UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

AMANDA THAIS ROCHA

APLICAÇÃO DE UMA HEURÍSTICA NO SEQUENCIAMENTO DE PRODUÇÃO DE UMA INDÚSTRIA DE PEÇAS DE AÇO

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA
2017

AMANDA THAIS ROCHA

APLICAÇÃO DE UMA HEURÍSTICA NO SEQUENCIAMENTO DE PRODUÇÃO DE UMA INDÚSTRIA DE PEÇAS DE AÇO

Trabalho de conclusão de curso apresentado ao curso de graduação em Engenharia de Produção da Universidade Tecnológica Federal do Paraná – Campus Ponta Grossa, como requisito parcial para obtenção do título de Bacharel em Engenharia de Produção.

Orientador: Prof. Dr. João Carlos Colmenero

PONTA GROSSA
2017

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS PONTA GROSSA
Departamento Acadêmico de Engenharia de Produção

TERMO DE APROVAÇÃO DE TCC

APLICAÇÃO DE UMA HEURÍSTICA NO SEQUENCIAMENTO DE PRODUÇÃO DE UMA INDÚSTRIA DE PEÇAS DE AÇO

por

Amanda Thais Rocha

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 28 de junho de 2017 como requisito parcial para a obtenção do título de Bacharel em Engenharia de Produção. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. **João Carlos Colmenero**
Orientador

_____
Prof. Dr. **Everton Luiz de Melo**
Membro titular

_____
_____
Prof. Dr. **Fábio Neves Puglieri**
Membro titular

"A Folha de Aprovação assinada encontra-se na Coordenação do Curso".

AGRADECIMENTOS

Agradeço, primeiramente, a Deus que me deu força e coragem nessa caminhada, por ter me dado as oportunidades que me fizeram chegar até aqui e concluído esse trabalho e esse curso de graduação.

A minha família, principalmente meus pais, Jane e Edson, que me deram todo apoio emocional e financeiro, que não mediram esforços para que eu chegasse até aqui. Também agradeço ao meu irmão Victor, por toda ajuda, encorajamento e conselhos durante essa etapa.

Ao meu orientador Prof. Dr. João Carlos Colmenero por ter me recebido de braços abertos, pela disponibilidade e paciência durante a execução deste trabalho. Também agradeço aos professores Dr. Everton Luiz de Melo e Dr. Fabio Jose Ceron Branco, pelos conhecimentos passados em suas respectivas disciplinas, que auxiliaram o interesse em realizar esse projeto e também pelas dicas que deram a respeito do trabalho.

A Universidade Tecnológica Federal do Paraná, seu corpo docente, direção e administração por terem me oferecido conhecimento, através de educação de qualidade e compromissada com seus alunos.

E a todos, que diretamente ou indiretamente, auxiliaram na execução deste trabalho e/ou fizeram parte da minha graduação, o meu muito obrigada.

ROCHA, A. T. Aplicação de uma heurística no sequenciamento de produção de uma indústria de peças de aço. 2017. Monografia (Bacharelado em Engenharia de Produção) – Universidade Tecnológica Federal do Paraná.

RESUMO

Este trabalho tem seu foco na aplicação de uma heurística de melhoria, a partir de uma solução inicial. O objetivo é encontrar um *makespan* muito bom e em tempo hábil para um ambiente de máquinas denominado máquinas paralelas não-relacionadas, será dado o exemplo de aplicação desse tipo de ambiente de máquinas, que é em uma empresa do ramo de fabricação de peças industriais sob encomenda. Esta indústria trabalha basicamente com corte, dobra, solda e pintura de chapas de aço, fazendo assim, peças de aço. As encomendas deste tipo de empresa são bastante dinâmicas, e a ocupação de determinadas máquinas é grande, necessitando de otimização no sequenciamento de tarefas nas máquinas. Além da heurística, também é utilizado um modelo de otimização para conjunto de menor número de máquinas e tarefas, onde se poderá ter um comparativo solução da heurística x solução ótima. Para conjuntos com maior número de máquinas e tarefas, será aplicada apenas a heurística. São geradas aleatoriamente dez instâncias com características diferentes em relação aos tempos de processamento, algumas são resolvidas através do método exato e heurístico, outras somente pelo heurístico. Ao final do trabalho, os resultados são apresentados e mostram-se usuais e efetivos em empresas com o tipo de produção apresentado.

Palavras-chave: heurística; otimização; máquinas paralelas; *scheduling*.

ROCHA, A. T. Application of a heuristic in the production sequence of a steel parts industry, 2017. Monografia (Bacharel em Engenharia de Produção) – Universidade Tecnológica Federal do Paraná.

ABSTRACT

This work focuses on the application of an improvement heuristic, from an initial solution. The goal is to find a very good makespan in a timely manner for an environment of machines called unrelated parallel machines, will be given the example of application of this type of machine environment, which is in a company of the manufacturing industry of industrial parts under order. This industry basically works with cutting, bending, welding and painting of steel sheets, thus making, steel parts. The orders of this type of company are very dynamic, and the occupation of certain machines is great, requiring optimization in the sequencing of tasks in the machines. In addition to the heuristic, an optimization model is also used for a set of smaller number of machines and tasks, where it is possible to have a comparative solution of the heuristic x optimal solution. For sets with more machines and tasks, only heuristics will be applied. Ten instances with different characteristics are generated randomly in relation to the processing times, some are solved by the exact and heuristic method, others only by the heuristic. At the end of the work, the results are presented and are shown to be usual and effective in companies with the type of production presented.

Palavras-chave: heuristic; otimization; parallel machines; scheduling.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

No planejamento e controle de produção, programar e sequenciar tarefas pode parecer algo simples, mas nem sempre é, principalmente tratando-se de um grande número de máquinas e tarefas. Quando buscamos um sequenciamento ótimo, as coisas se tornam ainda mais complexas, porque são necessários recursos computacionais para encontrar a solução em tempo hábil.

O planejamento e controle de produção se torna uma atividade cada vez mais necessária nas empresas globalizadas, isso porque toda atividade necessita ser planejada e controlada para que a organização atinja seus objetivos e atenda o cliente no prazo determinado. Portanto, o objetivo do planejamento e controle é obter a programação e realizar a manutenção da mesma (LOPES et MICHEL, 2007).

Para Pinedo (2010), o sequenciamento de produção é a habilidade de alocar e ordenar tarefas para serem executadas em máquinas. Já Maccarthy e Liu (1993) denominam o problema de máquinas paralelas como aquele em que estão disponíveis duas ou mais máquinas que podem executar a tarefa, mas somente uma executa.

Uma heurística é um procedimento realizado através de um modelo cognitivo, utilizado regras baseadas na experiência e conhecimento sobre o problema. Ao contrário dos métodos exatos, que buscam encontrar a solução ótima através de uma busca, ainda que implícita em todas as soluções possíveis, as heurísticas apresentam um certo grau de conhecimento do problema, gerando um número muito menor de possíveis soluções. Diante da complexidade em encontrar soluções ótimas para problemas de otimização, métodos heurísticos são utilizados com o objetivo de encontrar uma boa solução, ou solução factível, mesmo que não seja a ótima, em um tempo computacional razoável (FERNANDES;FILHO, 2010).

Nos atuais sistemas de produção, nem toda a produção é feita em série, podendo ser fabricados diferentes tipos de produtos a todo momento, de acordo com o pedido dos clientes, um exemplo muito comum são as empresas de fabricação de peças de aço, que comumente fazem peças sob encomenda. Sendo assim, o planejamento e controle de produção se torna ainda mais complexo.

Diante do exposto, essa monografia tratará da aplicação de uma heurística em um sistema de máquinas paralelas não-relacionadas, encontrado em empresas siderúrgicas de corte e dobra de peças de aço, que tratam de um grande número de máquinas e tarefas, onde o tempo para se encontrar uma solução ótima, utilizando os recursos computacionais é muito alto.

1.1 OBJETIVO GERAL

Resolver um problema de sequenciamento de máquinas em um ambiente denominado de máquinas paralelas não-relacionadas.

1.2 OBJETIVO ESPECÍFICO

- Estudar o funcionamento do sistema em questão, seus parâmetros e variáveis que são importantes para a resolução do problema;
- Aplicar uma heurística na resolução do problema de sequenciamento de uma indústria siderúrgica com produção puxada;
- Analisar e tirar conclusões sobre a utilização de modelos exatos ou heurísticas nesse tipo de produção.

1.3 JUSTIFICATIVA

Com o recente desenvolvimento do mundo dos negócios, as empresas buscam cada vez mais serem competitivas, isso exige que os recursos sejam aplicados de forma eficiência e eficaz. A Pesquisa Operacional vem sendo utilizada para encontrar soluções ótimas de problemas encontrados em diversas áreas, uma delas é o Planejamento e Controle de Produção, mais precisamente, a Programação da Produção, que trata de alocar tarefas em máquinas que elas possam ser processadas.

Os atuais esforços em desenvolver modelos exatos têm sido satisfatórios, no entanto, estes modelos, dependendo do número do número de máquinas e tarefas, demandam muito tempo de processamento. Isso resulta em modelos exatos, porém, não utilizáveis na prática, devido ao seu tempo de processamento nos computadores, o que pode resultar muitas vezes em anos de espera até se encontrar a solução ótima.

Diante do exposto, este trabalho se faz importante pois irá propor uma solução para o problema de um sistema de máquinas paralelas não-relacionadas de uma indústria siderúrgica, um modelo exato pode ser utilizado até certo número de máquinas e tarefas, após esse número, o número de interações torna-se muito alto, então é necessário utilizar uma heurística para obter uma boa solução factível no menor tempo possível.

2 REVISÃO DE LITERATURA

2.1 PLANEJAMENTO E CONTROLE DA PRODUÇÃO

O Planejamento e controle de produção (PCP) tem como objetivo garantir eficiência e eficácia aos processos de produção, para desta forma atender as necessidades dos clientes. Assim, a otimização do sequenciamento da produção garante a melhor forma de cumprir os prazos e assim gerar a satisfação dos clientes (LOPES, 2008).

Gigante (2010) diz que o planejamento é o detalhamento do que se tem a intenção de realizar em determinado período de tempo. O controle é o processo de verificação e realização de ajustes necessários para que as operações sejam realizadas de acordo com o estabelecido. Slack (2002) diz que o planejamento e controle são técnicas de conciliar demanda e fornecimento. A tomada de decisão para controlar e planejar determinada operação dependerá da natureza da demanda e do fornecimento de tal operação.

Chiavenato (2005) diz que os objetivos do PCP são os seguintes:

• Coordenar e integrar todos os órgãos envolvidos diretamente e indiretamente no processo de produção da empresa;

• Garantir a entrega dos produtos aos clientes nas datas previstas ou prometidas;

• Garantir disponibilidades de matérias-primas e componentes que serão requisitados pelos processos de produção;

• Balancear o processo produtivo de modo a evitar gargalos e desperdícios;

• Aproveitar o máximo da capacidade instalada, bem como os recursos aplicados;

• Estabelecer uma maneira racional de obtenção de recursos, como matéria-prima (compras), de mão de obra (pessoal) e de máquinas e equipamentos (engenharia), etc.

• Estabelecer, utilizando ordens de produção, padrões de controle para que o desempenho possa ser avaliado e melhorado;

• Distribuir a carga de trabalho proporcionalmente aos órgãos produtivos, de modo a assegurar a melhor sequência da produção e o resultado mais eficaz e eficiente.

O gerenciamento do PCP deve levar em conta restrições tecnológicas do ambiente, o tempo de execução das atividades e a demanda, suprindo suas necessidades. Para conciliar volume e tempo, são necessárias três atividades integradas: carregamento, sequência e programação. O carregamento pode ser definido como a quantidade de trabalho a ser colocada em cada centro de trabalho. O sequenciamento é o ordenamento de execução das operações. Trata-se de uma atividade bastante complexa, exemplo: uma máquina possui o sequenciamento

de cinco atividades, sendo 5! sequências possíveis para a execução das atividades ou 120 soluções possíveis de ordenação. No caso de dez atividades, o número de atividades sobe para 3.268.800 soluções possíveis, um problema ainda considerado pequeno. Já a programação define os volumes de cada produto, datas de início e termino de produção e equipamentos utilizados (GIGANTE, 2010).

Donaire (1995) diz que o PCP é o departamento responsável por controlar e planejar a produção do mês ou de um período, é uma ferramenta essencial para definir os recursos e itens que serão utilizados na produção. Podemos dizer que o PCP está completo quando ele responde as seguintes questões:

• O que produzir?
• Quando produzir?
• Onde produzir?
• Como produzir
• Com o que produzir?
• Com quem produzir?

A partir do momento que temos todas essas respostas, o PCP criará um Plano Mestre de Produção (PMP) que é a diretriz de produção, a Figura 1 apresenta o papel do setor do PCP:

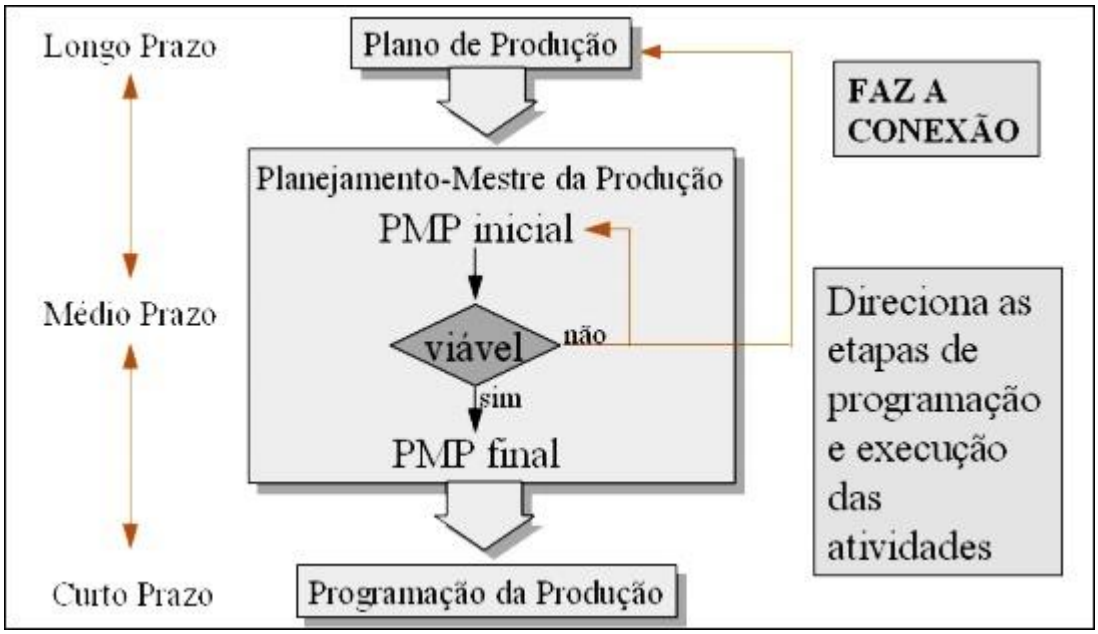

Figura 1 – O papel do setor de PCP

Fonte: Tubino (2000)

A Figura 1 mostra, principalmente, que o Plano de Produção acontece a longo prazo, já a programação da produção é em curto prazo, sendo feita diariamente na maioria das empresas. Também é sempre importante verificar se o Plano Mestre de Produção é viável, pois senão, é necessário voltar ao Plano de Produção para fazer um Plano Mestre de Produção viável.

Segundo Moreira (1999), o plano mestre de produção procura atrelar a produção com a demanda, com o menor custo possível, avaliando diversas alternativas previamente selecionadas, traçando um plano por um determinado mês que pode variar de 6 a 12. É determinado quanto e quando será produzido, programando recursos disponíveis, como mão-de-obra, equipamentos, matérias-primas, horas extras, etc.

Nambiar et al (1981) diz que a programação da produção está no nível mais baixo da hierarquia de um sistema de planejamento e controle de produção. Nessa hierarquia, o primeiro nível é o programa mestre, elaborado com base nas decisões de produção e capacidade. O segundo nível da hierarquia é onde se determina as quantidades a serem produzidas. O terceiro e último nível encontra-se após a definição das quantidades e datas de entregas, e compreende os programas de produção e alocação de recursos necessários. O objetivo de uma programação é atribuir e sequenciar a utilização desses recursos compartilhados, atendendo as restrições e minimizando custos.

2.1.1 Produção puxada e produção empurrada

Womack e Jones (1998) dizem que a produção puxada visa evitar o acúmulo de estoques através da premissa que um processo não deve produzir um bem ou serviço sem que o cliente ou outro processo o solicite. O conceito e produção puxada vem em contrapartida ao conceito de produção empurrada, que é muito utilizado na produção em massa. Na produção empurrada, grandes lotes são produzidos em ritmo máximo, pois trabalhadores e máquinas não devem ficar ociosos, assim, as necessidades de demanda não são consideradas, o que acaba gerando enormes excedentes, como ilustrado na Figura 2.

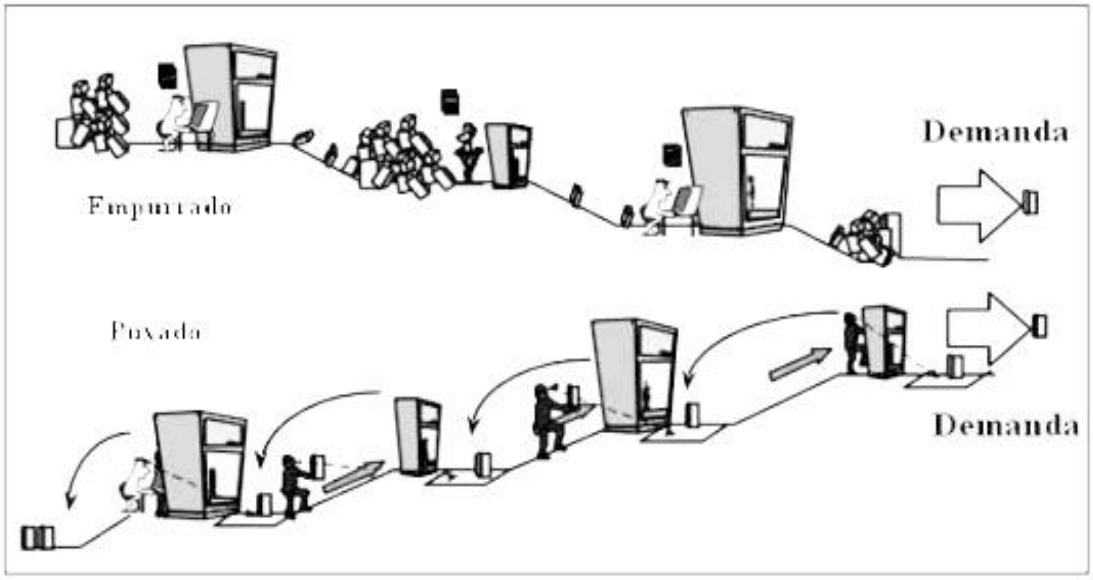

Figura 2 – Sistema empurrado e sistema puxado

Fonte: Corrêa e Corrêa (2004).

Além destas diferenças, os sistemas puxado e empurrado apresentam inúmeras outras diferenças, apresentadas no Quadro 1:

Quadro 1 – Diferenças dos sistemas puxado e empurrado.

	Produção Empurrada	Produção Puxada
Estoques	Médio	Baixo
Lead Time	Alto	Baixo
Variabilidade do *Lead Time*	Alto	Baixo
Setup	Alto	Baixo
Flexibilidade	Baixo	Alto
Variedade de produtos	Alto	Baixo
Planejamento	Alto	Baixo
Controle	Baixo	Alto
Complexidade do sistema	Alto	Baixo
Centralização	Alto	Baixo
Esforço computacional	Alto	Baixo
Programação	Ordens de produção	*Kanban*
Controle de estoque	Sistema computacional	Visual

Fonte: Vollman (2006)

A produção puxada é um dos conceitos do sistema Toyota de produção, também conhecido como produção enxuta, que é basicamente produzir somente o necessário. Tal

sistema de produção visa utilizar o mínimo de recursos possíveis, diminuindo assim os desperdícios.

Desperdício é todo e qualquer recurso que se gasta na execução de um produto ou processo além do necessário. É uma despesa extra que aumenta os custos normais do bem ou serviço sem trazer benefício para o cliente, ou seja, sem agregar valor ao produto (CAMPOS, 1996).

Os processos podem ser divididos em três tipos:

a) **Processos que geram valor**: são atividades que transformam a matéria-prima, modificando sua forma e gerando produtos ou subprodutos;

b) **Processos que não geram valor, mas são importantes para a produção e qualidade**: são atividades em que a matéria-prima não é beneficiada, mas têm muita importância, como o controle de qualidade, manutenção e segurança;

c) **Processos que não geram valor**: são atividades que não contribuem com as operações, tais como espera, estoque, movimentação desnecessária, etc.

Os desperdícios são resultados de processos que não agregam valor, de acordo com Shingo (1996), podem ser classificados em sete tipos principais, visualizados na Figura 3.

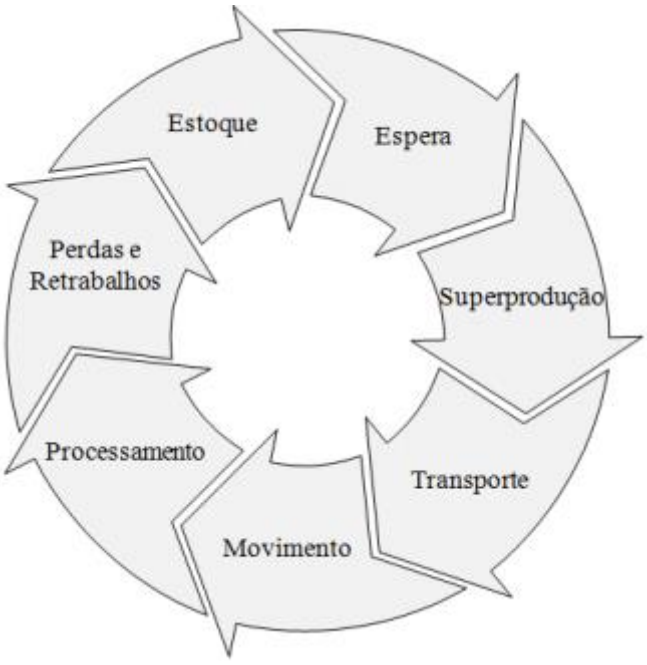

Figura 3 – Os sete tipos de perdas

Fonte: Shingo (1996)

Ohno (1997), similarmente a Shingo, também classifica os desperdícios:

• **Superprodução**: produzir mais do que o cliente é capaz de absorver, gerando excessos que se transforma em custos. É um desperdício bastante comum e que preocupa, pois pode causar outros desperdícios.

• **Estoque**: é resultado da superprodução e gera consequências graves, como o aumento do lead time, tempo que uma peça leva para percorrer todo o processo produtivo, está ligado diretamente com os giros de estoque. Quanto maior o *lead time*, maior a quantia em dinheiro parada na forma de estoques e, consequentemente, menos recursos para investimentos;

• **Transporte**: desperdício gerado pela movimentação de componentes, peças, matéria-prima ou produtos acabados dentro da fábrica ou entre outras fábricas;

• **Defeitos**: geram o retrabalho ou a perda da peça, isso aumenta os custos e não agrega valor ao cliente. Na mentalidade enxuta, deve-se fazer o certo da primeira vez.

• **Processos desnecessários**: relaciona-se aos processos que não agregam valor para o cliente e devem ser eliminados, como inspeções e verificações, pois utilizam recursos da empresa e não são revertidos em lucros.

• **Espera**: o colaborador deve utilizar todo o seu tempo realizando atividades que agregam valor, portanto o tempo gasto com espera é considerado um desperdício.

Shingo (1996) diz que a perda por tempo de espera consiste no tempo em que nenhum processo ou operação é executado pelo operador ou pelas máquinas, ele diz que existem três tipos de perda por espera:

a) **Espera de processo**: quando ocorre falta ou atraso de matéria-prima, atraso no processamento de lotes ou atraso devido a gargalos;

b) **Espera de lote**: ocorre quando algumas peças já passaram por um processo, mas têm que esperar o restante do lote para seguirem para o próximo processo ou etapa;

c) **Espera do operador**: quando o operário permanece ocioso.

2.2 SEQUENCIAMENTO DE PRODUÇÃO

Segundo Tubino (2000), processos sequenciais em lotes são padronizados e seguem um conjunto de operações e procedimentos que definem um produto. Neste sistema de produção o trabalho é mais flexível e menos especializado o que auxiliar na demanda de produção de diversas quantidades ou variedades de produtos. A programação desse tipo de produção, deve avaliar o que produzir e os recursos disponíveis para a produção. As regras de sequenciamento auxiliam a partir de informações sobre lotes ou estado do sistema de produção, sequenciamento

de produtos em fila e qual produto terá prioridade de processamento (*lead time*), conseguindo uma estimativa de data de entrega de produtos a clientes.

Desta forma, as regras de sequenciamento de produção podem ser classificadas de diversos status:

• **Regras Estáticas**: não alteram as prioridades quando ocorrem mudanças no ambiente produtivo;

• **Regras Dinâmicas**: acompanham as mudanças alternando as prioridades;

• **Regras Globais**: consistem as informações de outros recursos, como antecessor e sucessor na definição das prioridades.

Contudo, Tubino (2000) ainda comenta que as regras têm que ser separadas em prioridades simples, ponderadas e heurísticas sofisticadas, sendo as regras simples baseadas em uma característica do trabalho a ser executado, como data de entrega, tempo de folga restante, tempo de processamento, etc. As regras de sequenciamento não são pré-estabelecidas e eficazes para todas as situações, depende de cada caso, como apresentado a seguir:

• **Primeiro que Entra, Primeiro que Sai (PEPS)** – os lotes serão processados de acordo com o momento em que chegam ao recurso. Esta regra é bem simples, porém pouco eficiente, já que nela, os lotes com tempos longos, retardam toda a sequência da produção, gerando tempo ocioso nos próximos processos, fazendo com que o tempo de espera médio dos lotes seja alto.

• **Menor Tempo de Processamento (MTP)** – os lotes são processados de acordo com os menores tempos de processamento no recurso. Esta regra obtém um *lead time* médio baixo, reduzindo estoques em processamento, agilizando o carregamento das próximas máquinas e melhorando o nível de atendimento ao cliente. Um ponto negativo desta regra é que as ordens com tempos longos de processamento sejam desprezadas, principalmente se for grande o número de chegada de ordens com tempos menores, uma solução seria associar uma regra complementar em que uma ordem fosse desprezada um determinado número de vezes, ou após um certo tempo, avançasse ao topo da lista.

• **Índice de Prioridade (IPI)** – os lotes são processados de acordo com o valor da prioridade atribuída a um cliente ou produto.

• **Menor Data de Entrega (MDE) –** a produção será realizada de acordo com a menor data de entrega, isso faz com que aumente o número de estoque de produtos em processamento.

• **Índices Críticos (ICR) –** serão processados os lotes de menor valor ((data de entrega – data atual)/tempo de processamento), pode ocorrer atraso na entrega.

• **Índice de Folga (IFO)** – o cliente será priorizado. A tarefa a ser processada, é aquela com o menor índice de folga. O índice de folga é obtido pela diferença entre a data de entrega da tarefa

e o somatório do tempo de processamento das operações faltantes, dividido pelo número de operações faltantes.

• **Índice de Falta (IFA)** – busca-se produzir de acordo com o menor valor disponível do item em estoque, ou seja, quantidade em estoque menos a taxa de demanda.

• **Algoritmo de Johnson**, fornece um tempo mínimo de processamento para sequenciamento de n tarefas. As tarefas com tempos de processamento mais curtos são colocadas logo, se o tempo de processamento estiver na primeira máquina, e depois, se o tempo de processamento estiver na segunda máquina. Isso maximiza o tempo de operação simultâneo de ambos os centros de trabalho.

Segundo Martins e Laugeni (1999), esses critérios são aplicados dependendo dos níveis de trabalho existentes nas fábricas, onde determinam três diferentes regiões:

• **Região carga baixa**: a capacidade de produção fabril está em nível normal e com capacidade para produzir e atender mais demanda, sendo assim, qualquer critério atende as datas de entrega.

• **Região de carga excessiva**: a fábrica já está produzindo no seu limite máximo, qualquer critério já não atende à demanda.

• **Região de programação**: a demanda é variável e os critérios são mudados de acordo com a necessidade, dificultando a programação e a entrega.

Pinedo (2010) diz que o sequenciamento de tarefas é um processo de decisão que é utilizado em processo de produção e também em empresas de serviços. Os problemas de sequenciamento de tarefas podem ser descritos por $\alpha \mid \beta \mid \gamma$. O campo α descreve o ambiente de máquinas e possui uma única entrada, o campo β fornece as características e restrições de processamento e pode conter uma, nenhuma ou múltiplas entradas, já o campo γ descreve o objetivo a ser otimizado, e pode conter uma única entrada (mono-objetivo) ou mais de uma (multiobjetivo). De acordo com o autor, são os seguintes possíveis ambientes de máquinas:

• **Máquina única (1):** este caso é o mais simples de todos os ambientes de máquinas possíveis e é um caso especial dentro os outros tipos de máquinas. O ambiente de máquina única está ilustrado na Figura 4.

Figura 4 - Ambiente máquina única

Fonte: Espirito Santo (2014)

• **Máquinas paralelas idênticas (P_m):** são m máquinas idênticas paralelas. Cada tarefa requer uma operação e essa operação pode ser executada em qualquer uma das máquinas ou em qualquer uma que pertença a um subconjunto específico de máquinas.

• **Máquinas paralelas uniformes (Q_m):** há m máquinas em paralelo, mas com velocidades diferentes. A velocidade de cada máquina l é denotada por v_l. O tempo que a tarefa *j* gasta na máquina l é nomeado como p_{lj}, e é igual p_j/v_1. Se todas as máquinas têm a mesma velocidade, então o ambiente é de máquinas paralelas idênticas.

• **Máquinas paralelas não relacionadas (R_m):** são m máquinas diferentes paralelas. A máquina l pode processar a tarefa j na velocidade v_{ij}. O p_{ij}, tempo de processamento da tarefa *j* na máquina l é igual a $p_j \backslash v_{ij}$. Se as velocidades das máquinas são independentes das tarefas, então o ambiente é de máquinas paralelas uniformes.

Os ambientes de máquinas paralelas estão ilustrados na Figura 5.

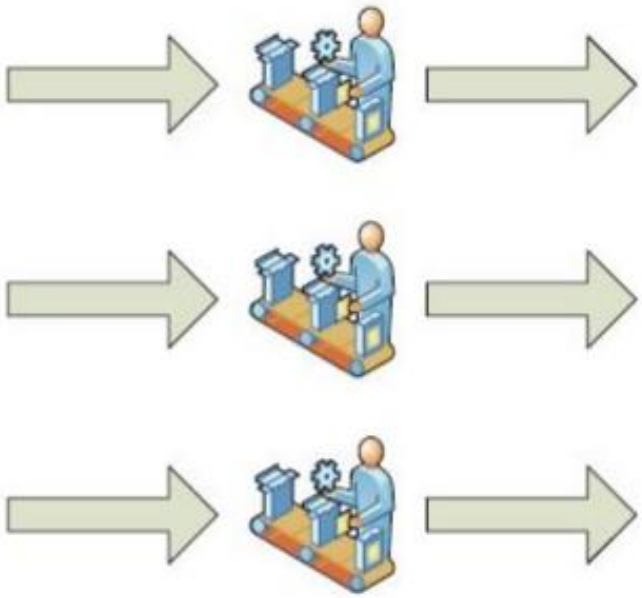

Figura 5 – Ambiente de máquinas paralelas
Fonte: Espirito Santo (2014)

• ***Flowshop* (F_m):** existem m máquinas em série, cada tarefa deve ser executada em cada uma das máquinas. Todas possuem o mesmo fluxo, são processadas primeiro na máquina 1, depois na máquina 2, assim por diante. O ambiente *flowshop* está ilustrado na Figura 6.

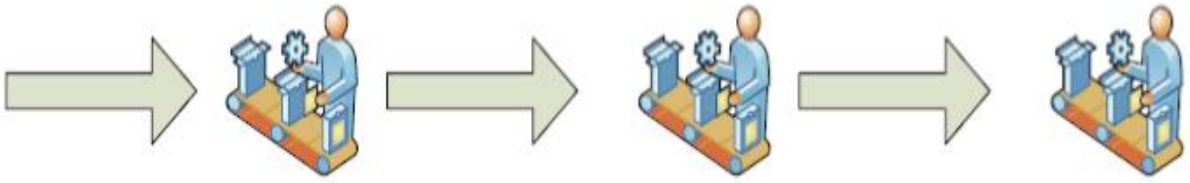

Figura 6 – Ambiente de *flowshop*
Fonte: Espirito Santo (2014)

• ***Flowshop* hibrido (FF_c):** o *flowshop* hibrido é um gênero do *flowshop* e dos ambientes de máquinas paralelas. Ao invés de máquinas em série, existem c estágios em série, cada estágio possui um número de máquinas idênticas em paralelo, cada tarefa é executada primeiro no estágio 1, em seguida, no estágio 2, e assim por diante. O ambiente *flowshop* híbrido está ilustrado na Figura 7.

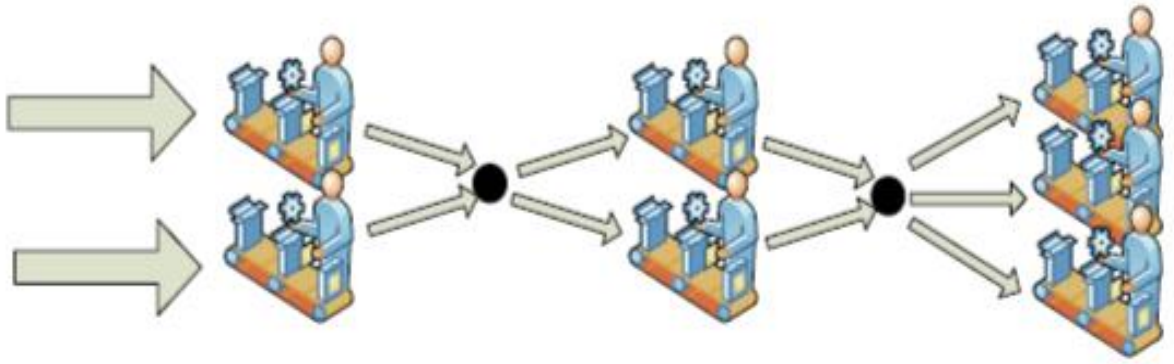

Figura 7 – Ambiente de *flowshop* híbrido
Fonte: Espirito Santo (2014)

• ***Jobshop* (J_m):** no *jobshop*, cada tarefa deve ser executada em cada uma das m máquinas e o seu próprio fluxo de máquinas pré-determinado.
• ***Jobshop* híbrido (FJ_c):** o *jobshop* híbrido é um gênero de *jobshop* e o ambiente de máquinas paralelas. Ao invés de máquinas em série, existem c estágios em série, cada estágio possui um determinado número de máquinas idênticas paralelas.

Os ambientes *jobshop* estão ilustrados na Figura 8.

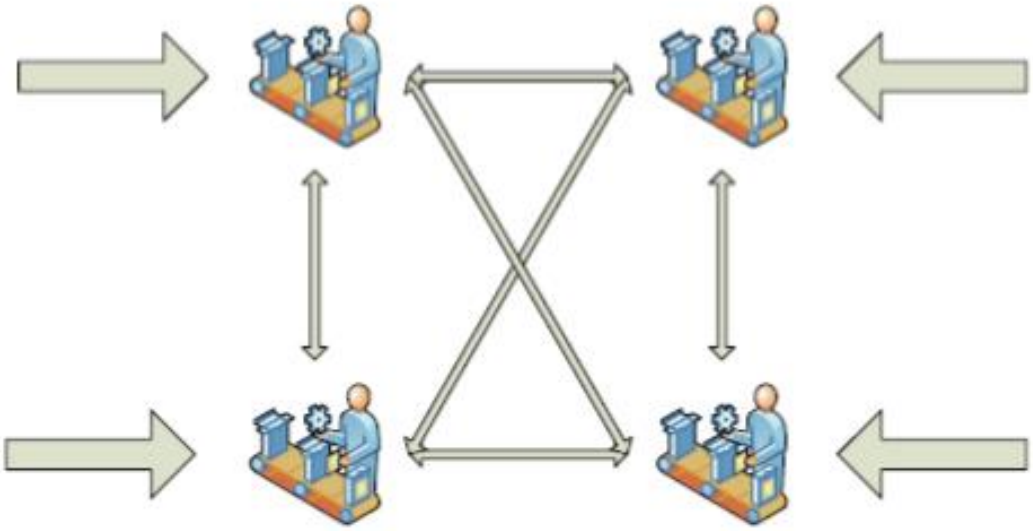

Figura 8 – Ambiente de *jobshop*
Fonte: Espirito Santo (2014)

• ***Openshop* (O_m):** no *openshop* cada tarefa deve ser executada em cada uma das m máquinas. No entanto, não existem restrições sobre o fluxo de máquinas em que as tarefas podem ser executadas.

Alguns exemplos de características que o campo β pode conter são:

• **Tempos de liberação (r_j):** a tarefa j não pode iniciar o seu processamento antes do seu tempo de liberação r_j.

• **Restrições de precedência (prec**): as restrições de precedência exigem que uma ou mais tarefas sejam concluídas antes do início do processamento de uma outra tarefa. Existem várias formas especiais de restrições de precedência: se cada tarefa tem, no máximo, um antecessor e, no máximo, um sucessor, as restrições são chamadas de cadeias. Se cada tarefa tem no máximo um sucessor, as restrições são chamadas de *intree*, se cada operação tem no máximo uma antecessora, são chamadas de *outtree*.

• **Tempos de *Setup* dependentes da Sequência (s_{ljk}):** s_{ljk} representa o tempo de setup da máquina l, quando a tarefa j precede a tarefa k. s_{ljk} pode ser diferente de s_{lkj}, pois o tempo de setup depende da sequência que está sendo executada, quando os tempos de setup são independentes da sequência, eles são incluídos nos tempos de processamento das tarefas.

• **Elegibilidade de Máquinas (M_j)**: ocorre se nem todas as m máquinas são capazes de processar a tarefa j. M_j é o conjunto de máquinas que podem executar a tarefa j.

• **Permutação (prmu):** esta restrição só aparece em ambientes *flowshop*. Ela indica que a ordem (permutação) em que as tarefas passam pela primeira máquina é mantida em todas as outras máquinas.

• **Bloqueio (*block*)**: o bloqueio também é uma situação que só pode ocorrer em *flowshop*. Essa restrição implica que a máquina que está executando uma operação de uma dada tarefa não é liberada enquanto a mesma não iniciar a sua próxima operação, ou seja, a máquina fica bloqueada.

• ***no-wait* (nwt):** essa característica também só ocorre em ambientes *flowshops*, e significa que as tarefas não podem esperar entre duas operações sucessivas, ou seja, assim que terminar uma operação, a próxima deverá iniciar.

Existem diversos objetivos que podem ser objeto de interesse em problemas de otimização (campo γ), como, por exemplo:

• **Minimização do *Makespan* (C_{max}):** o *makespan* é o instante de tempo em que a última tarefa é completada.

• **Minimização do maior *Lateness* (L_{max}):** o *lateness* de uma tarefa é definido por $L_j = C_j - d_j$, sendo C_j o instante em que a tarefa j é concluída e d_j o instante em que a tarefa deve ser entregue. Se L_j é positivo, a tarefa foi concluída depois do tempo de entrega, e se é negativo, a tarefa foi concluída antes da data de entrega.

• **Minimização do somatório ponderado do *Tardiness* ($\sum w_j T_j$)**: o *tardiness* de uma tarefa é definido por $\max\{L_j,0\}$, ou seja, se houver atraso, o *tardiness* é igual ao *lateness*, e, se não houver atraso, o *tardiness* é igual a zero. Nesse caso, a cada tarefa é atribuído um peso w_j, de acordo com a importância da tarefa.

2.2.1 Máquinas paralelas não-relacionadas

A programação da produção exige decisões sobre sequenciamento e alocação de recursos, quando há apenas um recurso, máquina única, a alocação desse recurso é determinada pela sequência de decisões, ou seja, não há distinção entre sequenciamento e alocação de recursos. Para fazer uso dessa distinção, devemos examinar modelos com mais de uma máquina, que podem ser: máquinas paralelas, *flowshop* e *jobshop*. Em sistema paralelos, os trabalhos consistem em uma operação, como no modelo de máquina única, porém como há mais de uma máquina, a estrutura é um pouco mais complexa (ARENALES et al, 2007).

Em ambientes de máquinas paralelas, há três tipos, que se diferenciam de acordo com a velocidade:

• **Idênticas**: todas as máquinas têm velocidades de processamento iguais para uma mesma tarefa;
• **Uniformes**: as máquinas têm velocidades diferentes, mas a velocidade é constante e não depende da tarefa;
• **Não-relacionadas**: a velocidade de cada máquina depende das tarefas executas, podendo ser diferente para cada máquina.

Por possuir velocidades diferentes para cada tarefa, o ambiente de máquinas paralelas não-relacionadas é um pouco mais complexo do que os demais, no entanto, em modelos exatos, é possível utilizar o mesmo modelo para os três tipos.

Arenales et al (2007) apresentam um modelo matemático para representar o ambiente de máquinas paralelas não-relacionadas como o objetivo de minimizar o *makespan*:

Parâmetros

P_{ik}: tempo de processamento da tarefa i na máquina k.
G: Número grande.

Variáveis

C_{ik}: Instante de término do processamento da tarefa i na máquina k.
x_{ijk}: Assume 1 se a tarefa i precede imediatamente a tarefa j na máquina k e 0, caso contrário.
C_{max}: Maior instante de término de processamento entre todas as máquinas ou *makespan*.

No modelo será utilizado uma tarefa fictícia 0, que será a predecessora da primeira tarefa de cada máquina. Isso garante que cada tarefa tenha uma predecessora e também que cada tarefa seja processada em uma única máquina. Sendo assim, em cada máquina, a tarefa que for sucessora da tarefa fictícia 0 será a primeira tarefa a ser processada na máquina.

1) $Minimizar\ Cmax$

Sujeito a

2) $\sum_{k=1}^{m}\sum_{i=1}^{n} x_{ijk} = 1$ j = 1,..., n

3) $\sum_{j=1}^{n} x_{0jk} \leq 1$ k = 1,...,m

4) $\sum_{i=0;i\neq h}^{n} x_{ihk} - \sum_{j=0;j\neq h}^{n} x_{hjk} = 0$ h = 1,...n; k = 1,...,m

5) $C_{jk} \geq C_{ik} - G + (p_{jk} + G)x_{ijk}$ i = 0,...n; j = 1,...,n; k = 1,...,m

6) $C_{max} \geq C_{ik}$ i = 1,...,n; k = 1,...,m

7) $C_{max} \geq 0$

8) $C_{ik} \geq 0$ i = 1,...,n; k = 1,...,m

9) $x_{ijk} \in \{0,1\}$ i = 0,...n; j = 0,...,n; k = 1,...,m

Onde:

1) Representa a função-objetivo do problema, Cmax é o *makespan* geral do sistema.
2) Restrição: Cada tarefa tem uma única predecessora em uma única máquina. Para cada valor de j, i e k variam até se encontrar algum (somente um) x_{i1k} que seja igual a 1. O i varia de 0 a n, pois uma tarefa fictícia 0 (zero) precede a primeira tarefa de cada máquina. Assim, é garantido que cada tarefa tenha uma única predecessora e também que cada tarefa seja processada em uma máquina.
3) Restrição: Cada máquina pode ter, no máximo, uma sequência de tarefas. A sequência de cada máquina é iniciada pela tarefa fictícia 0 (zero), essa restrição garante que em cada máquina k, no máximo, um único x_{0jk} seja 1.
4) Restrição: cada tarefa deve ter uma única sucessora, exceto pela tarefa 0 (zero). Para um dado k e um dado h: se houver uma tarefa i predecessora de h (algum $x_{ihk} = 1$), então haverá alguma tarefa j sucessora de h (algum $x_{hjk} = 1$), nem que i ou j seja a tarefa fictícia 0 (zero).
5) Restrição: garante instantes de término adequados para cada tarefa. A restrição é ativada com $x_{ijk} = 1$, fazendo com que o término da tarefa j seja o término da predecessora mais o seu tempo de processamento.

6) Restrição: faz com o que o *makespan* seja igual ao maior tempo de processamento entre os tempos de processamento de todas as máquinas.
7) Tipo de variável: C_{max} deve assumir um valor maior ou igual a zero.
8) Tipo de variável: as variáveis C_{ik} devem assumir valores maiores ou iguais a zero.
9) Tipo de variável: variáveis binárias.

Este é um modelo de minimização do *makespan* do sistema, porém, o mesmo pode ser adaptado para outras funções-objetivo, como minimização do atraso total, atraso máximo, tempo de fluxo total (*flowtime*), etc, bastando fazer a adição dos parâmetros utilizados para tais funções, como, por exemplo, as datas de entregas. Para encontrar a solução ótima, basta adaptar esse modelo em linguagem de programação de algum *software* de otimização.

Bake e Triesch (2009) dizem que minimizar o *makespan* é bastante difícil para ambientes mais complexos, mas para máquinas paralelas tendem a ser relativamente simples, porém, para $m \geq 3$ os tempos de processamento tendem a ser muito altos em modelos ótimos. Assim, esses modelos somente são viáveis para problemas relativamente pequenos. Embora soluções ótimas sejam difíceis de serem obtidas para $m \geq 3$, alguns procedimentos heurísticos funcionam muito bem. Uma heurística simples apresentada pelos autores é chamada de "agendamento de lista" e consiste em alocar a primeira tarefa da fila a uma máquina livre, o que é basicamente, alocar a primeira tarefa na primeira máquina, a partir da segunda tarefa, a alocação é feita para a máquina que possui o menor *makespan* até o atual momento. Isso significa que as tarefas são alocadas uma a uma, olhando apenas para as máquinas, não para os tempos de processamento das tarefas e as diferenças que eles têm de uma máquina para a outra.

3 METODOLOGIA

A metodologia deste trabalho consiste na formulação de um problema, construção de hipóteses e a identificação das relações entre variáveis, que são constituídas por uma referência teórica ou um conceito de pesquisa (GIL, 1994).

3.1 EMPRESA DO ESTUDO

Trata-se de um estudo baseado na produção de uma empresa de fabricação de peças industriais (corte e dobra de aço), localizada na cidade de Ponta Grossa, estado do Paraná, tal empresa trabalha com encomendas (produção puxada) destas peças, sendo assim, há uma variação muito grande de peças produzidas e tempos de processamento, as tarefas possuem diferentes tempos de corte e dobra, porque dependem sempre do tamanho e material da peça em questão. Essa empresa de Ponta Grossa será usada como exemplo, no entanto, há diversas empresas que possuem o mesmo sistema de produção e que podem ter esse trabalho aplicado.

As empresas de produção de peças industriais podem ter produção do tipo empurrada, ou seja, podem manter um padrão na produção, pois são elas que decidem quais peças produzir, há também as produtoras de peças com produção puxada, mas que normalmente atendem sempre os mesmos pedidos, ou seja, já possuem a informação sobre quais peças irão produzir pelos próximos meses, e assim manter um certo padrão na programação da produção, há também as empresas aqui exemplificadas, que possuem uma variedade muito grande de pedidos, muitas vezes, recebendo um pedido e tendo que atende-lo no mesmo dia, sendo assim, o sequenciamento de produção é diário, e tem de ser feito o mais rápido possível.

3.2 MODELOS E HEURÍSTICA UTILIZADOS

Para resolver o problema de sequenciamento na empresa do ramo acima citado, pode-se utilizar o modelo matemático apresentado na Subseção 2.2.1, bastando adaptá-lo a uma linguagem do *software* Lingo.

O Lingo é uma ferramenta utilizada para a resolução de problemas lineares e não-lineares, encontrando a solução ótima, isso significa que o Lingo faz todas as combinações possíveis para determinados problemas, colocando como solução ótima a melhor solução factível encontrada. O modelo Lingo está apresentado no APÊNDICE 1.

Como o modelo exato do Lingo é efetivo apenas para um determinado número de máquinas e tarefas, para as instâncias com mais de 2 máquinas e 8 tarefas, será viável utilizar uma heurística de melhoria, a partir de uma solução inicial. A heurística será utilizada para todas as instâncias, para as menores, para servir de comparativo com a solução ótima e para instâncias maiores, para que a solução factível encontrada seja dada como solução para o problema de sequenciamento.

A heurística apresentada na Subseção 2.2.1 tem como ponto positivo não permitir que muitas tarefas sejam alocadas na mesma máquina, pois sempre que uma máquina estiver vazia (ou com menor instante de término de processamento), a tarefa será alocada para essa, no entanto, tal heurística não possui um comparativo entre os tempos de processamento diferentes para cada tarefa em cada uma das máquinas, nem alterações em posições para possíveis testes de melhoria.

Sendo assim, será utilizada uma heurística de melhoria, partindo de uma solução inicial. A solução inicial consistirá em colocar cada tarefa na máquina que ela possui o menor tempo de processamento, explicada a seguir (com um exemplo curto, todos os tempos de processamento estão expressos em segundos):

Solução Inicial

1º passo: seleciona-se a primeira tarefa da lista (Figura 9a).

2º passo: compara-se os tempos de processamento dessa tarefa em cada uma das máquinas (Figura 9b).

3º passo: aloca a tarefa para a máquina que possui o menor tempo de processamento para ela (Figura 9c).

4º passo: seleciona a (s) próxima (s) tarefa (s) da lista, repetindo os passos 2 e 3 (Figura 9d).

5º passo: a solução é definida, calculando-se os instantes de término de processamento de cada máquina e também o *makespan* do sistema. Essa solução é definida como solução atual (Figura 10).

Onde:

m_k: máquina;

n_j: tarefa;

t_{mn}: tempo de processamento da tarefa n na máquina m.

a)

	m1	m2	m3
n1	10	20	15
n2	17	18	21
n3	19	34	21
n4	23	24	37
n5	27	37	28
n6	31	40	11

b)

	m1	m2	m3
n1	10	20	15
n2	17	18	21
n3	19	34	21
n4	23	24	37
n5	27	37	28
n6	31	40	11

c)

	m1	m2	m3
n1	10	20	15
n2	17	18	21
n3	19	34	21
n4	23	24	37
n5	27	37	28
n6	31	40	11

d)

	m1	m2	m3
n1	10	20	15
n2	17	18	21
n3	19	34	21
n4	23	24	37
n5	27	37	28
n6	31	40	11

Figura 9 – Passo a passo da heurística de solução inicial.

Fonte: A autora

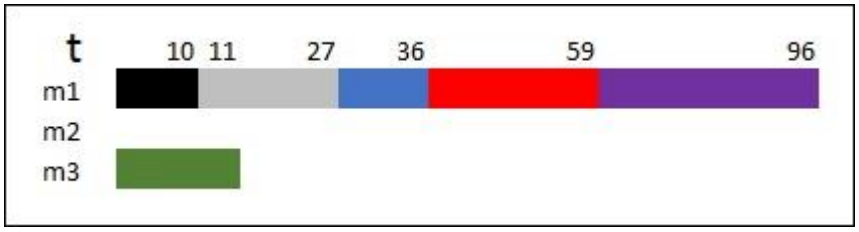

Figura 10 – Gráfico de Gantt da solução inicial para o exemplo

Fonte: A autora

Através do Gráfico de Gantt, podemos observar que cinco tarefas foram alocadas para a máquina 1 (tarefas 1,2,3,4 e 5) e uma tarefa (6) foi alocada para a máquina 3, nenhuma tarefa foi alocada para a máquina 2. O instante de término de processamento da máquina 1 é 96 segundos, da máquina 2 é zero e da máquina 3 é 11 segundos, sendo assim, o *makespan* do sistema é 96 segundos (maior instante de término processamento entre os instantes de término de processamento de todas a máquinas).

Após a definição desta solução inicial, será utilizada a **heurística de melhoria** que consiste nos seguintes passos:

1º passo: seleciona-se a primeira tarefa da lista. (Figura 11)

2º passo: aloca a tarefa selecionada para a primeira máquina da lista (se ela foi alocada inicialmente para essa máquina, então aloca-a para a máquina seguinte). As demais tarefas permanecem alocadas conforme a solução atual. (Figura 11)

	m1	m2	m3
n1	10	20	15
n2	17	18	21
n3	19	34	21
n4	23	24	37
n5	27	37	28
n6	31	40	11

Figura 11 – Primeiro e segundo passos da heurística de melhoria

Fonte: A autora

3º passo: com todas as tarefas alocadas para alguma máquina (e a primeira tarefa alocada para uma máquina diferente da solução inicial) calcula-se os instantes de término de processamento de cada máquina e também o *makespan* do sistema. O *makespan* calculado com a tarefa máquina modificada é definido de solução teste.

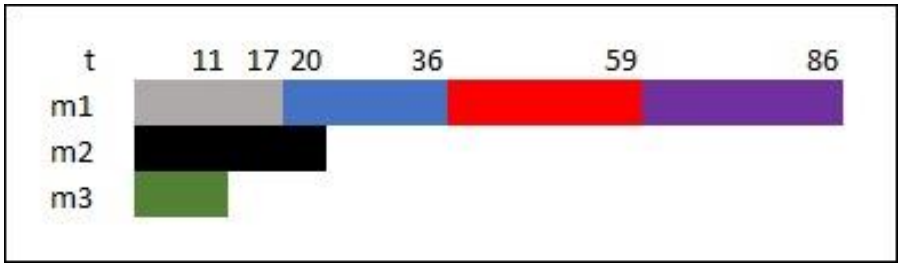

Figura 12 - Gráfico de Gantt após a primeira alocação

Fonte: A autora

4º passo: compara-se o *makespan* da solução atual (se for a primeira iteração, a solução atual será a solução inicial) com o *makespan* da solução teste.

5º passo: se o *makespan* da solução atual for melhor -> mantém a solução atual, se o *makespan* da solução teste for melhor -> a solução teste é definida como solução atual (houve melhoria).

No exemplo houve melhoria, a segunda solução é definida como solução atual.

6º passo: seleciona-se a mesma tarefa e a aloca na próxima máquina, caso a tarefa já tenha sido alocada e testada em todas as máquinas, então é selecionada a próxima tarefa.

Neste caso, a tarefa 1 foi retirada da máquina 2 (onde ela está atualmente pela solução atual) e foi colocada para ser processada na máquina 3, repetindo os passos 3,4 e 5, assim sucessivamente, para todas as tarefas em todas as máquinas.

7º passo: se durante os passos 1,2,3,4,5 e 6, a solução teste foi definida como solução atual, então, repete-se novamente todos estes passos, caso contrário, fim da heurística, a solução encontrada é a definida como solução atual e final.

Ao final de todas as iterações, foi obtida a seguinte solução atual e final:

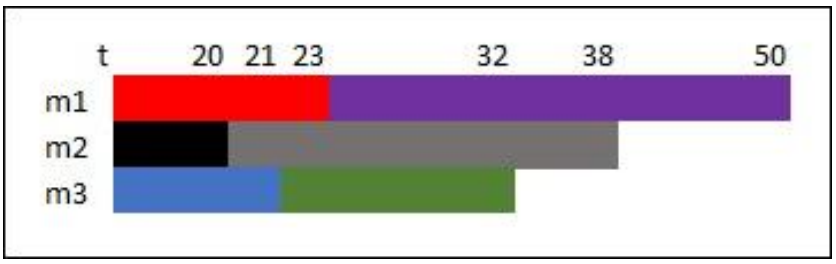

Figura 13 - Gráfico de Gantt do resultado da heurística de melhoria para o exemplo.
Fonte: a autora

Pode-se observar uma mudança na distribuição das tarefas, onde é visto um melhor aproveitamento das máquinas, com 2 tarefas alocadas em cada uma, o *makespan* do sistema é 50 segundos, como já dito, não é garantido que este seja o *makespan* ótimo, já que estamos trabalhando com heurísticas.

Para a aplicação desta heurística, será utilizado um modelo em linguagem de programação C, apresentado no APÊNDICE 2.

A heurística de melhoria corrige diversos problemas que a solução inicial pode apresentar, apesar de colocar as tarefas nas máquinas mais rápidas, a solução inicial pode ser problemática caso uma máquina seja muito melhor do que as outras, pois assim, todas as tarefas serão alocadas para essa máquina, deixando as demais ociosas (como foi observado no exemplo).

Já a heurística apresentada pelo autor na subseção 2.2.1, não considera as diferenças de tempos entre as máquinas para processar uma mesma tarefa, o que pode ocorrer que uma tarefa seja colocada em uma máquina que possui desempenho muito abaixo da outra, não havendo o melhor aproveitamento possível da eficiência e disponibilidade das máquinas.

3.3 APLICAÇÃO

A empresa exemplificada, para a produção de suas peças, utiliza de chapas inteiras que são transformadas nas peças, que possuem projetos feitos pelos clientes, ou pela própria empresa, quando isso também for requisitado pelo cliente.

São utilizadas na fabricação diversas máquinas industriais, incluindo calandras, dobradeiras e peças de corte. Para o estudo nesse trabalho, serão utilizadas as máquinas de corte, pois são diversas máquinas, com características diferentes para diferentes peças (tarefas), o que se encaixará no problema de máquinas paralelas não relacionadas.

Entre as máquinas que realizam o corte,

- Guilhotina: realiza corte por cisalhamento, podendo ser mecânica, hidráulica ou pneumática. A mecânica possui capacidade para cortar chapas de até 13 mm de espessura, a hidráulica corta chapas de até 25 mm e a pneumática é utilizada para chapas mais finas, de até 1,2 mm de espessura. É um dos cortes mais baratos. Esse é um dos cortes mais baratos e com velocidade razoável, no entanto, a precisão da guilhotina não é boa, sendo indicada para peças que não demandem muito essa característica.

- Corte plasma: surgiu através do processo TIG (gás inerte de tungstênio) em que se utiliza um gás ionizado, que sai através de um bocal com diâmetro reduzido, o gás do corte flui pelo centro da peça, que possui eletrodo negativo, assim um arco elétrico ioniza o gás, formando o plasma. A alta temperatura do plasma funde o metal, produzindo o corte. O gás pode ser argônio, hidrogênio ou nitrogênio. A capacidade de corte da máquina de corte a laser depende de muitos fatores como: tipo de gás utilizado no corte, quantidade de vazão, diâmetro do bico de corte e tensão do arco elétrico. O corte de plasma não possui alto custo e é recomendado para qualquer espessura de peça, sendo até mais rápido que o corte a laser.

- Corte a laser: é feito através de estimulação radioativa da luz, que é amplificada, produzindo um potente feixe de luz, esse feixe (que é energia luminosa concentrada em um raio) irá alterar a composição molecular de uma superfície estável, no caso, o aço a ser cortado, permitindo assim, o corte dessa superfície. A espessura que a máquina de laser pode cortar, depende do modelo e fabricante dessa máquina, mas ela fica em torno de no máximo 20 mm. É um corte barato e rápido.

- Jato d'água: Esse corte utiliza um jato de água com grande velocidade e pressão, para peças de aço, é necessário utilizar uma substância abrasiva, aumentando o poder do corte. É a máquina mais precisa de todas, normalmente, é usada para peças específicas que precisam de alta

precisão, este corte é mais caro que os demais, justamente por causa da substância abrasiva utilizada e por utilizar mais tempo para cortar, é um corte lento.

Para esse estudo de caso, não foram coletados dados, como tempo de processo de corte de uma peça em cada uma das máquinas, isso porque a variedade de peças é grande, bem como os tempos necessitam de maior conhecimento sobre os modelos de máquinas utilizados, frequência de manutenção, etc.

Sendo assim, foram geradas 10 instâncias aleatórias, as 5 primeiras com 2 máquinas e 8 tarefas em cada uma, já as 5 últimas com 10 tarefas e 4 máquinas em cada uma. As 5 primeiras instâncias foram resolvidas pelo software Lingo e também pela heurística, já as 5 últimas foram resolvidas apenas pela heurística.

3.4 GERAÇÃO DE INSTÂNCIAS

A instância 1 (tabela 1) teve todos os tempos de processamento gerados entre 20 e 60 segundos. A instância 2 (tabela 2) teve tempos de processamento gerados entre 30 e 50 segundos. A instância 3 (tabela 3) teve tempos de processamento gerados entre 35 e 45 segundos. A instância 4 (tabela 4) teve tempos gerados de 20 a 40 segundos para a máquina 1 e de 41 a 60 segundos para a máquina 2 (máquina 2 muito mais rápida). A instância 5 (tabela 5) teve tempos de processamento gerados entre 20 e 40 segundos para a máquina 2 e 41 e 60 segundos para a máquina 1 (máquina 1 mais rápida).

As instâncias a serem resolvidas pelo lingo e pela heurística se apresentam a seguir:

Onde:

m_k = máquina

n_i = tarefa

x_{mn} = tempo de processamento da tarefa n na máquina m (em segundos).

Tabela 1 – Instância 1 com tempos de processamento entre 20 e 60 segundos

	m1	m2
n1	41	49
n2	27	31
n3	39	26
n4	37	32
n5	41	34
n6	27	31
n7	58	27
n8	45	43

Fonte: A autora

Tabela 2 – Instância 2 com tempos de processamento entre 30 e 50 segundos

	m1	m2
n1	39	39
n2	42	41
n3	44	37
n4	49	32
n5	30	45
n6	37	50
n7	36	39
n8	45	47

Fonte: A autora

Tabela 3 – Instância 3 com tempos de processamento entre 35 e 45 segundos

	m1	m2
n1	45	45
n2	39	40
n3	45	41
n4	38	36
n5	45	36
n6	42	42
n7	44	43
n8	43	39

Fonte: A autora

Tabela 4 – Instância 4 com máquina 2 mais rápida

	m1	m2
n1	34	52
n2	33	48
n3	40	41
n4	24	48
n5	35	60
n6	40	58
n7	36	59
n8	20	58

Fonte: A autora

Tabela 5 – Instância 5 com a máquina 1 mais rápida

	m1	m2
n1	49	30
n2	57	30
n3	41	34
n4	59	35
n5	58	39
n6	57	40
n7	52	23
n8	47	34

Fonte: A autora

Como não serão abordados ambiente de somente duas máquinas, serão geradas aleatoriamente 5 instâncias com 4 máquinas e 10 tarefas, no entanto, com esse número, os métodos exatos são limitados no fator tempo, as próximas instâncias serão resolvidas apenas pela heurística.

Na instância 6 (tabela 6), foram gerados tempos de processamento aleatórios entre 10 e 50 segundos para ambas máquinas. Na instância 7 (tabela 7) foram gerados tempos de processamento entre 20 e 40 segundos para ambas as máquinas. Na instância 8 (tabela 8), será tentado entrar, o máximo possível, no ambiente real apresentado, sendo assim, a primeira e segunda máquinas terão tempos de processamento aleatórios entre 40 e 50 segundos, a terceira máquina terá tempos entre 16 e 39 segundos e a quarta máquina tempos entre 5 e 15 segundos (máquina 4 mais rápida de todas, máquina 3 mais rápida que a 1 e 2, máquina 1 e 2 com desempenho semelhante). Já na instância 9 (tabela 9) a máquina 1 será considerada como a mais

rápida, gerando números entre 5 e 15 segundos, as demais máquinas terão desempenhos semelhantes, com tempos de processamento entre 16 e 40 segundos. A instância 10 (tabela 10) considerará diferenças de tempos de tarefas, sendo assim, para as tarefas de 1 a 5, serão gerados números entre 15 e 25 segundos e para as tarefas de 6 a 10 terão tempos de processamento entre 26 e 40 segundos.

As instâncias, a serem resolvidas exclusivamente pela heurística, são apresentadas nas tabelas a seguir:

Tabela 6 – Instância 6 com tempos de processamento entre 10 e 50 segundos

	m1	m2	m3	m4
n1	26	30	10	47
n2	14	25	15	31
n3	16	11	33	15
n4	25	28	17	29
n5	10	10	29	40
n6	29	33	48	27
n7	25	11	30	27
n8	24	11	10	36
n9	18	44	35	15
n10	35	36	39	39

Fonte: A autora

Tabela 7 – Instância 7 com tempos de processamento entre 20 e 40 segundos

	m1	m2	m3	m4
n1	20	26	36	26
n2	31	29	31	25
n3	30	33	21	22
n4	36	21	27	35
n5	34	34	40	22
n6	25	29	25	30
n7	31	32	28	39
n8	36	24	35	39
n9	37	23	35	39
n10	28	37	34	24

Fonte: A autora

Tabela 8 – Instância 8 com máquina 4 mais rápida

	m1	m2	m3	m4
n1	43	45	31	10
n2	40	43	34	10
n3	46	49	28	13
n4	45	45	22	11
n5	41	49	24	6
n6	48	46	22	10
n7	46	50	30	10
n8	46	47	24	13
n9	42	47	30	9
n10	41	48	22	5

Fonte: A autora

Tabela 9 – Instância 9 com máquina 1 mais rápida

	m1	m2	m3	m4
n1	9	19	38	31
n2	13	21	29	21
n3	9	30	36	37
n4	10	17	19	17
n5	8	25	26	36
n6	14	24	32	25
n7	10	29	18	25
n8	9	22	36	35
n9	5	23	24	24
n10	15	36	39	32

Fonte: A autora

Tabela 10 – Instância 10 com variação de tempo entre tarefas

	m1	m2	m3	m4
n1	20	17	18	23
n2	25	19	18	25
n3	20	17	17	25
n4	15	25	19	21
n5	19	24	15	22
n6	31	39	31	26
n7	40	27	35	33
n8	29	28	32	39
n9	29	32	40	26
n10	26	40	37	26

Fonte: A autora

É importante destacar que, em ambientes reais, podemos trabalhar com um número muito maior de tarefas (e até de máquinas), porém estes exemplos são apenas simulações para verificarmos como aplicamos a heurística e como apresentamos os resultados.

O modelo do *software* Lingo, bem como o programa em linguagem de programação em C que foi compilado no *software* Dev C++ foram executado em um computador com 4 GB de memória RAM e que utiliza um processador Intel Core i5 5200U de 2 núcleos, com frequência de 2,2 Ghz e 3 MB de memória cache.

4 RESULTADOS E DISCUSSÕES

Resolvendo as 5 primeiras instâncias (tabelas 1, 2, 3, 4 e 5), através do *software* Lingo e das heurísticas de construção e melhoria, obtemos os seguintes resultados de *makespan* em segundos apresentados na tabela 11:

Tabela 11 – Resultados das instâncias 1 a 5

Instância	Número de máquinas	Número de tarefas	Solução do Lingo (s)	Solução Inicial (s)	Solução da Heurística de Melhoria (s)
1	2	8	132	162	136
2	2	8	149	187	149
3	2	8	163	195	171
4	2	8	149	262	155
5	2	8	156	265	171

Fonte: A autora

Observando estes resultados, podemos concluir que a heurística de melhoria teve melhoria significativa, em comparação aos resultados da solução inicial. Já quanto aos resultados ótimos, foram encontrados resultados próximos. Na segunda instância, o resultado encontrado pela heurística é igual ao resultado ótimo

Na instância 2, a solução inicial ficou razoavelmente próxima da solução ótima e da solução de melhoria, isso acontece porque, em alguns casos, dependendo dos tempos de processamento, a solução inicial já pode apresentar resultados muito bons.

Podemos verificar deficiências na solução inicial nas instâncias 4 e 5, onde uma máquina é mais rápida do que a outra, neste caso, a solução inicial tende a colocar todas as tarefas na máquina mais rápida, deixando a outra máquina sem tarefas para processar, a heurística encontra um melhor equilíbrio entre a distribuição das tarefas.

Partindo para casos mais complexos, onde acrescentamos máquinas e tarefas, temos as instâncias 6 a 10, já mencionadas anteriormente, tais instâncias estão mais próximas do que acontece em casos reais, e não podem ser resolvidas com modelos ótimos (devido ao elevado tempo de processamento), sendo assim, somente podemos utilizar de heurísticas para encontrar um sequenciamento factível e bom ou muito bom.

Resolvendo as 5 últimas instâncias (tabelas 6, 7, 8, 9 e 10) apenas pela solução inicial e heurística de melhoria, obtemos os resultados da tabela 12.

Tabela 12 – Resultados das instâncias 6 a 10

Instância	Número de máquinas	Número de tarefas	Solução Inicial	Solução da Heurística de melhoria
6	4	10	59	47
7	4	10	71	71
8	4	10	83	50
9	4	10	102	53
10	4	10	89	61

Fonte: A autora

Novamente verificamos melhorias significativas dos resultados de *makespan*, principalmente em instâncias em que uma máquina é mais rápida do que as outras, verificado na instância 9. Na instância 9 verificamos maior diferença entre os valores de *makespan* da solução inicial com o *makespan* da solução da heurística de melhoria, isso se mostra pelo fato de haver uma máquina mais rápida que as demais.

Na instância 7 verificamos que a heurística de melhoria não conseguiu melhorar a solução inicial, no entanto, isso não demonstra deficiências na heurística de melhoria, e sim, que a solução inicial também pode encontrar uma boa solução, dependendo do valor dos tempos de processamento de cada tarefa em cada máquina.

Tais resultados implicam que a aplicação é viável em ambientes de produção puxada, como nas empresas exemplificadas, tais empresas possuem diversas máquinas que podem processar a mesma peça, com velocidades diferentes, e possuem dificuldades no sequenciamento devido à complexidade do sistema. Problemas de sequenciamento complexos precisam, obrigatoriamente, de recursos computacionais para um sequenciamento ótimo ou muito bom, que vise economia de tempo e redução de custos.

É notável a vantagem no quesito tempo, visto que, nas primeiras 5 instâncias (Tabelas 1,2,3,4 e 5), o *software* Lingo levou cerca de 6 a 7 minutos para encontrar o resultado ótimo, enquanto a heurística encontrou uma boa solução em menos de 10 segundos. A heurística também resolveu as instâncias das tabelas 6,7,8,9 e 10 em poucos segundos.

5 CONCLUSÕES

Através deste trabalho, foi possível entender e conhecer melhor com funciona o sistema de produção de indústrias de peças de aço, tal sistema apesar de ser exemplificado como de produção puxada, é um pouco mais complexo do que isso, principalmente se tratando de pedidos que devem ser e entregues no mesmo dia.

O sequenciamento de produção nesse tipo de empresa é complexo, pois são muitas as máquinas que podem executar as tarefas e muitas vezes, não se faz um sequenciamento correto, o que é necessário para economizar tempo e atender os clientes mais rapidamente.

Foi possível exemplificar através de simulações de tempos de processamento, que a aplicação de heurísticas de sequenciamento é viável neste tipo de empresa, principalmente se tratando do fator tempo, em poucos segundos encontramos soluções factíveis aplicáveis no sistema de produção e ambiente de máquinas em questão.

Utilizando o modelo lingo, encontramos a solução ótima para instâncias com menor número de máquinas e tarefas, também foi possível realizar um comparativo entre a solução ótima e a solução encontrada pela heurística (coisa que não foi possível fazer em instâncias maiores) e concluímos que a heurística em questão apresenta uma solução muito boa, pois em uma das cinco instâncias analisadas, a heurística encontrou um resultado ótimo.

A heurística apresentada é aplicável para um grande número de máquinas e tarefas, sendo assim, pode ser utilizada não somente nestas empresas em questão, mas em todas que fazem o uso do ambiente de máquinas denominado máquinas paralelas não-relacionadas.

Foram utilizadas instâncias diversas em relação aos tempos de processamento e máquinas mais rápidas, o desempenho da heurística foi similar para todas as instâncias utilizadas. Apenas a solução inicial apresenta deficiências em instâncias em que uma máquina é mais rápida do que as outras, a heurística de melhoria aloca tarefas para outras máquinas, não permitindo que nenhuma máquina fique sem tarefas para processar, isso garante a eficiência dessa heurística em uma variedade muito grande de máquinas e tarefas, encontradas em diversos casos reais de sequenciamento.

Em pesquisas futuras, podem ser observados outros tipos de empresa que fazem uso deste ambiente de máquinas ou mesmo incluir novas constantes no sistema, como datas de entrega, atrasos, adiantamento, atrasos, *flowtime*, número de tarefas atrasadas ou outros interesses que possam depender da ordem em que as tarefas são processadas e em quais máquinas elas são processadas.

6 REFERÊNCIAS

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. **Pesquisa Operacional**. São Paulo: Editora CAMPUS, 2007.

BAKER, Kenneth R, TRIETSH, Dan. **Principles of Sequencing e Scheduling**. Wiley Publishing, 2009.

CAMPOS, V. F. **Gerenciamento pelas diretrizes**. 2. ed. Belo Horizonte: Fundação Christiano Otoni, Escola de Engenharia da UFMG, 1996.

CHIAVENATO, Idalberto. **Administração da produção**: uma abordagem introdutória. Rio de Janeiro: Campus, 2005.

CORRÊA, H. L.; CORRÊA, C. A. **Administração da produção e operações – manufatura e serviços**: uma abordagem estratégica. São Paulo: Atlas, 2004.

DONAIRE, Denis. **Gestão Ambiental na Empresa**. São Paulo: Atlas, 1995.

ESPÍRITO SANTO, Laura Cristina. **Algoritmos Genéticos Aplicados ao Sequenciamento de Máquinas**. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade Estadual de Londrina, Londrina-PR, 2014

FERNANDES, F.C.F.; FILHO, M.G. **Planejamento e controle da produção**: dos fundamentos ao essencial. São Paulo: Editora Atlas, 2010.

GIGANTE, R.L. **Heurística construtiva para a programação de operações flowshop permutacional**. Dissertação (Mestrado em Engenharia de Produção). Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

GIL, Antonio Carlos. **Métodos e Técnicas de Pesquisa Social**. São Paulo: Editora Atlas, 1994.

LOPES, J. **Análise e Otimização do Sequenciamento de Produção de uma Empresa de Médio Porte de Embalagens Plásticas.** Juiz de Fora. Universidade Federal de Juiz de Fora (monografia), 2008.

LOPES, R., MICHEL, M. **Planejamento e controle da produção e sua importância na administração**. Revista científica eletrônica de ciências contábeis. Ano V – Número 09, 2007.

MACCARTHY, B. L.; LIU, J. Y. **Adressing the gap in scheduling research**: a review of optimization and heuristic methods in production scheduling. International Journal of Production Research, London, v. 31, n. 1, p. 59-79, 1993.

MARTINS, Petrônio Garcia, LAUGENI, Fernando Piero. **Administração da Produção**. 1º Edição, São Paulo: Saraiva, 1999.

MOREIRA, Daniel Augusto. **Administração da Produção e operações**. São Paulo: Pioneira Thonson Learning, 1999.

NAMBIAR, J. M. et al. **A Large Scale Location-Allocation Problem in the Natural Rubber Industry**. Europear Journal of Operational Reseach, v.6, p. 183-189, 1981.

OHNO, T. **O sistema toyota de produção** – Além da produção em larga escala. Bookman, 1997.

PINEDO, M.L. **Scheduling: theory, algorithms and systems**. New Jersey, Prentice-Hall. 4ª ed, 2010.

SHINGO, SHINGEO. **O Sistema Toyota de Produção**. 2. ed. Porto Alegre: Artes Médicas, 1996.

SLACK, N. **Vantagem competitiva em manufatura**. São Paulo: Atlas, 2002.

TUBINO, Dalvio Ferrari. **Manual de Planejamento e controle da Produção**. São Paulo: Editora Atlas S.A 2º Edição 2000.

VOLLMANN, T. E.; BERRY, W.L.; WHYBARK, D.C. e JACOBS, F.R.. **Sistemas de planejamento e controle da produção para o gerenciamento da cadeia de suprimentos**. 5. ed. Porto Alegre: Bookman, 2006.

WOMACK, J. P.; JONES, D. T. **A mentalidade enxuta nas empresas**: elimine o desperdício e crie riqueza. 5 ed. Rio de Janeiro: Campus, 1998.

APÊNDICES

APÊNDICE 1

```
SETS:

n/1..8/: y;
m /1 2/;
n0 / 1 2 3 4 5 6 7 8 0 /;
matriz (n0,n0,m):x;
matrizT (n, m):P;
matrizH (n0,n0):C,w;

ENDSETS

DATA:

P = 49 30,
 57     30,
 41     34,
 59     35,
 58     39,
 57     40,
 52     23,
 47     34;

G=10000; !número grande

ENDDATA

!função objetivo;

MIN = Cmax; !makespan;

!RESTRIÇÕES;

@FOR(n(j):@SUM (m(k):@SUM (n0(i):x(i,j,k)))=1);
@FOR(m(k):@SUM (n(j):x(9,j,k))<=1);
@FOR(n(h): @FOR (m(k):@SUM (n0(i) | i#NE#h: x(i,h,k)) - @SUM ( n0(j) | j#NE#h:
x(h,j,k)) = 0));
@FOR (n0(i):@FOR (n(j): @FOR (m(k): C(j,k) >= C(i,k) - G+ (p(j,k) + G)*x(i,j,k))));
@for (n0(i): @for (n0(k): Cmax >= c(i,k))); !makespan;

!Tipos de Variaveis;

@for(n(i): @BIN (y(i)));
@FOR(n0(i):@FOR(m(k):(C(i,k))>=0));
@FOR(n0(i):@FOR (n0(j):@FOR(m(k):@BIN (x(i,j,k)))));
@FOR(n0(i):@FOR(m(k):@free(C(i,k))));
```

APÊNDICE 2

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>
#include <math.h>
typedef struct{
int ValorInteiro;
float ValorReal;
} Estrutura;
typedef Estrutura *pEstrutura;
FILE *Arquivo;
int NumEleA, NumeroTarefa, NumeroMaquina, m , n, a,b,c,d,e,f,menor, mks, loop, mksteste, melhoria;
char Instancia[30];
int MatrizTempoProcessamento[1000][200], Vetorposicao[1000], Vetorposicaoteste[1000],makespan[1000], makespanteste[1000];
void LerArquivo(){ //funcao que le a instancia a ser resolvida. */
int i,j,k;
 Arquivo=fopen(Instancia,"r");
 fscanf(Arquivo,"%d",&NumeroTarefa); // Le numero de tarefas da instancia.
 fscanf(Arquivo,"%d",&NumeroMaquina); // Le numero de maquinas.
for (i=0;i<NumeroTarefa;i++){  // Lê os tempos de proc no arq e guarda na matriz.
for (k=0;k<NumeroMaquina;k++){
fscanf(Arquivo,"%d",&MatrizTempoProcessamento[i][k]); } }
printf("\nInstancia lida:\n", i); // Imprime a instancia na tela:
printf(" Tarefas: %d\n",NumeroTarefa);
printf(" Maquinas: %d\n",NumeroMaquina);
printf(" Tempos de processamento: \n");
for (i=0;i<NumeroTarefa;i++){  // Mostra os tempos de proc e guarda na matriz.
 for (k=0;k<NumeroMaquina;k++){
 printf(" %d",MatrizTempoProcessamento[i][k]); }
printf(" \n"); }
printf(" \n");
fclose(Arquivo); }
int main(){
int i,j,k;
int VetorB[1000]; // Vetor de inteiros com 1000 posições (0 a 999)
Estrutura VetorC[1000]; // Vetor de elementos do tipo Estrutura com 1000 posições
Estrutura VetorA[4];
// LENDO INSTANCIAS:
for (i=1;i<2;i++){ //***ALTERAR PARA LER OUTRA INSTÂNCIA
sprintf(&Instancia[0],"Inst%d.txt",i+1); // Alterar aqui o nome da instancia a ser lida.
printf("\n\nInstancia: %d", i+1);
 LerArquivo(); }
for (m=0;m<NumeroMaquina;m++){
makespan[m] = 0; }
//PARTE 1 ^^^^^^^^^^^^^^^^^^^^^^^^
```

```
46.  for (n=0;n<NumeroTarefa;n++){ //COLOCA A TAREFA NA MÁQUINA QUE ELA
POSSUI MENOR TEMPO DE PROCESSAMENTO
47.  menor = 9999;
48.  for  (m=0;m<NumeroMaquina;m++){
49.  if (MatrizTempoProcessamento[n][m] < menor) {
50.  menor = MatrizTempoProcessamento[n][m];
51.  Vetorposicao[n] = m; }}}
52.  for (n=0;n<NumeroTarefa;n++){ //calcula os makespan para cada uma das máquinas
53.  for  (m=0;m<NumeroMaquina;m++){
54.  if (Vetorposicao[n] == m){
55.  makespan[m] = makespan[m] + MatrizTempoProcessamento[n][m];}}}
56.  for  (n=0;n<NumeroTarefa;n++){
57.  Vetorposicaoteste[n] = Vetorposicao[n];
58.  printf("\no vetor posição da tarefa %d é: %d", n, Vetorposicao[n]); }
59.  for (m=0;m<NumeroMaquina;m++){ //joga os makespans das máquinas na tela
60.  printf("\no makespan da maquina %d é: %d", m, makespan[m]); }
61.  mks = 0;
62.  for (m=0;m<NumeroMaquina;m++){
63.  makespanteste[m] = 0;
64.  if(makespan[m] >= mks) {
65.  mks = makespan[m]; }}
66.  printf("\no makespan geral eh %d", mks); //imprime o makespan da solução inicial
67.  mksteste = 0; //define o mks teste como zero
68.  melhoria = 1; //define melhoria como 1 (pra entrar no loop)
69.  loop = 0;
70.  //PARTE 2 ^^^^^^ FAZ MELHORIAS NO ALGORITMO ANTERIOR
71.  while (melhoria == 1) { //função melhoria
72.  melhoria = 0;
73.  for (n=0; n<NumeroTarefa;n++){ //for para cada uma das tarefas
74.  for (m=0; m<NumeroMaquina;m++){ //for para máquinas
75.  Vetorposicaoteste[n] = m;
76.  for (a=0;a<NumeroTarefa;a++){  //calcular os makespanteste p cada maquina
77.  for  (b=0;b<NumeroMaquina;b++){
78.   if (Vetorposicaoteste[a] == b){
79.  makespanteste[b] = makespanteste[b] + MatrizTempoProcessamento[a][b];}}}
80.  for (c=0;c<NumeroMaquina;c++){
81.  if(makespanteste[c] >= mksteste) {
82.  mksteste = makespanteste[c]; }}
83.  printf("\no mksteste calculado eh %d", mksteste);
84.   if (mksteste < mks) {
85.   melhoria = 1;
86.   mks = mksteste;
87.  for (d=0;d<NumeroMaquina;d++){
88.   makespan[d] = makespanteste[d];}
89.  for  (e=0;e<NumeroTarefa;e++){
90.    Vetorposicao[e] = Vetorposicaoteste[e];} }
91.   else {
92.  for  (e=0;e<NumeroTarefa;e++){
93.    Vetorposicaoteste[e] = Vetorposicao[e]; }}
94.  for (f=0;f<NumeroMaquina;f++){ //coloca makespanteste como zero
```

```
95.      makespanteste[f] = 0 ;          }
96.     mksteste = 0;
97.     }}}
98.     //IMPRIME O RESULTADO FINAL NA TELA
99.     for  (n=0;n<NumeroTarefa;n++){ //printa na tela o vetorposicao e o makespan
100.            printf("\no vetor posição da tarefa %d é: %d", n, Vetorposicao[n]); }
101.    for (m=0;m<NumeroMaquina;m++){
102.            printf("\no makespan da maquina %d é: %d", m, makespan[m]);}
103.    mks = 0;
104.    for (m=0;m<NumeroMaquina;m++){
105.            makespanteste[m] = makespan[m];
106.            if(makespan[m] >= mks) {
107.                    mks = makespan[m]; }}
108.            printf("\no makespan do sistema eh %d", mks);
109.     return 0; }
```

OBSERVAÇÃO: Esse código foi adaptado do código em linguagem de programação C apresentado pelo professor Everton Luiz de Melo na disciplina de Tópicos Avançados em Planejamento e Controle de Produção (EP04P), ministrada no 2º semestre de 2015 para o curso de Engenharia de Produção da Universidade Tecnológica Federal do Paraná – Campus Ponta Grossa. A heurística de melhoria também foi apresentada nessa disciplina pelos alunos Amanda Thais Rocha (autora desse Trabalho de Conclusão de Curso) e Leonardo Giovanetti Rossi como requisito parcial para aprovação na disciplina através de um relatório técnico não publicado.