevel = sharedPreactor.ReadFieldString(tblItem,
clnItemsItemLevel, i);
                    Item.CapacityUoM = sharedPreactor.ReadFieldString(tblItem,
clnItemsCapacityUoM, i);
                    Item.PlanningResourceGroup = sharedPreactor.ReadFieldString(tblItem,
clnItemsPlanningResourceGroup, i);
                    Item.ReorderMultiple = sharedPreactor.ReadFieldDouble(tblItem,
clnItemsReorderMultiple, i);
                    Item.MinimumReorderMultiple =
sharedPreactor.ReadFieldDouble(tblItem, clnItemsMinimumReorderMultiple, i);
                    Item.MinimumCoverDays = sharedPreactor.ReadFieldDouble(tblItem,
clnItemsMinimumCoverDays, i);
                    Item.TargetCoverDays = sharedPreactor.ReadFieldDouble(tblItem,
clnItemsTargetCoverDays, i);
                    Item.MaximumCoverDays = sharedPreactor.ReadFieldDouble(tblItem,
clnItemsMaximumCoverDays, i);
                    ItemsList.Add(Item);
                }
            }

            return 0;
        }
```

```
        //Funcao de exportar dados abaixo.

        public int readExportData()
        {
            for(int i = 1; i <= sharedPreactor.RecordCount(tblDemand); i++)
            {
                MPSExport MPSExportLine = new MPSExport();
                MPSExportLine.ItemCode = sharedPreactor.ReadFieldString(tblDemand,
clnDemandCode, i);
                MPSExportLine.CapacityUsed =
sharedPreactor.ReadFieldDouble(tblDemand, clnDemandCapacityUsed, i);
                MPSExportLine.Period = sharedPreactor.ReadFieldString(tblDemand,
clnDemandDate, i);
                MPSExportLine.InitialInventory =
sharedPreactor.ReadFieldDouble(tblDemand, clnDemandOpeningStock, i);
                MPSExportLine.GrossRequirements =
sharedPreactor.ReadFieldDouble(tblDemand, clnDemandDemand, i);
                MPSExportLine.MPS = sharedPreactor.ReadFieldDouble(tblDemand,
clnDemandMPS, i);
                MPSExportLine.PlanningResource =
sharedPreactor.ReadFieldString(tblDemand, clnDemandPlanningResource, i);
                MPSExportLine.TargetStock = sharedPreactor.ReadFieldDouble(tblDemand,
clnDemandTargetStock, i);
                MPSExportLine.ClosingStock =
sharedPreactor.ReadFieldDouble(tblDemand, clnDemandClosingStock, i);
                MPSExportLine.MinStock = sharedPreactor.ReadFieldDouble(tblDemand,
clnDemandMinStock, i);
                MPSExportLine.MinDaysCover =
sharedPreactor.ReadFieldDouble(tblDemand, clnDemandMinDaysCover, i);
                MPSExportLine.TargetDaysCover =
sharedPreactor.ReadFieldDouble(tblDemand, clnDemandTargetDaysCover, i);
                MPSExportLine.TotalDaysCover =
sharedPreactor.ReadFieldDouble(tblDemand, clnDemandTotalDaysCover, i);
                MPSExportList.Add(MPSExportLine);
            }

            return 0;
        }

        public void exportData()
        {
            System.Threading.Thread.CurrentThread.CurrentCulture =
System.Globalization.CultureInfo.CreateSpecificCulture("en-US");
            readExportData();
            using (SaveFileDialog sfd = new SaveFileDialog() { Filter = "Excel
Workbook|*.xls", ValidateNames = true })
```

```
        {
            Microsoft.Office.Interop.Excel.Application app = new
Microsoft.Office.Interop.Excel.Application();
            Workbook wb = app.Workbooks.Add(XlSheetType.xlWorksheet);
            Worksheet ws = (Worksheet)app.ActiveSheet;
            app.Visible = false;
            List<string> props = typeof(MPSExport).GetProperties().Select(f =>
f.Name).ToList();
            int i = 1;
            foreach(var prop in props)
            {
               ws.Cells[1, i] = prop;

               i++;

            }
            int j = 2;
            foreach (MPSExport MPSExport in MPSExportList)
            {
               ws.Cells[j, 1] = MPSExport.ItemCode.ToString();
               ws.Cells[j, 2] = MPSExport.CapacityUsed.ToString();
               ws.Cells[j, 3] = MPSExport.Period.ToString();
               ws.Cells[j, 4] = MPSExport.OnHand.ToString();
               ws.Cells[j, 5] = MPSExport.InitialInventory.ToString();
               ws.Cells[j, 6] = MPSExport.GrossRequirements.ToString();
               ws.Cells[j, 7] = MPSExport.StandardLotSize.ToString();
               ws.Cells[j, 8] = MPSExport.NetRequirements.ToString();
               ws.Cells[j, 9] = MPSExport.MPS.ToString();
               ws.Cells[j, 10] = MPSExport.TargetStock.ToString();
               ws.Cells[j, 11] = MPSExport.MinStock.ToString();
               ws.Cells[j, 12] = MPSExport.ClosingStock.ToString();
               ws.Cells[j, 13] = MPSExport.MinDaysCover.ToString();
               ws.Cells[j, 14] = MPSExport.TargetDaysCover.ToString();
               ws.Cells[j, 15] = MPSExport.TotalDaysCover.ToString();
               ws.Cells[j, 16] = MPSExport.PlanningResource.ToString();
               ws.Cells[j, 17] = MPSExport.RequirementsMet.ToString();
               ws.Cells[j, 18] = MPSExport.RequirementsNotMet.ToString();
               ws.Cells[j, 19] = MPSExport.ServiceLevel.ToString();
               ws.Cells[j, 20] = MPSExport.AverageServiceLevelPeriod.ToString();
               ws.Cells[j, 21] = MPSExport.EndingInventory.ToString();
               ws.Cells[j, 22] = MPSExport.AverageInventoryPeriod.ToString();
               ws.Cells[j, 23] = MPSExport.TotalAverageInventoryPeriod.ToString();
               ws.Cells[j, 24] = MPSExport.BelowSafetyInvetory.ToString();
               ws.Cells[j, 25] = MPSExport.BelowSafetyInvetoryPeriod.ToString();
               j++;
            }
```

```
            wb.SaveAs("MPSFile", XlFileFormat.xlWorkbookDefault, Type.Missing,
Type.Missing, true, false, XlSaveAsAccessMode.xlNoChange,
XlSaveConflictResolution.xlLocalSessionChanges, Type.Missing, Type.Missing);
            app.Quit();
            MessageBox.Show("Seu arquivo foi exportado com sucesso!", "Mensagem");
          }
       }


}
}
```

APÊNDICE B – RESULTADOS TOTAIS DA HEURÍSTICA IMPLEMENTADA

	Semana 2					Semana 3					Semana 4					Mês 2					Mês 3				
	A	B	C	D	Capacidade utilizada	A	B	C	D	Capacidade utilizada	A	B	C	D	Capacidade utilizada	A	B	C	D	Capacidade utilizada	A	B	C	D	Capacidade utilizada
Em estoque	100	300	250	350																					
Estoque inicial	100	300	250	350		500	0	470	450		500	210	0	450		350	510	220	300		2050	1090	2120	2000	
(período)																									
Necessidades grosseiras	400	400	500	500		600	650	650	600		550	600	600	550		1100	1100	1100	1100		2000	2000	2000	2000	
Lote padrão	200	200	200	200		200	200	200	200		200	200	200	200		200	200	200	200		200	200	200	200	
Estoque de segurança	400	500	400	300		400	500	400	300		400	500	400	300		400	500	400	300		400	500	400	300	
Necessidades líquidas	800	600	800	600		600	1200	600	600		600	900	1000	400		2800	2600	3000	2800		2000	3000	2000	2000	
MPS																									
L1			720		48	600				40	900				45	2800				186,67	2000				133,33
L2	800				40		960			48		1000			33,33		1680	3000		184		3000			150
L3			600		40			600		40	400		400		42,67			2800		186			880	2000	147
Total	800	0	720	600		600	960	0	600		400	900	1000	400		2800	1680	3000	2800		2000	3000	880	2000	
Demanda atendida	400	300	500	500		600	650	470	600		550	600	600	550		1100	1100	1100	1100		2000	2000	2000	2000	
Demanda não atendida	0	100	0	0		0	0	180	0		0	0	0	0		0	0	0	0		0	0	0	0	
Nível de serviço	1	0,75	1	1		1	1	0,72	1		1	1	1	1		1	1	1	1		1	1	1	1	
Média do nível de serviço no período					0,94					0,93					1,00					1,00					1,00
Estoque final	500	0	470	450		500	210	0	450		350	510	220	300		2050	1090	2120	2000		0	0	0	0	
Média do estoque no período	300	150	360	400		500	105	235	450		425	360	110	375		1200	800	1170	1150		1025	545	1060	1000	
Média total do estoque no período					1210					1290					1270					4320					3630
Abaixo do estoque de segurança	0	500	0	0		0	290	400	0		50	0	180	0		0	0	0	0		400	500	400	300	
Abaixo do estoque de segurança no período					500					690					230					0					1600