Pontifícia Universidade Católica de Goiás

Programa de Mestrado em Engenharia de Produção e Sistema

Escalonamento Da Produção Para Sistema De Manufatura Job Shop Com Parâmetros Inteligentes

Dayvid Wesley Pereira Martins

Abril de 2018

Escalonamento Da Produção Para Sistema De Manufatura Job Shop Com Parâmetros Inteligentes

Dayvid Wesley Pereira Martins

Dissertacão de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para obtenção do título de Mestre em Engenharia de Produção e Sistemas.

Orientadora: Profª. Dra Maria José Pereira Dantas.

Goânia-Go,

Abril de 2018

Dados Internacionais de Catalogação da Publicação
(Sistema de Bibliotecas PUC Goiás)

M386e Martins, Dayvid Wesley P.

Escalonamento Da Produção Para Sistema De Manufatura Job Shop Com Parâmetros Inteligentes [recurso eletrônico]/ Dayvid Wesley Pereira Martins. – 2018.

90f.; il.

Texto em português com resumo em inglês

Dissertação (mestrado) – Pontifícia Universidade Católica de Goiás, Programa de Pós-Graduação Stricto Sensu em Engenharia de Produção e Sistemas, Goiânia, 2018.

Inclui referências f. 79-85

1. *Scheduling*, 2. Aplicativo *web*, 3. *Benchmark sets*, 4. Algoritmo genético, 5. Otimização heurística. I. Dantas, Maria José P. . II. Pontifícia Universidade Católica de Goiás. III. Título

CDU: 658.62(043)

Escalonamento Da Produção Para Sistema De Manufatura Job Shop Com Parâmetros Inteligentes

Dayvid Wesley Pereira Martins

Esta Dissertação foi julgada adequada para obtenção do título de Mestre em Engenharia de Produção e Sistemas, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica de Goiás em abril e 2018.

Prof. Marcos Lajovic Carneiro, *Dr.*
Coordenador do Programa de
Pós-Graduação
em Engenharia de Produção e Sistemas

Banca Examinadora:

Profª. Dra Maria José Pereira Dantas.
Orientadora

Prof. Dr. Ricardo Luiz Machado, *Dr.*
PUC-Goiás – Membro

Prof. Dr. Iran Martins do Carmo, *Dr.*
IFG – Membro

Goânia-Go
Abril de 2018

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

AGRADECIMENTOS

A Deus, princípio e fundamento de toda a existência, pelas bênçãos e oportunidades recebidas.

Aos meus familiares que sempre estiveram presentes nos momentos difíceis.

Aos professores que se dedicam arduamente a tarefa de formar profissionais preparados para atuação no mercado de trabalho.

À minha orientadora, Profª. Dra Maria José Pereira Dantas, pelas sabias orientações e diretrizes que contribuíram para realização deste trabalho.

À FAPEG, por fomentar esta pesquisa através da concessão da bolsa de estudos.

"Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)

RESUMO

Resumo da Dissertação apresentada ao MEPROS/ PUC Goiás como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia de Produção e Sistemas (M.Sc.)

Escalonamento Da Produção Para Sistema De Manufatura Job Shop Com Parâmetros Inteligentes

Dayvid Wesley Pereira Martins

Abril de 2018

Orientadora: Profª. Dra Maria José Pereira Dantas.

RESUMO: A elaboração de um processo de planejamento eficaz, para a sequência de processamento de ordens de produção na programação de sistemas de manufatura, é uma tarefa com alto grau de complexidade. A ausência de plataformas para experimentações simuladas da escala de produção, qualquer que seja a tipologia: *flow shop, job shop, open shop,* dificulta a curva de aprendizagem metodológica para este tipo de problema. Este trabalho propõe a concepção de um aplicativo web, que implementa um algoritmo genético `(AG)` personalizado para minimizar o *makespan* (tempo de finalização), de modo a permitir experimentações simuladas dos *benchmark sets* de escalonamento da produção em sistemas de manufatura do tipo *job shop*. O aplicativo web está disponível em <http://iproductionscheduling.com> e foi desenvolvido utilizando as linguagens Python e HTML5. Desse modo, é possível realizar online simulações otimizadas da escala de produção de instâncias do tipo `abz, dum, ft, yn, la, orb, swv` e `ta`, seguindo a premissa que o *job* emerge segundo uma ordem de produção emitida com especificações de roteiro de fabricação e tempo de processo com particularidades próprias contidas em um *benchmark set*. Os operadores genéticos propostos (crossover por roleta e mutações) foram adaptados para promover a intensificação e exploração no espaço de busca. Utilizou-se o elitismo e imigrantes aleatórios como técnica de controle da diversidade populacional. Na fase de ensaios, os operadores genéticos foram testados de forma isolada com a instância `abz5` 10×10 para verificar o impacto de diferentes variações nos parâmetros do `AG` no resultado esperado. Após isto, o aplicativo foi avaliado a partir de duas instâncias, sendo a `abz5` 10×10 e `ft06` 6×6, com resultados compatíveis aos da literatura recente, obtidos por outros métodos heurísticos. As experimentações realizadas comprovaram que o algoritmo implementado no núcleo da página *web*, se aproxima dos atuais limites ótimos e acrescenta quando disponibiliza um ambiente de experimentação e mostra os resultados do escalonamento em Gráficos de Gantt, além de apresentar tabelas e gráficos para avaliação do processo de otimização com os parâmetros determinados pelo usuário.

Palavras-chave: *Scheduling*, Aplicativo *web*, *Benchmark sets*, Algoritmo genético, Otimização heurística.

ABSTRACT

Abstract of Dissertation presented to the MEPROS/ PUC Goiás as part of the requirements required for obtaining a Master's degree in Engineering of Production and Systems (M.Sc.)

Production Scheduling For Manufacturing System Job Shop With Intelligent Parameters

Dayvid Wesley Pereira Martins

Abril de 2018

Advisor: Profª. Dra Maria José Pereira Dantas.

Abstract: The development of an effective planning process for the sequence of processing orders in manufacturing systems programming is a task with a high degree of complexity. The absence of platforms for experimentation simulation of the scale of production, whatever the typology: flow shop, job shop, open shop, hampers the methodological learning curve for this type of problem. This one work proposes the design of a web application, which implements a genetic algorithm `(GA)` to minimize the makespan (completion time), to allow simulation of benchmark sets of production scheduling in job shop manufacturing systems. The web application is available at <http://iproductionscheduling.com> and was developed using Python and HTML5 languages. In this way, it is possible to carry out optimized simulations of the instances of the type `abz, dum, ft, yn, la, orb, swv` and `ta`; following the premise that the job emerges according to a production order issued with manufacturing schedule and time specifications with particularities contained in a benchmark set. The genetic operators (roulette crossover and mutations) were adapted to promote intensification and exploration in the search space. Elitism and random immigrants were used as a technique for controlling population diversity. In the testing phase, were tested in isolation with the `abz5` 10×10 of different variations in `GA` parameters in the expected result. After this, the application was evaluated from two instances, `abz5` 10×10 and `ft06` 10×10, with results compatible with those of the recent literature, obtained by other heuristic methods. At Experiments carried out proved that the algorithm implemented in the core of the page the current optimal limits and adds when it provides experimentation and shows the results of the Gantt chart, in addition to shown tables and graphs to evaluate the optimization process with the parameters determined by the user.

Key words: Scheduling, Web Application, Benchmark sets, Genetic Algorithm, Optimization Heuristic.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

JSS	*Job Shop Schedule*
JSP	*Job Shop Problem*
MRP	*Material Requirement Planning*
MRPII	*Manufacturing Resource Planning*
ERP	*Enterprise Resource Planning*
APS	*Advanced Planning & Scheduling*
MES	*Manufacture Execution System*
PCP	*Planejamento e Controle da Produção*
OP	*Ordem de Produção*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo, são apresentados os aspectos que suscitaram a elaboração deste trabalho e os objetivos que o direciona. Relata-se e caracteriza o problema, justifica o tema escolhido.

1.1 Caracterização do Problema

Segundo Martins e Laugeni (2015) a função produção pode ser compreendida como um conjunto de processos com a capacidade de transformar um bem tangível em outro de maior utilidade. O atual modelo de consumo exige grandes variedades de produtos com vida útil reduzida ao menor custo possível. Para que esta necessidade seja atendida é necessário um sistema de produção organizada com processos cada vez mais complexos e interativos, buscando formas para economizar em cada possível detalhe com o claro objetivo de se obter custos competitivos.

A ascensão de um paradigma de manufatura interativo concerniu o processo de transformação ao qual a indústria vem sofrendo. Conceitos como *internet* das coisas, Internet industrial, fabricação baseada em nuvem e a fabricação inteligente, são os pilares da indústria 4.0. Neste contexto, o problema de escalonamento é uma constante em indústrias de manufatura e em empresas de serviços que objetivam a redução dos custos operacionais de modo a elevar sua produtividade. Portanto, preestabelecer e determinar etapas para escalonar tarefas em sistemas produtivos reais é uma atividade extremamente necessária em um modelo de produção organizada.

Para Harjunkoski *et al.* (2014), são quatro as principais questões decisórias envolvidas no processo de programação/escalonamento da produção: i) quais tarefas executar? ii) onde processar as tarefas de produção (atribuição de tarefas a Recursos)? iii) em que sequência produzir (sequenciamento de tarefas)? iv) quando executar as tarefas de produção (tempo das tarefas)?. Dessa forma, o ideal é que estas decisões sejam tomadas simultaneamente, pois geralmente estão fortemente vinculadas, devido a sincronização da utilização de recursos.

O escopo de decisões simultâneas em problema escalonamento existe não apenas em determinados tipos de ambientes de fabricação, mas também nos setores de serviços e gerenciamento, áreas de computação, entre outros. A programação cíclica está envolvida em inúmeras aplicações do mundo real, sendo um problema complexo, de natureza combinatória. A nível operacional, o escalonamento de produção é uma das atividades mais importantes de uma empresa, pois garante a competitividade em um mercado consumidor exigente. Está relacionado à otimização de várias medidas de desempenho, visando o bom funcionamento do sistema e a satisfação do cliente, tais como: uso eficiente de

recursos, entrega de produtos em prazos determinados e redução de custos de produção (FUCHIGAMI; RANGEL, 2017; KECHADI; LOW; GONCALVES, 2013).

O escalonamento de produção refere-se à tomada de decisão relativa à designação de empregos para os recursos disponíveis e sua ordem posterior para otimizar medidas de desempenho pré-definidas. As restrições impostas ao fluxo de escalonamento é geralmente determinada por uma das tipologias clássicas para sistema de manufatura: *flow shop*, *job shop*, *open shop*, *job shop*; [1] as inúmeras variações existentes costumam ser, na verdade, extensões dos respectivos modelos clássicos.

Através de diversificados desfechos, sempre apresentou grande potencial de pesquisa e aplicação com relevância no mundo real. Desse modo, o interesse continuo de uma significante parcela de pesquisadores em áreas como sistemas de manufatura, planejamento e controle da produção, planejamento de recursos computacionais, logística, tem se mantido ao longo das últimas décadas. Foi abordado por pesquisadores pioneiros Manne (1960) Taillard (1993) e Maccarthy e Liu (1993), e mais recentemente, Güçemir e Selim (2017), Shen, Liji; Dauzère-pérès (2017), Ham (2017) e Hoorn (2018) dentre outros, sob perspectivas diversificadas; clássica, extensões flexíveis, mista ou personalizadas.

Devido à ampla gama de problemas de escalonamento, várias abordagens foram desenvolvidas para produzir melhores soluções. Dentre estas abordagens estão a programação manual suportada por computador (por exemplo, gráficos de Gantt interativos), programação matemática (PL[2], PI[3], PD[4]), métodos heurísticos, algoritmos evolutivos e inteligência artificial, entre outras. A maioria destes métodos é frequentemente apresentada na literatura de um ponto de vista de modelagem conceptiva, em que são experimentados exemplos em escala reduzida (HARJUNKOSKI *et al.*, 2014).

As práticas de engenharia e gestão exigem lidar com a incerteza e identificar fatores que interagem e influenciam o ambiente ao qual se está inserido, para então predizer a eficácia das etapas predefinidas no escopo de um planejamento com coordenação interativa entre diferentes níveis de operação. Por razões de economia de escala, o correto é explorar ao máximo a capacidade de oferta do sistema frente as subsequentes demandas. Mediante este cenário, um planejamento eficaz, em conformidade com capacidade produtiva do sistema, porém flexível para adaptar-se às intempéries de instabilidade situacionais de modo a não violar restrições de capacidade sistêmicas e manter a otimização dos custos, é dependente de ferramentas computacionais e elevado grau de conhecimento técnico (EROL *et al.*, 2016; EROL; SIHN, 2017).

Usualmente a programação da produção, escalonamento ou sequenciamento da

[1] ***flow shop, job shop, open shop*:** abordado na seção 2.2
[2] **PL:** Programação Linear
[3] **PI:** Programação Inteira
[4] **PD:** Programação Dinâmica

produção é obtida da análise de informações pertinentes ao controle de produção que está geralmente a cargo do analista de PCP [5]. Porém, existe um *gap* entre os modelos teóricos e os modelos implementados no mundo real. Em parte esta lacuna se dá devido a falta de flexibilização dos modelos computacionais em relação a alguma restrição específica do sistema produtivo. Entretanto, houve avanços significativos na ultima década, e os modelos computacionais efetivados ao núcleo dos *softwares* para escalonamento da produção (Exemplo: Preactor), são hoje auto-moldáveis para atender requisitos e características especificas de cada sistema produtivo. Portanto, o elevado grau de conhecimento técnico exigido na operacionalização de um *software* de escalonamento da produção, e as possíveis dificuldades na busca por esta qualificação, é o fator de maior preponderância na existência deste *gap*.

1.2 Justificativa

A curva de aprendizagem metodológica em problemas de escalonamento da produção, qualquer que seja a tipologia, é longa. Pois, replicar experimentações de otimização simulada da escala de produção, por meio de *benchmark sets* presentes na literatura é pouco acessível devido a ausência de plataformas para experimentações simuladas. Para além do conhecimento específico, tais experimentações costumam exigir o domínio de alguma linguagem de programação computacional. Este fato dificulta o aprendizado de um problema por si só complexo.

Devido a considerável atenção que esse tipo de problema recebeu nas últimas décadas, vários conjuntos de instâncias de *benchmark* estão disponíveis para comparar a qualidade dos diversos métodos desenvolvidos. Neste sentido, Hoorn (2018) contribui com estado da arte fornecendo uma visão geral, precisa do estado atual dos limites para oito conjuntos de instâncias de *benchmark sets*, realizando uma revisão cuidadosa das referências e mantendo uma página *web* com resultados atualizados.

Algumas publicações com resultados numéricos que indicam uma metodologia capaz de produzir soluções superiores, deixam uma aparente lacuna no que se refere á replicabilidade acessível das simulações apresentadas. Sendo este o caso de Kundakc e Kulak (2016) que apresenta metodologias eficientes de algoritmo genético híbrido `AG` para minimizar o tempo de conclusão da escala de produção e Bierwirth e Kuhpfahl (2017) que propõe uma heurística que minimiza o atraso total, com um *design* de concepção baseado em um modelo avançado de gráfico disjuntivo.

Este trabalho implementa uma metodologia heurística de otimização em um aplicativo *web* para realizar experimentações simuladas dos *benchmark sets* disponíveis na literatura. De modo semelhante à proposta de Hoorn (2018), pode auxiliar na aquisição

[5] **PCP:** Planejamento e Controle da Produção

do conhecimento metodológico para resolução desta categoria de problema. No entanto, existem fatores preponderantes que exercem influência direta na escolha tipológica do sistema produtivo a ser representado por um ambiente de experimentação simulada.

O estudo de problemas de otimização combinatória, exemplo: mochila, caixeiro viajante, escalonamentos, menor-caminho, corte de materiais, planejamento da produção, tem como pré-requisito a compreensão do modelo em sua forma clássica. Neste sentido, a designação clássica em problemas de otimização combinatória implica que esta é uma representação pertencente à classe primária do problema, ou seja, modelo preeminente do qual outros problemas correlatos derivam-se.

O *flow shop* e o *job shop* clássico são modelos representativos do ambiente operacional de empresas de manufatura com alto grau de extensões (problemas correlacionados). Ambos possuem roteiro de fabricação sequencial distinguindo-se apenas no que diz respeito ao volume, alto no *flow shop* e baixo no *job shop*, de produção e diversidade, baixo no *flow shop* e alta no *job shop*, de produtos. Entretanto, na concepção de algoritmos para resolução deste tipo de problema deve ser levado em consideração que um algoritmo de escalonamento *job shop* tem a capacidade de escalonar funções *flow shop*. Porém, o inverso não é verdadeiro.

Diante do que foi exposto, ofertar pequenos aplicativos com acessibilidade facilitada, para elucidar o estudo metodológico do sequenciamento otimizado da produção para sistemas de produtivos com fluxo de trabalho *job shop* clássico, é plenamente plausível.

1.3 Objetivo

O objetivo deste trabalho é conceber um aplicativo *web*, com a capacidade de realizar experimentações simuladas de *benchmark sets* disponíveis na literatura, para o escalonamento da produção em sistemas de manufatura do tipo *job shop*. Os objetivos específicos são descritos a seguir.

1. Propor e implementar um algoritmo genético `(AG)` personalizado, em linguagem Python, com operadores genéticos adaptados para atuarem sobre vetores que codificam as soluções factíveis do problema e minimizar o tempo de finalização (*makespan*);

2. Desenvolver e implementar a aplicação *web* com ferramentas *open source*;

3. Garantir que o algoritmo desenvolvido tenha capacidade de ser operado online e seja um ambiente de experimentação de escalonamento da produção à partir de *benchmark sets*, com a definição de parâmetros por parte do usuário.

O aplicativo procura simular e determinar a programação da produção (*scheduling*) com a premissa de que cada produto a ser fabricado emerge através de uma ordem de produção emitida com especificações e particularidades contidas em um *benchmark sets*.

1.4 Estrutura do Trabalho

A estrutura deste trabalho é composta por cinco capítulos:

- No segundo capítulo está realizada a revisão bibliográfica composta por base teórica crucial para o desenvolvimento da modelagem matemática e da rotina computacional.
- O terceiro capítulo evidencia a metodologia utilizada para atingir o objetivo proposto. Sua composição é pautada pela descrição da problematização e dos procedimentos adotados para atingir o objetivo desejado.
- No capítulo quatro apresenta-se a análise e resultados obtidos.
- No quinto capítulo estão expostas as conclusões referentes ao desenvolvimento deste trabalho.
- E por fim, as referências bibliográficas utilizadas ao longo do desenvolvimento do estudo.

2 REFERENCIAL TEÓRICO

Este capítulo irá aprofundar conhecimentos teóricos dos temas pertinentes ao parâmetro das incertezas existentes na otimização da instância tático/operacional de planejamento.

2.1 Planejamento e Controle da Produção

Foram utilizados com frequência, algumas terminologias fundamentais. Assim, seu entendimento é fundamental: Ordem de produção [1], *Setup*[2], *Makespan*[3].

A linha tênue que separa o planejamento de controle nem sempre é perceptível, Slack (2009) entende que o planejamento é a formalização do que se almeja realizar em um momento futuro, já o controle é o processo de lidar com as variações que podem ocorrer durante a implementação do planejamento, pois muito embora os planos sejam baseados em expectativas, este não garante que um evento vá de fato acontecer. Isto pode significar que os planos necessitem ser refeitos a curto prazo ou de uma intervenção para trazer de volta aos trilhos, uma operação.

Segundo Tubino (2009), o plano de produção trabalha com um horizonte de longo prazo, onde as incertezas são grandes, surge então a necessidade de uma dinâmica de replanejamento possível de ser empregada sempre que uma variável importante do plano se alterar. O planejamento hierárquico é desenvolvido em níveis, entre os quais normalmente se consideram o estratégico, o tático e o operacional. Cada nível orienta e restringe sucessivamente, o planejamento do nível imediatamente inferior (LUSTOSA *et al.*, 2013).

Decisões referentes a políticas de gestão, desenvolvimento organizacional, conciliação dos recursos internos de modo a atender os requisitos externos em paralelo aos objetivos da organização, estão relacionadas ao planejamento estratégico. Estas decisões são relacionadas ao plano das facilidades produtivas tais como: local, planta industrial, tamanho, equipamentos, linhas de produtos sistema logístico. A acertabilidade destas decisões define o índice de crescimento da empresa bem como sua competitividade de mercado e de fato pode determinar o êxito ou insucesso. Decisões referentes ao planejamento estratégico podem ser afetadas por informações internas e externas, envolvem índice de investimentos e possuem implicações a longo prazo. Por princípio são tomadas nos níveis hierárquicos mais elevados da administração (LUSTOSA *et al.*, 2013).

[1] **Ordem de produção:** documento com roteiro de fabricação e quantidade de um produto a ser produzida

[2] ***Setup:*** preparação de uma máquina para ficar apta a executar uma determinada tarefa

[3] ***Makespan:*** tempo total de produção de um cenário de escalonamento, calculado como o tempo decorrido entre o início do escalonamento das ordens de produção e o final da última operação da ordem escalonada mais tardiamente.

Decisões com foco em processo e na combinação de recursos estão relacionadas ao planejamento tático da produção. A alocação de recursos, tais como: capacidade de produção e armazenagem, é o aspecto base deste nível de planejamento. Tipicamente, este nível engloba as definições: turnos de produção (turno normal e extra ou subcontratação); alocação de capacidade para os produtos a se produzir segundo as demandas dos mesmos, formação ou não de estoques, escoamento, transporte e centros de distribuição da produção. Este horizonte de planejamento envolve decisões de médio prazo com foco em um grupo ou família de produtos que possua características de produção (LUSTOSA *et al.*, 2013).

Decisões relativas ao cotidiano operacional e sua programação as quais necessitam de uma desagregação plena das informações concebidas nos níveis superior, são pertencentes ao universo do controle operacional. É de responsabilidade deste nível, o sequenciamento da produção, atividade de controle de estoque, expedição e processamento de pedidos, administração de materiais, emissão e liberação de ordens, em suma, a operacionalização do "chão de fábrica". A programação da produção está encarregada de definir quanto e quando comprar, fabricar ou montar de cada item necessário a composição dos produtos acabados propostos pelo plano (FERNANDES; FILHO, 2010; TUBINO, 2009).

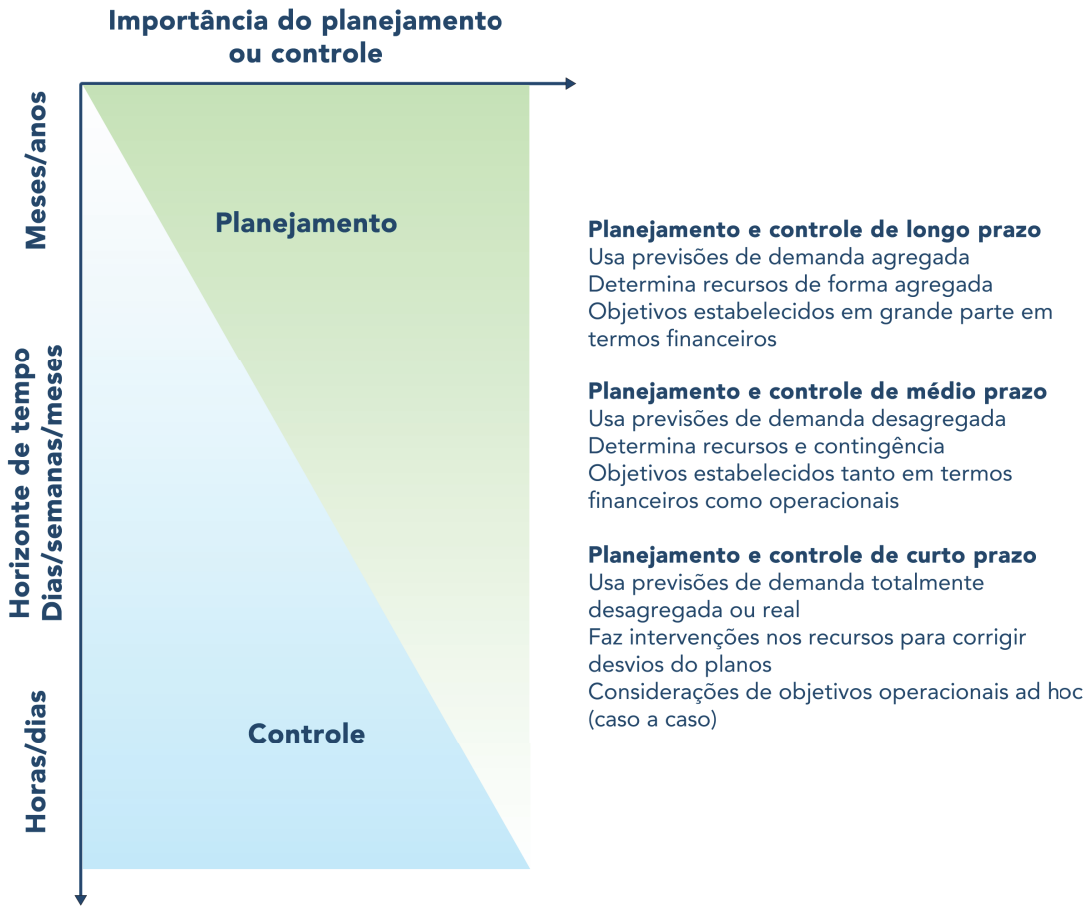
O ato de planejar, por princípio, exige antecipar as decisões referente ao posicionamento a ser tomado com a finalidade de alcançar um resultado futuro desejado. A abordagem hierárquica do planejamento da produção garante a divisão entre estas categorias, desta forma, torna se possível a fragmentação do processo de decisão, permitindo decomposição das decisões em problemas menores e ajustados ao contexto hierárquico de seu respectivo nível, garantindo o pleno relacionamento do nível mais alto com o nível mais baixo (LUSTOSA *et al.*, 2013).

O Planejamento e controle da produção é uma das operações fundamentais da função produção. A exigência desta atividade nas organizações acontece por ocasião da imprevisibilidade, tanto na capacidade produtiva como na variabilidade das demandas. Demanda e capacidade sofrem influências consideráveis de variáveis ambientais, bem como alocação e utilização dos recursos disponíveis, respectivamente. Por existir tal instabilidade as organizações precisam estabelecer um planejamento ajustado a um controle de produção, e então conciliar estes aspectos, Figura 1 . A estabilidade adaptada entre capacidade e demanda pode oportunizar altos lucros e a satisfação da clientela. Entretanto, o desequilíbrio destes aspectos é de predomínio desastroso, Slack (2009, p. 283–284) defende esta concepção.

> "A natureza do planejamento e controle muda ao longo do tempo. No longo prazo, gerentes de produção fazem planos relativos ao que eles pretendem fazer, que recursos precisam e quais objetivos esperam atingir. A ênfase é mais no planejamento, porque existe ainda pouco a ser controlado (...). No planejamento e controle de médio prazo refere-se a planejar em mais detalhe. Olha para frente para avaliar a demanda global que a operação deve atingir de forma parcialmente desagregada. (...). No planejamento a curto prazo, muitos recursos terão sido definidos e

será difícil fazer mudanças de grande escala nos recursos. Todavia, as intervenções de curto prazo são possíveis se as coisas não correm conforme os planos. Neste estágio, a demanda será avaliada de forma totalmente desagregada. (...) os aspectos de planejamento e controle variam em importância, conforme à proximidade da data do evento."

Figura 1 – Atividade de Planejamento e Controle da Produção



Fonte: Adaptado de Slack (2009)

As operações fundamentais do planejamento e controle da produção são vinculadas aos custos, capacidade, tempo, qualidade e suas respectivas limitações. Tem-se como princípio que cada recurso disponível possui um custo, e que cada produto, via de regra, deve ser fabricado anexado a um orçamento pré-definido. Como todos produtos possuem necessidade de ser fabricados dentro de um prazo estabelecido, há então limitação de tempo, para que este represente valia significativa para o cliente (SLACK, 2009).

O PCP se segmenta em três níveis, conforme os horizontes de planejamento: longo, médio e curto prazo. Na literatura, Nogueira (2010), Fernandes e Filho (2010), Tubino (2009) há denominações diferentes para os distintos horizontes, apesar da variabilidade

denominativa, os cernes segmentários aplicados pelos vários autores são similares a nomenclatura:

- **Planejamento da Capacidade** tem por objetivo conciliar a demanda e o fornecimento a longo prazo, a capacidade de um processo é uma medida da quantidade que pode ser produzida, sendo está expressa em razões. A capacidade projetada ou capacidade do projeto de uma planta industrial é a capacidade máxima sob condições ideais de operação (LUSTOSA *et al.*, 2013).

- **Planejamento Agregado** tem por objetivo a obtenção de um plano de produção para cada família de produto, esta é a razão do nome agregado pois o planejamento agregado não é feito para cada tipo específico de produto e sim os tipos de produtos com características semelhantes são unidos em famílias. Esta agregação proporciona um erro menor em termos de produção garantindo que a demanda prevista seja atendida e que os custos envolvidos sejam minimizados. Geralmente, diligencia-se com um horizonte de médio prazo, planejamento de 3 meses a 1 ano. Alternativas flexíveis com a modificação da demanda, utilizar hora extra, geração e utilização estoques, subcontratar, contratar ou demitir, são opções a serem utilizada neste planejamento (FERNANDES; FILHO, 2010).

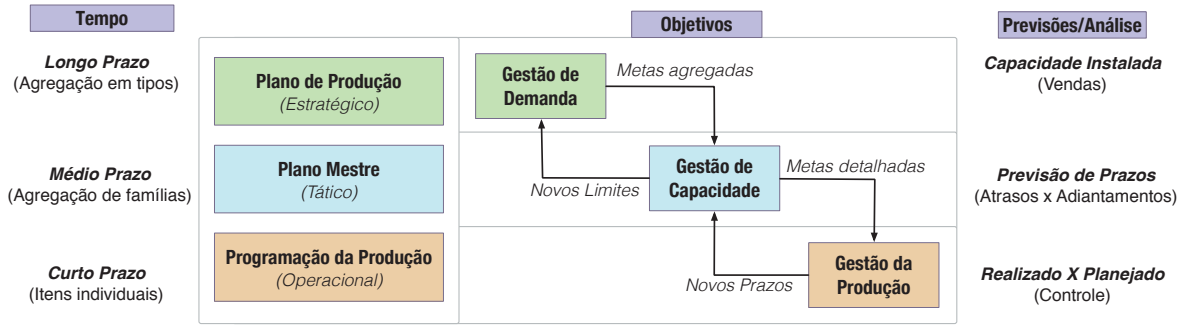
- **Programa ou Plano Mestre da Produção** pode ser gerado a partir da desagregação de plano agregado, sendo este a primeira atividade do controle da produção e tem por objetivo estabelecer quais famílias produtos serão fabricados em um determinado período de tempo e em que quantidades. Seu horizonte é de curto prazo, uma vez que geralmente é impossível controlar o fluxo de materiais baseando se em um plano mestre de médio prazo, pois este sofreria mudanças radicais que acabaria se tornando inútil na prática (FERNANDES; FILHO, 2010).

De um modo geral, as decisões que diligenciam um horizonte maior de planejamento exigem um grau de desagregação menor e reciprocamente (SLACK, 2009). As estimativas agregadas, pré-estabelecidas acerca de uma família ou conjunto de produtos, têm por tendência o surgimento de erros alusivamente menores do que as estimativas desagregadas. O motivo pelo qual isto se sucede é devido às estimativas especificas por produto, que ao ser executada denotam erros maiores e erros menores, a depender do produto. Contrariamente, quando se opta por uma estimativa agregada, há auto compensação entre os erros maiores e menores, isto tem como consequência uma estimativa mais precisa. Sendo possível garantir a aleatoriedade, quanto maior for o número de produtos a serem fabricados, mais se distribuem os erros. Entretanto, o grau de acertabilidade da estimativa é inversamente proporcional ao seu horizonte de planejamento. Isto é, um horizonte de planejamento mais extenso tem maior a probabilidade de erros (CORREA; GIANESI; CAON, 2014).

2.2 Escalonamento em Sistemas de Manufatura

Tradicionalmente, a atividade de planejamento em uma empresa ou organização em geral sucede-se em três diferentes instâncias: estratégico, tático e operacional, Figura 2. O planejamento estratégico é responsável por direcionar a missão da empresa em termos de ofícios e aspirações, a cargo da alta administração. O planejamento tático averigua alternativas viáveis para a realização da missão, preferencialmente de responsabilidade da média gerência. Finalmente, o planejamento operacional define cronogramas específicos de curto prazo e alvos mensuráveis, sendo executado pelo pessoal de supervisão (CARVALHO; SILVA; FERNANDES, 1998; DELLAGI; CHELBI; TRABELSI, 2017).

Figura 2 – Atividade de Planejamento e Controle da Produção



Fonte: Fundamentado em Tubino (2009), Fernandes e Filho (2010)

Entre as instâncias que sucedem os planos de produção, a tomada de decisão em nível operacional exige a interatividade entre setores diversos — marketing, compras, finanças e produção — fazendo-se necessário o uso de aplicações que encontre soluções otimizadas e torne possível a sua compreensão pelos diversos setores. Porém, a solução apresentada deve dispor da possibilidade de flexibilização segundo exigências setoriais e mudanças contextuais, tornando assim possível o seu cumprimento. A integração de aplicações em um ambiente interativo permite a manipulação das soluções encontradas e ainda avaliar possíveis cenários alternativos (GUERRINI; BELHOT; JÚNIO, 2014).

O escalonamento das ordens de produção é um problema estabelecido sob as instâncias tático/operacional do planejamento de produção, momento em que é necessário alocar os recursos produtivos — como máquinas e mão de obra — com objetivo de garantir a execução das ordens de produção, de modo a não violar as restrições de capacidade ofertada em cada etapa, processo ou recursos. Uma vez que as operações de monitoramento e escalonamento são mais objetivas em uma configuração de manufatura em fluxo, problemas de sequenciamento dentro de sistemas de manufatura costumam ser classificado segundo as seguintes características (PINEDO, 2012). Para classificar problemas de escalonamento Graham *et al.* (1979) e Maccarthy e Liu (1993) apresentam o fluxo de exe-

cução — roteiro de fabricação — dos *jobs* como fator classificatório. Dentre os problemas com múltiplos processos — etapas ou máquinas — de fabricação os seguintes fluxos são os que mais se destacam:

- ***Flow Shop:*** todos os *jobs* possuem o mesmo roteiro de fabricação com fluxo de trabalho processado sequencialmente na mesma ordem. Desta forma, todos os *jobs* compartilham as mesmas etapas de fabricação, não havendo espaço para personalização de produtos. Características: invariabilidade de produtos, alto volume de produção.
- ***Flow Shop* flexível:** a fabricação é dividida em processos. Cada processo com mais de uma máquina para execução dos *jobs*, de modo que no "Processo X "existe a possibilidade do *job* ser executado em qualquer uma das máquinas disponível para este processo. Semelhante ao *Flow Shop* clássico todos os *jobs* possuem o mesmo roteiro de fabricação, ou seja, inicia sua produção pelo processo A, depois B, e assim sucessivamente até que esteja concluso. Características: alta diversidade de produtos, baixo volume de produção.
- ***Job Shop:*** diferente do *Flow Shop*, os *jobs* são classificados por produtos ou famílias de produtos, possuindo portanto roteiros de fabricação distintos. Roteiros distintos permitem diferentes sequências entre os *jobs* e seus estágios de fabricação. Porém, cada *job* possui uma sequência de máquinas a seguir. Assim, um *job* pode ter como seu primeiro estágio a máquina 1 enquanto um outro *job* a máquina 4.
- ***Open Shop:*** cada *job* possui um roteiro de fabricação específico, semelhante ao *Job Shop*, porém este roteiro não é sequencial, ou seja se um *job* possui roteiro A,B,C não necessariamente A é o seu estágio inicial, podendo este ser fabricado no ordenamento B,C,A ou C,B,A ou A,B,C.
- ***Job Shop* flexível:** Ao contrário do *Job Shop* tradicional, onde cada estágio é executado por uma unica máquina, no *Job Shop* flexivel, um estágio pode ser processada por um conjunto de máquinas.Assim, o termo flexível aplica-se ao fato de ser possível usar máquinas distintas para processar alguns ou todos os estágios de um *job*.

O escalonamento tem objetivos que variam de forma independente, ao minimizar o tempo máximo de conclusão da manufatura pode-se permitir a entrega antecipada das ordens de produção pelo sistema produtivo, reduzindo os níveis de ociosidade minimizar os tempos de espera dos entre etapas de processamento aumenta-se a capacidade de absorção de demanda do sistema (FREITAG; HILDEBRANDT, 2016; GÜÇEMIR; SELIM, 2017).

O foco primordial deste trabalho é o *Job Shop clássico*. portanto, nas seções subsequentes os demais problemas de escalonamento não serão abordados.

2.2.1 Escalonamento Job Shop

O *Job Shop Schedule* (JSP)[4] e um problema clássico de escalonamento de tarefas e por sua relevância na indústria de manufatura tem sido estudado desde a revolução industrial, tendo sido amplamente investigado ao longo das últimas décadas. Ao produzir em escala, geralmente há um conjunto fixo de operações que devem ser processadas. Ao modelar isso como um *Job shop* clássico, é necessário incluir um elevado número repetições para uma mesma operação dentro do sequenciamento. O grande número de empregos individuais dificulta a resolução. Desse modo, o conjunto de operações devem ser sequenciado de forma cíclica (ELMI; TOPALOGLU, 2016).

O escalonamento de tarefas ou ordens de produção em um ambiente *job shop*, pode ser formalmente descrito como um arquétipo conceitual de pesquisa operacional. Na literatura diversos autores trabalharam o assunto de escalonamento *job shop* schedule e propuseram modelos de programação linear inteira mista, resultando em um problema claramente estabelecido, podendo ser definido da seguinte forma: seja um conjunto de ordens de produção (*jobs*) $J = \{1, ..., n\}$, em que cada ordem J_i possui um conjunto de operações — estágios — $O = \{1, ..., o\}$, que devem ser obrigatoriamente processadas em um roteiro pré-estabelecido na máquinas do conjunto $M = \{1, ..., m\}$, sendo i o índice da ordem de produção, j o índice da operação e l o índice da máquina, com tempo de processamento pré-estabelecido T_{ijl}. De modo a exemplificar, considere, por exemplo 5 tarefas e 3 máquinas, denotadas por 1,2 e 3 (GRAHAM *et al.*, 1979; LIAO; YOU, 1992; WAGNER, 1959) sujeito as seguintes restrições:

- A série de operações é estabelecida e determinada no instante de início da produção da fábrica;
- Nenhuma etapa da OP pode ser iniciada até que a etapa anterior para essa OP seja concluída.
- Um processo só pode executar uma etapa por vez;
- Uma etapa, uma vez iniciada, deve ser executada até sua conclusão.

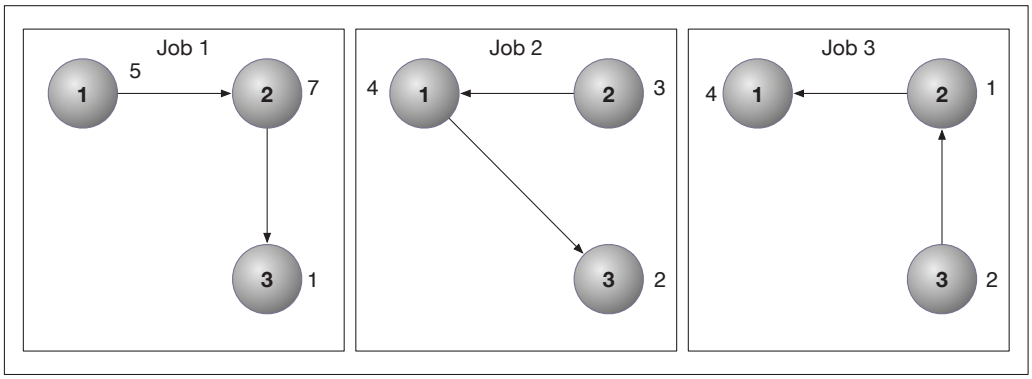
Para representar de forma matricial, considere as matrizes O e P, representam respetivamente a matriz de operações, e a matriz de tempos de processamento nessas máquinas:

$$O = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \\ 3 & 2 & 1 \end{bmatrix} \qquad P = \begin{bmatrix} 5 & 7 & 1 \\ 3 & 4 & 2 \\ 2 & 1 & 4 \end{bmatrix}$$

[4] **JSP:** *Job Shop Problem*

A primeira linha da matriz O indica que a tarefa 1 é processada nas máquinas 1, 2, 3, nesta ordem, com tempos de processamento de 5, 7 e 1 unidades de tempo respetivamente, correspondentes aos elementos da primeira matriz P, melhor compreendido nos grafos da Figura 3.

Figura 3 – Grafos *Jobs*



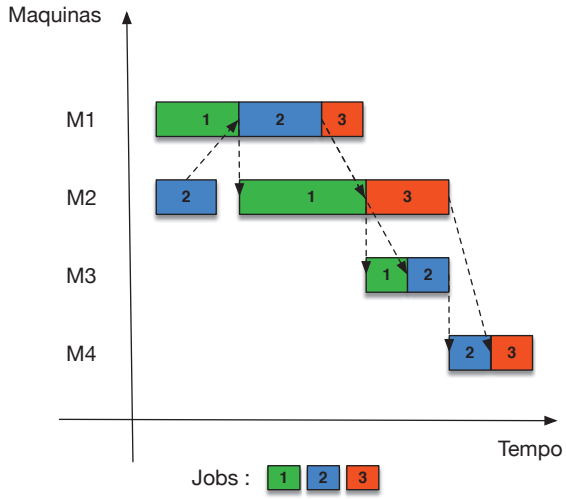
Fonte: Elaborado pelo autor (2017)

2.2.2 Representações Gráficas Para Escalonamento *Job Shop*

No escalonamento de produção, os métodos mais utilizados para representação gráfica são o Gráfico de Gantt, Figura 4, e o Grafo Disjuntivo, Figura 5, que permitem ilustrar de forma detalhada as sequências de produção. O gráfico de Gantt permite uma visualização gráfica e intuitiva de um possível sequenciamento. Enquanto o Grafo Disjuntivo, permite a modelagem matemática dos problemas de programação através da interface gráfica e possibilita o desenvolvimento de técnicas mais eficazes de solução exata e aproximada. O modelo de gráfico disjuntivo fornece um esquema de representação com estrutura de pesquisa local para problemas de escalonamento *job shop* por intermédio de uma estrutura gráfica chamada árvore crítica. Assim, efetivamente um algoritmo pode se orientar pelo *ranking* do caminho de forma adaptativa aos melhores tempos de processo (BIERWIRTH; KUHPFAHL, 2017; HAO *et al.*, 2017).

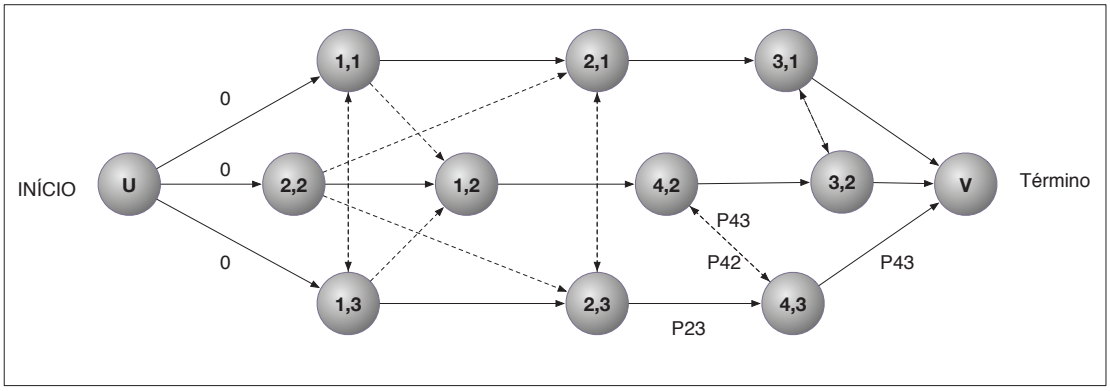
As arestas sólidas do grafo disjuntivo, representado pela Figura 5, são chamadas conjuntivas e representam a sequência de máquinas i, pré-estabelecidads no roteiro, para cada *job*. O valor da aresta é o tempo de processamento do *job* j na máquina i, para a operação (i, j). As arestas tracejadas são disjuntivas e representam escolhas para o sequenciamento de uma máquina, ligando todas aos *jobs* para uma determinada máquina. Uma vez definida a orientação da aresta disjuntiva, tem-se a sequência de operações que serão efetuadas em uma máquina (PINEDO, 2012; SOBEYKO; MÖNCH, 2016).

Figura 4 – Possível Escalonamento *Job Shop* em Gantt



Fonte: Adaptado de Pinedo (2012)

Figura 5 – Possível Grafo Disjuntivo *Job Shop*



Fonte: Adaptado de Pinedo (2012)

Existem várias formulações de programação matemática para a *job shop schedule* sem recirculação, incluindo uma série de formulações de programação inteira. No entanto, a formulação mais utilizada é a denominada formulação de programação disjuntiva. Essa programação disjuntiva está intimamente relacionada à representação gráfica disjuntiva da *job shop schedule*. Para apresentar a formulação de programação disjuntiva, a variável y_{ij} indicar o tempo de início da operação (i, j). O conjunto N designa o conjunto de todas as operações (i, j) e define `A` o conjunto de todas as restrições de roteamento $(i, j) \rightarrow (k, j)$ que exigem que o trabalho j seja processado na máquina i antes de ser processado na

máquina k. A Equação 2.1 minimiza o *makespan* (PINEDO, 2012).

$$Minimizar\ C_{max} \tag{2.1}$$

Sujeito a:

$$\begin{cases} y_{kj} - y_{ij} \geq P_{ij} & \forall\ (i,j) \rightarrow (k,j) \in A \\ C_{max} - y_{ij} \geq P_{ij} & \forall\ (i,j) \in N \\ y_{ij} - y_{il} \geq P_{ij} \text{ ou } y_{il} \geq P_{ij} & \forall\ (i,j) \text{ e } (i,l)\, i = 1, \ldots m \\ y_{ij} \geq 0 & \forall\ (i,j) \in N \end{cases}$$

Parâmetros:

A – Conjunto de pendências;

N – Conjunto de operações;

n – Número de tarefas;

m – Número de máquinas ou processos;

p_{ij} – Tempo de processamento da atividade i na máquina j.

Variáveis:

y_{ij} – Tempo inicial de produção da atividade i na máquina j;

C_{max} – *Makespan*.

Nesta formulação, o primeiro conjunto de restrições garante que a operação (k, j) não pode começar antes da operação (i, j) estar concluída. O terceiro conjunto de restrições é chamado de restrições disjuntivas; eles garantem que algumas ordens existem entre operações de diferentes trabalhos que precisam ser processadas na mesma máquina. Devido a essas restrições, esta formulação é referida como uma formulação de programação disjuntiva. O *Makespan* do modelo pode ser calculado através da soma dos tempos de processamento do caminho crítico, definido como o maior caminho entre o início de processamento das tarefas (n^0) até a conclusão de todas as tarefas em todas as máquinas (n^*). Se o gráfico contém um ciclo o caminho crítico se tornará infinito. Portanto, infactível, enquanto toda configuração sem ciclos representará uma solução factível do problema (BIERWIRTH; KUHPFAHL, 2017; PINEDO, 2012).

2.3 Pesquisa Operacional

Formalmente os primeiros indícios de atividades relacionado a pesquisa operacional (PO) acontecem durante a Segunda Guerra Mundial na Inglaterra, uma equipe de cientistas britânicos opta por tomar decisões com base cientificas sobre a melhor utilização dos recursos de guerra. Com o fim da guerra as ações antes propostas para ações militares são adaptadas para melhorar a eficiência do setor civil (TAHA, 2008)

A ciência aplicada cujo objetivo é a melhoria da performance em organizações e trabalha por meio da formulação de modelos matemáticos a serem resolvidos com o auxílio de computadores recebe o nome de Pesquisa Operacional (PO). A PO agrega em sua teoria quatro ciências fundamentais para o processo de análise e preparação de uma decisão: a economia, a matemática, a estatística e a informática (DÁVALOS, 2002).

Ao formular o planejamento e escalonamento da produção, surge a necessidade de escolher uma técnica apropriada associada ao grau de complexidade do modelo que representa o problema em questão. Há problemas quantitativos do tipo bem-estruturado e mal estruturado. Modelos do tipo mal estruturado são mais flexíveis em sua formulação. Entretanto, requerem uma heurística costumeiramente pouco precisa para solucionar os problemas. As soluções provenientes deste princípio criam cenários sujeitos a uma interpretação subjetiva daquele a cargo de tal tarefa, o qual, pode tomar decisões incorretas. Já os modelos do tipo bem-estruturados são mais rígidos em sua formulação matemática, porém as soluções geradas são precisas (SILVA; CEZARINO; RATTO, 2009).

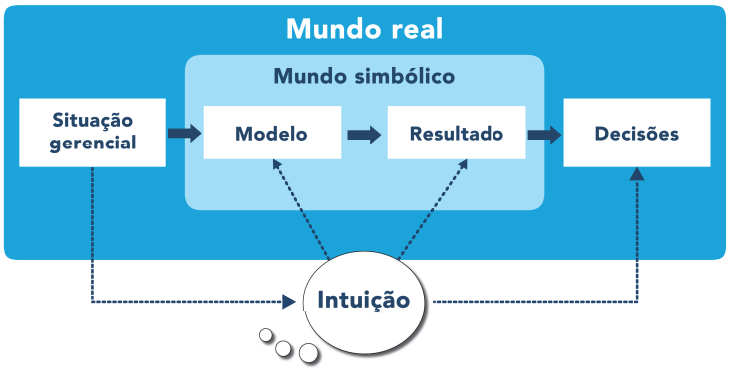
Sempre que uma organização ou empresa se propor a produzir algo em escala, surge adjunto desta preposição os problemas de *mix* de produção ou seja a fabricação de produtos diversos. Estas implicações evolvem decidir quais produtos e em que quantidade fabricar em um espaço de tempo. Sendo uma verdade factual a capacidade limitada de produção (recursos de matéria prima, recursos humanos, maquinário, estocagem, recursos financeiros, etc.) e o mix de produto que se pode fabricar, havendo assim a necessidade de determinar o que e o quanto fabricar de forma a minimizar os custos de produção e maximizar os lucros da empresa (ARENALES *et al.*, 2007).

Em problemas de planejamento agregado, no mundo real, pode ocorrer presença de incerteza referente ao otimizar de fato. Ao construir modelos de otimização é comum adotar a premissa que os elementos envolvidos no processo são conhecidos e então ignorar as influências que as incertezas associadas a este mesmo processo possa vir a causar na otimização das soluções. Para problemas com parâmetros incertos adota-se abordagem de metodologias clássicas que incluem análise de sensibilidade e otimização estocástica. Assim, obtém-se um modelo de otimização robusto capaz de abordar os parâmetros de incerteza por um aspecto de compreensão computacional (COLIN, 2013; FERNANDES; FILHO, 2010).

2.3.1 Programação Linear

A Programação Linear (PL) é sem dúvida uma das técnicas gerenciais mais poderosas, tradicionalmente usada para garantir lucratividade e sobrevivência de organizações industriais no longo prazo. Em suma, a PL arquiteta solucionar problemas de alocação ótima de recursos escassos, para execução de determinada atividade. Os recursos limitados representa a natureza de uma realidade existencial finita de recursos a qual se esta inseridos. Os modelos de otimização linear, ou seja, aqueles com representações simplificadas do comportamento de uma realidade, que objetive a otimização expressa na forma de equações matemáticas lineares as quais tem o propósito de simular uma realidade especifica, têm sido amplamente utilizados na prática (COLIN, 2013); processo gerencial de tomada de decisão na Figura 6.

Figura 6 – Processo de Tomada de Decisão



Fonte: Adaptado de Lachtermacher (2013)

Em problemas de otimização linear com apenas duas variáveis de decisão é possível encontrar a solução ótima por intermédio do método gráfico. Entretanto, este método se tornar não viável quando envolve mais variáveis (LACHTERMACHER, 2013).

Em 1946 George Dantzig desenvolve o método simplex, que é um algoritmo que viabiliza a solução de muitos problemas de programação linear, sendo particularmente bem propício para uma abordagem computacional. Quando publicado em 1947 representou um marco para otimização linear, refletindo em diversas áreas de pesquisas e implementações eficientes, tais como agricultura, planejamento e controle da produção, logística, telecomúnicações, finanças entre outras. Em 1984, outro marco importante, a publicação do método de pontos interiores ou primal-dual desenvolvido por Narendra Karmarkar, ao qual também se seguiram intensas pesquisas. Ambos os métodos simplex e primal-dual são, atualmente, as principais ferramentas computacionais disponíveis que viabilizam soluções de problemas de otimização linear (ARENALES *et al.*, 2007; SULLIVAN, 2013).

Segundo Colin (2013) e Lachtermacher (2013) um problema de otimização linear é um problema de programação matemática cuja função-objetivo —Equação 2.2 — e suas restrições são lineares, isto é:

$$Otimizar: \ Z = f(x_1, x_2, x_3, \ldots, x_n) \tag{2.2}$$

$$\text{Sujeito a:} \quad \begin{cases} g_1(x_1, x_2, x_3, \ldots, x_n) \leq b_1 \\ g_2(x_1, x_2, x_3, \ldots, x_n) = b_2 \\ \vdots \\ g_m(x_1, x_2, x_3, \ldots, x_n) \geq b_m \end{cases}$$

Em que:

$$f(x) = (x_1, x_2, \ldots, x_n) = C_1x_1 + C_2x_2 +, \ldots, +C_mx_n$$
$$g_i(x_1, x_2, \ldots, x_n) = a_i1x_1 + a_i2x_2 +, \ldots, +a_mx_n$$
$$i = 1, \ldots, m$$

Onde:

n é o número de variáveis do problema;

m é o número de restrições do problema;

i é o índice de determinada variável;

cj é o coeficiente da variável xj da função objetivo;

a_{ij} é o coeficiente na i-esma restrição da variável xj;

b_i é a constante da i-esma restrição.

Problema de programação linear encontram-se em sua forma padrão, se houver uma maximização da função-objetivo e se todas as restrições forem do tipo menor igual e os termos constantes não assumirem valores negativos. A forma padrão do problema de programação linear garante que as entradas do algoritmo simplex tenham uma estrutura igualmente padronizada (COLIN, 2013; LACHTERMACHER, 2013; SULLIVAN, 2013). Demonstrado na Equação 2.3:

$$Maxmizar: f(x) = (x_1, x_2, \ldots, x_n) = Z = C_1x_1 + C_2x_2 +, \ldots, +C_mx_n) \tag{2.3}$$

$$
\begin{gathered}
a_1 1 x_1 + a_1 2 x_2 +, \ldots, + a_1 n x_n = b_1 \\
a_2 1 x_1 + a_2 2 x_2 +, \ldots, + a_2 n x_n = b_2 \\
\vdots \\
a_m 1 x_1 + a_m 2 x_2 +, \ldots, + a_m n x_n = b_m \\
x_1 \geq 0 \\
x_2 \geq 0 \\
\vdots \\
x_1 \geq 0
\end{gathered}
$$

Entretanto, nem todos os problemas de programação linear se encontram na forma padrão, ou seja, são problemas de maximização com todas as restrições do tipo menor igual, quando isto acontece, pode ser ou não um problema de minimização, o que importa de fato é que, quando o formato não for o padrão, deve se usar diversos métodos antes de aplicar o simplex, a exemplo, a introdução de uma variável de folga, função-objetivo artificial e introdução de uma variável de excesso, dependendo do caso em questão, já que, o algoritmo simplex parte de um problema colocado na forma padrão e avança de uma solução viável para outra de modo que o valor da função-objetivo é diminuído até que o ponto ótimo seja alcançado. O algoritmo possui três instâncias (COLIN, 2013; LACHTERMACHER, 2013):

1. **Início**, prepara os dados de entrada;
2. **Interação**, repete diversas vezes o procedimento, até atingir a otimização do modelo;
3. **Regra de parada**, avalia se a solução ótima foi obtida e se possível obtê-la.

Simplex pode obter uma solução:

- **Inexistente**, quando não há uma solução que satisfaça todas as restrições;
- **Limitada**, se o algoritmo não consegue encontra um ponto de máximo num problema de maximização ou o ponto mínimo num problema de minimização;
- **Múltipla**, quando mais de uma solução satisfaz todas as restrições e alcança o valor extremo da função-objetivo.

A solução manual do algoritmo é feita com auxílio de quadro ou tabela do simplex. No entanto, o elevado número de variáveis e restrições faz com que seja praticamente inconcebível resolver um problema de programação linear sem auxílio de meios computacionais. Há alternativas bem mais atrativas disponíveis como o suplemento solver fornecido com

o Excel e o software Lingo da Lindo Systems Inc, o Tora e a linguagem de modelagem AMPL, a programação linear possui aplicabilidade em áreas diversas (LACHTERMACHER, 2013; TAHA, 2008):

- Administração da produção;
- Análise de investimento;
- Planejamento regional;
- Logística;
- Custo de transporte;
- Rede de distribuição;
- Alocação de recursos.

2.3.2 Programação Inteira

Com a limitação da Programação Linear no tratamento de alguns problemas que restringem o uso de variáveis inteiras, surge naturalmente a Programação Inteira (PI) segundo Colin (2013, p. 173):

> Em 1957, Gomory trabalhava como um consultor da marinha dos EUA. Numa das frequentes viagens que fazia a Washington, ele se deparou com um grupo de pessoas que apresentou um modelo de programação linear de uma força-tarefa da Marinha. Um dos participantes do grupo comentou que seria interessante se eles tivessem uma resposta com números inteiros, tendo em vista que uma resposta do tipo "1,3 havia de carga"não significava nada. Esta observação chamou a atenção de Gomory, que começou a procurar uma solução para aquele tipo de problema

A programação inteira pode ser compreendida como um caso específico da programação linear, onde há exigência que todas as variáveis sejam inteiras, neste caso programação inteira pura, ou exigência parcial onde apenas parte das variáveis deva ser inteira, neste caso denomina-se programação inteira mista e há ainda os casos de programação inteira binária onde todas as variáveis assumem valores 0 ou 1. Assim sendo, o termo o mais correto seria programação linear inteira (COLIN, 2013; TAHA, 2008).

Um problema de programação inteira pode ser compreendido como um problema de programação linear com uma restrição a mais, que as variáveis devam ser inteiras, parcialmente inteiras ou binárias exemplificado na Equação 2.4 (LACHTERMACHER, 2013).

$$Otimizar: \ Z = f(x_1, x_2, x_3, \ldots, x_n) \tag{2.4}$$

$$\text{Sujeito a:} \quad \begin{cases} g_1(x_1, x_2, x_3, \ldots, x_n) \leq b_1 \\ g_2(x_1, x_2, x_3, \ldots, x_n) = b_2 \\ \vdots \\ g_m(x_1, x_2, x_3, \ldots, x_n) \geq b_m \end{cases}$$

$x_1, x_2, x_3, \ldots, x_n$ São inteiros, parcialmente inteiros ou binários Onde:

$$f(x) = (x_1, x_2, \ldots, x_n) = C_1x_1 + C_2x_2 +, \ldots, +C_mx_n$$
$$g_i(x_1, x_2, \ldots, x_n) = a_i1x_1 + a_i2x_2 +, \ldots, +a_mx_n$$
$$i = 1, \ldots, m$$

Ao estudar algoritmos de programação inteira, deve ter-se em mente que, muito embora seja comprovado que estes algoritmos convergem em um número finito de interações, há um inerente erro de arredondamento quando da sua implementação em computadores, sendo esta uma experiência prática diferente da teórica, está falta de consistência na resolução com inteiros representa uma desvantagem dos algoritmos de programação inteira (TAHA, 2008).

Existe uma variedade de algoritmo para resolver problemas de programação inteira, o mais intuitivo seria o arredondamento das soluções não inteiras para o inteiro mais próximo. É possível, porém improvável que se encontre a solução ótima de um problema com o uso deste procedimento. Entretanto, esta escolha pode ocasionar o não atendimento das restrições bem como uma solução não ótima. Isto não significa que arredondamentos não devam ser feitos, a depender do problema, o arredondamento pode representar uma solução simples e adequada, apenas quando este não pode ser aplicado deve se buscar outras abordagens. Dentre as abordagens de não arredondamento pode-se classificar os algoritmos de programação inteira como genéricos e específicos. Os algoritmos genéricos foram criados para resolver qualquer problema que possa ser formulado na forma padrão da programação inteira. Os algoritmos específicos são mais eficientes, pois se beneficiam de particularidades dos problemas para os quais foram concebidos. Estes são desenvolvidos especificamente para solução de um problema ou são derivações melhoradas de algoritmos genéricos. Programação inteira tem sido aplicada em áreas com problemas onde um número não inteiro não faz sentido (COLIN, 2013):

- Rede de distribuição;
- Logística;
- Alocação de recursos;
- Escalonamento da Produção.

2.3.3 Programação Dinâmica

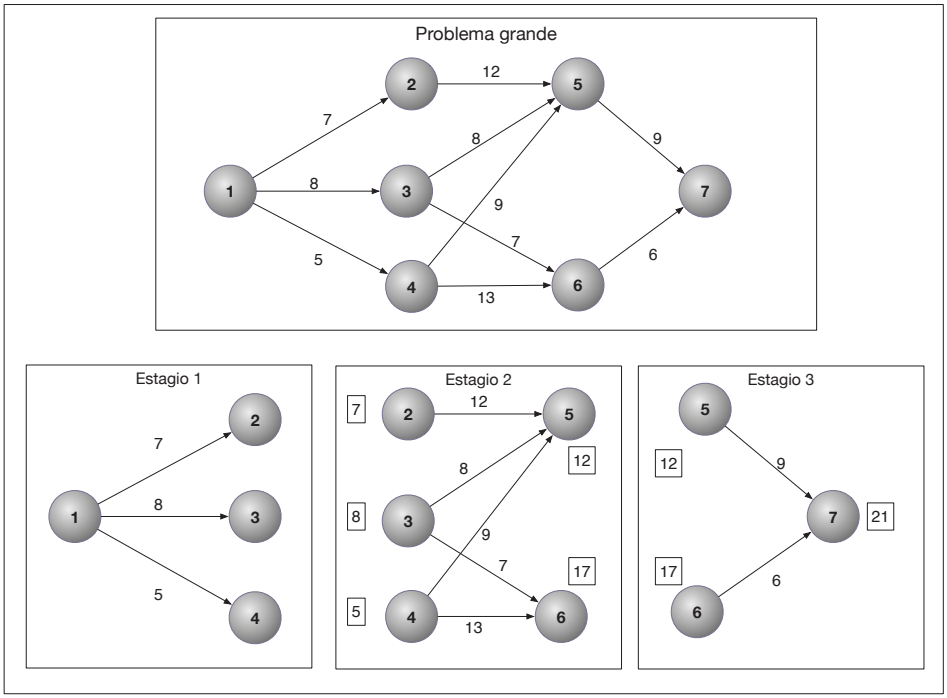
A Programação Dinâmica (PD) determina a solução ótima de um problema de multivariáveis decompondo-o em estágios. Assim, cada estágio compreende subproblema uma única variável. Diferentemente da programação inteira, esta emerge de forma suave entre 1940 e 1957 com a publicação do livro Programação Dinâmica, de Richard Ernest Bellman. A decomposição assegura a vantagem de ampla aplicabilidade, o processo de otimização em cada estágio envolve apenas uma variável. Isto garante uma abordagem mais simplificada em termos de cálculo de que lidar com todas as variáveis de forma simultânea. Nas palavras de Bellman: uma solução de problema de programação dinâmica, não é simplesmente um conjunto de funções ou números, mas sim uma regra que diz ao tomador de decisões o que fazer; uma política ou no jargão mais moderno, uma estratégia, deste modo a principal dificuldade desta técnica é a necessidade de modelagem individuais para cada problema. Diferentemente da programação linear, não há um algoritmo aplicável genericamente, possuindo em sua modelagem mais semelhanças com um problema de álgebra ou probabilidade do que de PL (COLIN, 2013; TAHA, 2008).

Segundo Colin (2013), a Programação Dinâmica pode ser utilizada para resolver problemas com variáveis inteiras ou contínuas, problemas lineares ou não lineares, problemas com horizonte de tomada de decisão finito ou infinito, problemas determinístico ou estocástico ou ainda uma combinação destes tipos. Isto a faz uma técnica extremamente versátil. Entretanto a esta versatilidade tem um custo ou "instabilidade"como denominou Bellman, a particularização da solução do problema.

A programação dinâmica, pode ser entendida como uma técnica de decomposição, problemas grandes possuem complexidade elevada. Em contrapartida problemas menores possuem baixa complexidade. Tendo isto como princípio, seria interessante decompor um problema grande em vários problemas menores, resolvê-los e então recompô-los novamente de modo que o problema grande também fosse resolvido. Entretanto, como garantir que ao recompor as soluções das partes pequenas encontrar-se-á a solução do problema grande? A PD serve exatamente para isto; decompor um problema grande em partes menores, resolver as partes, e recompô-las novamente, de modo a garantir que a solução do problema grande seja correta. Assim, os cálculos em PD são elaborados de modo que a solução ótima de um problema é usado como dado de entrada para o subproblema subsequente. A Figura 7 ilustra de decomposição de um problema de menor caminho (COLIN, 2013; TAHA, 2008).

Portanto, a programação dinâmica é uma técnica de decomposição de problemas de otimização em uma família de sub problemas que podem ser resolvidos mais facilmente. A aplicação da PD tem obtido êxito na solução de problemas originários das mais diversas áreas (ARENALES *et al.*, 2007).

Figura 7 – Rede de Rotas, Decomposição do Problema de Menor Caminho



Fonte: Adaptado de Taha (2008)

- Problemas de panejamento da produção;
- Gestão de estoques;
- Determinação de tamanhos de lotes de produção;
- Problemas de corte e empacotamento;
- Alocação de recursos limitados;
- Determinação da capacidade de expansão de usinas geradoras de energia elétrica;
- Gestão de projetos;
- Gestão financeira;
- Determinação da confiabilidade de sistemas de comunicação;
- Programação de tarefas em sistemas de manufatura flexível; manutenção e reposição de equipamentos.

2.4 Métodos de Otimização

Os problemas de otimização são problemas de maximização ou minimização de uma função de uma ou mais variáveis num determinado domínio, sendo que, geralmente,

existe um conjunto de restrições nas variáveis. Os algoritmos usados para a solução de um problema de otimização podem ser, basicamente, determinísticos ou probabilísticos.

Para melhor entendimento dos algoritmos de otimização, faz-se necessário o conhecimento de alguns conceitos e definições utilizados na literatura (COLIN, 2013). A seguir são listados alguns termos usualmente relacionados a um problema de otimização qualquer:

- **Parâmetros:** São aqueles que se alteram durante o processo de otimização, podendo ser contínuos (reais), inteiros ou discretas;
- **Restrições:** São funções de igualdade ou desigualdade sobre as variáveis de decisão que descrevem situações de projeto consideradas não-desejáveis;
- **Espaço de busca:** é o conjunto, espaço ou região que compreende as soluções possíveis ou viáveis sobre as variáveis de decisão do problema a ser otimizado, sendo delimitado pelas funções de restrição;
- **Função-objetivo:** é a função de uma ou mais variáveis de decisão que se quer otimizar, minimizando-a ou maximizando-a;
- **Ponto ótimo:** é o ponto formado por valores das variáveis de decisão que maximizam (minimizam) a função-objetivo e satisfazem as restrições;
- **Valor ótimo:** é o valor da função objetivo no ponto ótimo;
- **Algoritmo:** Sequência de instruções que gera um determinado resultado mediante a uma determinada entrada.

A função-objetivo e as funções de restrição podem ser lineares ou não-lineares em relação às variáveis de decisão e, por esta razão, os métodos de otimização podem ser, também, lineares ou não-lineares, respectivamente.

2.4.1 Métodos Determinístico

Os métodos de otimização baseados nos algoritmos determinísticos — maioria dos métodos clássicos — geram uma seqüência determinística de possíveis soluções. Na maioria das vezes, requerem o uso de pelo menos a primeira derivada da função-objetivo em relação às variáveis de decisão, dentre eles (FEOFILOFF, 1999).

- Simplex (método mais tradicionalmente usado);
- Gaus–Jordan;

- Gaus–Jordan–Chio;
- Simplex–Chio;
- Yamnitsky–Levin;

Os métodos determinísticos apresentam teoremas que lhes garantem a convergência para uma solução ótima, que não é necessariamente a solução ótima global. Como nesses métodos a solução encontrada é extremamente dependente do ponto de partida fornecido, pode-se convergir para um ótimo local, o que não é desejável. Assim, não possuem bom desempenho quando aplicados à otimização de funções multimodais, isto é, funções que possuem vários ótimos locais (FEOFILOFF, 1999).

Este trabalho não aplica os tradicionais métodos determinístico para otimização. Portanto, nas seções subsequentes os demais métodos de otimização não são abordados.

2.4.2 Métodos Heurísticos

O termo heurístico origina-se da palavra grega *heurísko* que significa encontrar, descobrir, achar. Este procedimento visa simplificar tarefas difíceis por soluções viáveis otimizadas ou aproximadamente otimizada. Existem inúmeros problemas para os quais deseja-se desenvolver um algoritmo eficiente. Muitos destes são problemas de otimização (numérica, combinatorial), outros são de síntese de um objeto (programa de computador, circuito eletrônico etc.) e, em outros, busca-se um modelo que reproduza o comportamento de determinado fenômeno (*machine learning*[5]). Para vários desses problemas é freqüentemente possível encontrar um algoritmo que ofereça uma solução ótima ou aproximadamente ótima. Alguns desses algoritmos requerem, no entanto, o conhecimento do modelo matemático que representa o problema, informação esta muitas vezes não disponível ou difícil de ser obtida.

2.4.3 Computação Evolucionaria

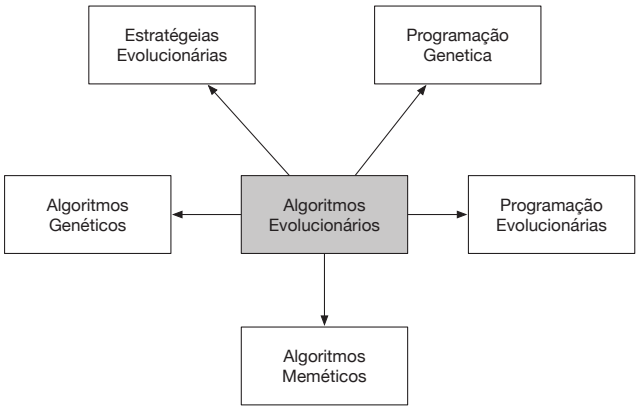
A Computação Evolucionária teve origem no final da década de 50 do século XX (BÄCK; FOGEL; MICHALEWICZ, 1997) e permaneceu relativamente desconhecida da comunidade científica por aproximadamente três décadas, devido principalmente à falta de computadores eficientes na época, mas também devido à metodologia pouco desenvolvida durante as primeiras pesquisas. Durante a década de setenta, os trabalhos de Holland, Rechenberg, Shwefel e Foger foram fundamentais para modificar a imagem da Computação Evolucionária que, a partir de então, começou a ser largamente desenvolvida. Os Algoritmos Evolucionários (AE) formam uma classe de métodos de otimização probabilísticos que são inspirados por alguns princípios baseados em mecanismos evolutivos

[5] **machine learning:**Aprendizagem de máquina

encontrados na natureza, como auto-organização e o comportamento adaptativo (BEYER, 2001; BEYER *et al.*, 2002)

Os AE dispensam informações auxiliares e são usados em métodos aplicados na solução de problemas com grandes espaços de busca, de difícil modelagem, ou para os quais não há um algoritmo eficiente disponível. A Figura 8 mostra os algoritmos classificados como evolucionários: Algoritmos Genéticos (AG), Estratégias de Evolução, Programação Evolucionária, Programação Genética.

Figura 8 – Algoritmos Evolucionários



Fonte: Elaborado pelo autor 2017

Um algoritmo evolucionário se distingue dos métodos determinísticos mais comuns basicamente por:

- empregar uma população de indivíduos, ou soluções;
- trabalhar sobre uma codificação das possíveis soluções (genótipos) e não sobre as soluções (fenótipos) propriamente ditas;
- empregar regras de transição probabilísticas;
- não requerer informações adicionais (derivadas, por exemplo) sobre a função a ser otimizada e sobre as restrições.

Assim, a busca de soluções pode se dar em conjuntos não-convexos, com funções-objetivo não-convexas e não-diferenciáveis, podendo-se trabalhar simultaneamente com variáveis reais, lógicas e inteiras.

2.4.4 Algoritmos Genéticos

Os algoritmos genéticos (AG) foram desenvolvidos por John Holland no final da década de 60 do século XX, inspirado pela teoria da evolução de Darwin descrita em seu

famoso trabalho *The origin of Species* (Darwin, 1859). À época, o interesse de Holland práticastava limitado a problemas de otimização. O foco dos seus estudos era voltado para sistemas adaptativos complexos. No entanto, os algoritmos podem ser aplicados em problemas de otimização pela versatilidade e robustez que apresentam, embora não devam ser considerados estritamente minimizadores de funções. Nas últimas décadas, os AG vêm resolvendo problemas em vários domínios e se tornaram entre os algoritmos evolucionários os mais utilizados (GOLDBERG, 2006).

Em um AG, as variáveis do problema são representadas como genes em um cromossomo, também denominado indivíduo. Utiliza os princípios da sobrevivência dos mais aptos e a troca de informação genética ocorre de forma estruturada, porém aleatória. O AG apresenta um grupo de soluções candidatas, denominada população, na região de soluções. Através de mecanismos como a seleção natural e o uso de operadores genéticos, tais como a mutação e o cruzamento, os cromossomos com melhor aptidão são encontrados. A seleção natural garante que os cromossomos mais aptos gerem descendentes nas populações futuras. Usando um operador de cruzamento, o AG combina genes de dois ou mais cromossomos de pais previamente selecionados para formar novos cromossomos, os quais têm grande possibilidade de serem mais aptos que os seus genitores (LIDEN, 2012).

No algoritmo 1 apresenta-se um AG clássico. Neste caso, uma população é inicializada aleatoriamente e avaliada. Inicia-se o processo de evolução. Um conjunto de pais é selecionado da população atual, e em seguida os descendentes são produzidos após operações de cruzamento e mutação, respectivamente. Os descendentes são avaliados e tem-se, então, uma nova população, . Isto ocorre até que uma condição de parada seja atendida. Retorna-se à melhor solução produzida durante o processo de evolução (LIDEN, 2012) .

Algoritmo 1: Pseudocódigo do algoritmo genético

Gerar população inicial $P(0)$;
Avalie $P(0)$;
início $\leftarrow 0$;
comprimento $\leftarrow$ *tamanho*;
para $i \leftarrow 0$ **até** $i =$ *condição de parada* **faça**
  incio $\leftarrow 0$;
  Passos:
  - $P'(t) \leftarrow$ Pais selecionados em $\{P(t),$ é a população no tempo $t\}$
  - P"$(t) \leftarrow$ Descendentes após cruzamento e mutação

  Avalie P"(t)
  P"$(t+1) \leftarrow P$"$(t) + P(t)$
  $t = t + 1$
fim
Saída: a melhor solução

O Algoritmo Genético ocupa lugar de destaque entre os paradigmas da Computação Evolucionária devido a uma série de razões, entre as quais:

1. apresenta-se como o paradigma mais completo da Computação Evolucionária, visto que engloba de forma simples e natural todos os conceitos nela contidos;
2. apresenta resultados bastante aceitáveis, com relação à precisão e recursos empregados (sendo de fácil implementação em computadores domésticos de médio porte), para uma ampla gama de problemas de difícil resolução por outros métodos;
3. é muito flexível, aceita sem grandes dificuldades uma infinidade de alterações na sua implementação e permite fácil hibridização (vantagem importante no caso de aprendizagem), inclusive com técnicas não relacionadas à Computação Evolucionária;
4. em relação aos outros paradigmas da Computação Evolucionária, é o que exige menor conhecimento específico do problema em questão para o seu funcionamento, o que o torna altamente versátil. Além disso, agrega conhecimento específico com pouco esforço;
5. é o paradigma mais usado dentro da Computação Evolucionária e, junto com as Redes Neurais, são os mais usados de toda a Computação Natural.

A principal vantagem do Algoritmo Genético é trabalhar com o conceito de população, ao contrário de muitos outros métodos que trabalham com um só ponto de partida. Tendo uma população de pontos bem adaptados, é reduzida a possibilidade de alcançar um falso ótimo. O AG consegue grande parte de sua amplitude simplesmente ignorando informação que não constitua parte do objetivo, enquanto outros métodos se sustentam fortemente nesse tipo de informação é em problemas para os quais a informação necessária não está disponíveis ou se apresenta de difícil acesso, que estes outros métodos falham (LIDEN, 2012; BEYER *et al.*, 2002). Cabe ainda destacar que o AG é um método de busca:

- **Cego:** não tem conhecimento específico do problema a ser resolvido, tendo como guia apenas a função-objetivo;
- **Codificado:** não trabalha diretamente com o domínio do problema e sim com as representações dos seus elementos;
- **Múltiplo:** executa busca simultânea em um conjunto de candidatos;
- **Estocástico:** combina regras probabilísticas e determinísticas com alguma proporção variável. Esse conceito se refere tanto às fases de seleção como às fases de transformação.

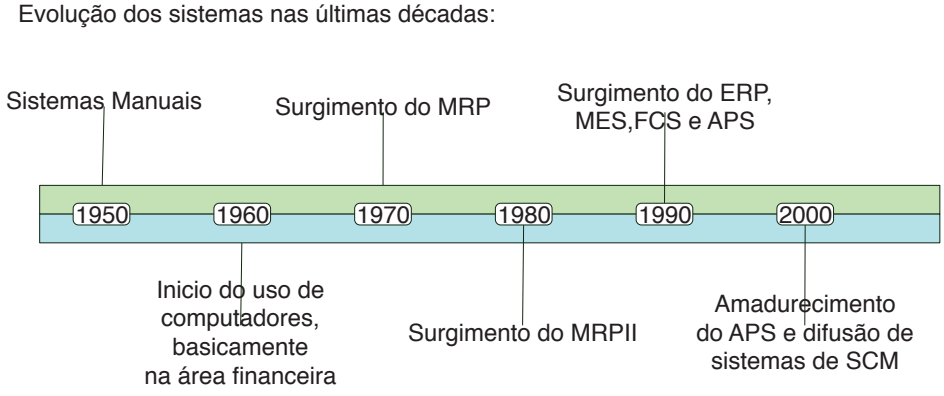
2.5 Plataformas de Auxílio ao Escalonamento

A operação de um sistema de manufatura automatizado é geralmente administrada com base em planos previamente gerados — cronograma de lista de eventos — fornecido por uma plataforma de planeamento, geralmente um software avançado de planejamento e agendamento (APS – *Advanced Planning and Scheduling*). Com os recurso computacionais atualmente disponível, é cada vez mais comum um planejamento e controle fundamentado em modelos de otimização por intermédio de soluções de programação baseadas em sistemas APS especializados no conceito de Capacidade Finita (Finite Capacity), respeitando assim as limitações de capacidade dos recursos — máquinas, mão de obra e ferramentas — disponibilidade dos materiais, as estratégias e políticas de atendimento dos pedidos e qualquer restrição operacional que gere impacto no negócio. Sistemas APS tem como característica precisa (JAIN; FOLEY, 2016):

- Alta performance no processamento e a precisão nas programações geradas;
- A elevada capacidade de refletir a realidade operacional dos diferentes sistemas de produção;
- A alta tecnologia com que são desenvolvidos;

Os Sistemas APS procuram considerar todas as restrições existentes no processo com o objetivo de otimizar o desempenho, utilizando regras de sequenciamento, heurísticas e métodos de otimização, isto garante uma extrema flexibilidade ao emprega-los, permitindo que estes sejam empregados praticamente em todos os ambientes de sistemas produtivos (VIDONI; VECCHIETTI, 2015b). O conceito APS como hoje é conhecido é relativamente novo, demonstrado na Figura 9.

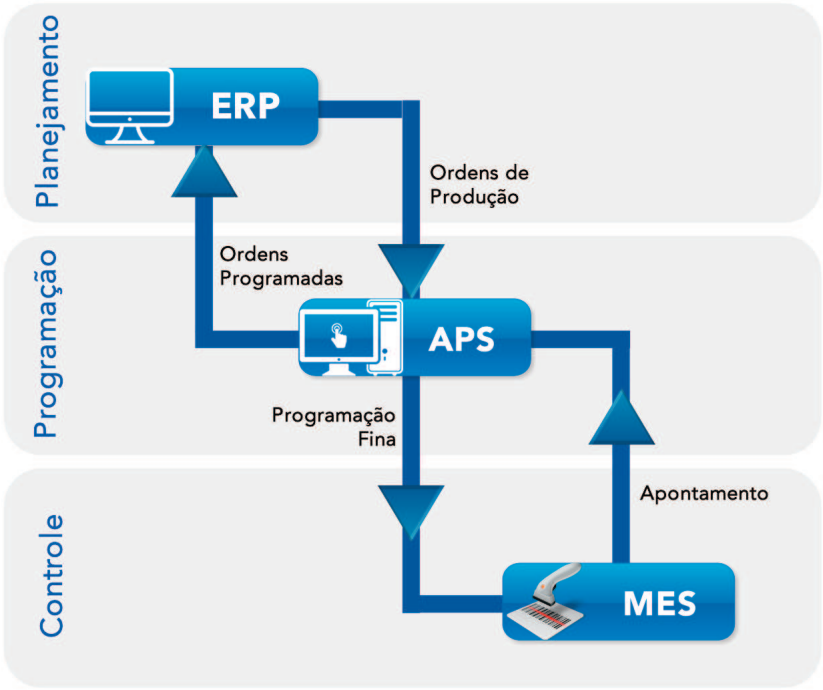
Figura 9 – Evolução dos Sistemas de Controle da Produção



Fonte: Adaptado de Faé e Erhart (2009)

Porém estes sistemas não foram desenvolvidos com o intuito de substituir nem de excluir os sistemas MRP (Manufacturing Resource Planninge), mas sim para suprir as suas carências e complementá-los, ou seja, apesar de estarem aptos a trabalhar de forma isolada (stand alone), o seu propósito de funcionar integrados com outros softwares, como forma de evitar a duplicidade de dados cadastrais e a sobreposição conflitante com os demais sistemas de gestão. Efetivamente, o MRP gera as ordens de produção e o APS realiza a programação exata da produção, disponibilizando as informações de execução para a fábrica em formato eletrônico através de um sistema MES [6](Manufacturing Execution System) demonstrado na Figura 10 (FAÉ; ERHART, 2009; VIDONI; VECCHIETTI, 2015a).

Figura 10 – Sistema de Manufatura com Integração Entre ERP, APS e MES



Fonte: Adaptado de Faé e Erhart (2009)

Segundo Faé e Erhart (2009), softwares APS podem ser divididos em três principais grupos de fornecedores:

(a) Fornecedores de ERP que possuem um módulo de APS como parte integrante da sua plataforma;

[6] **MES:** Manufacturing Execution System é uma ferramenta de controle da produção aplicada no chão-de-fábrica que permite visualizar e monitorar processos em tempo real, fornecendo informações que levam a uma melhor eficiência operacional

(b) Fornecedores de uma plataforma completa de Supply Chain Management que possuem um módulo de APS como parte integrante do seu pacote de soluções;

(c) Fornecedores independentes e especialistas em softwares APS.

O principal objetivo dos sistemas APS é a programação exata da produção, respeitando as restrições referentes a disponibilidade de materiais e máquinas complementando os sistemas ERP, tipicamente transacionais, maior suporte aos processos de tomada de decisão. Utilizando métodos heurísticos e programação linear, são ferramentas de suporte a decisão, podem simular diversos planos e programações com diversas restrições, permitindo a geração de planos otimizados; garantem a otimização de problemas complexos de planejamento, possuem grande velocidade de processamento, requerendo um hardware a altura. Porém, esta plataforma integrada de planejamento para sistema de manufatura nem sempre se encontra acessível, existindo uma lacuna que regularmente é preenchida usando o Microsoft Visual Basic for Application (VBA) aliado a técnicas de planilhamento propondo "pseudos APS"como alternativa integrativa dos sistemas avançado de planejamento e agendamento (CHEN *et al.*, 2016; VIDONI; VECCHIETTI, 2015b; VIDONI; VECCHIETTI, 2015a).

Apesar da abrangência e versatilidade das plataformas de planejamento integrado, é bastante usual em empresas de todos os portes a tomada de decisão auxiliada por planilhas eletrônicas. Em diferentes processos de gestão empresarial nas organizações como suporte a realidade da programação da produção, para suplementar manualmente aquilo que o sistema corporativo não atende (GIROTTI; MESQUITA, 2016).

2.6 Trabalhos Correlatos

Segundo Hoorn (2018) quase a metade das soluções conhecidas em problemas de *job shop schedule* são encontradas por algoritmos de busca tabu, seguidos de busca local, heuristica *Shifting Bottleneck* e técnicas vinculadas. Desde a década de 80 com Davis (1985) confronta-se resultados obtidos por intermédio de algoritmos genéticos a outras metodologia de resolução.

Existem vários outros trabalhos que utilizam técnicas de computação evolutiva na resolução do problema *job shop schedule*. Porém, os trabalhos citados na Tabela 1 , apresentaram um impacto mais significativo na compreensão e elaboração dos operadores genéticos implementados neste trabalho.

Tabela 1 – Trabalhos Correlatos.

Autor	Título	Periódico	Contribuição
Yamada, T.; Nakano (1992)	Parallel Instance Solving From Nature	A Genetic Algorithm Applicable To Large-Scale Job-Shop Instances	Codificação em matriz binária
Yamada e Nakano (1997)	Proceedings of Modern Heuristic for Decision Support	Genetic Algorithms for Job-Shop Scheduling Problems	propõe uma modelagem com algoritmo genético e *makespan* semi-ativo.
Zhang, Rao e Li (2008)	International Journal of Advanced Manufacturing Technology	An Effective Hybrid Genetic Algorithm For The Job Shop Scheduling Problem	apresenta um novo operador de crossover, denominado crossover de operações de precedência.
Zhang, Yang e Gao (2008)	International Symposium on Computational Intelligence and Design	A Genetic Algorithm And Tabu Search For Solving Flexible Job Shop Schedules	Algoritmo genético combinado a critério de busca local.
Baptiste, Flamini e Sourd (2008)	Computers and Operations Research	Lagrangian bounds for just-in-time job-shop scheduling	elaboração de um modelo matemático para problemas de *job shop schedule*
Liu (2009)	International Journal of Computer Integrated Manufacturing	Lot Streaming For Customer Order Scheduling Problem In Job Shop Environments	Divisão dos *jobs* em *sub-jobs* com sub-roteiros de fabricação.
Defersha e Chen (2012)	International Journal of Production Research	Jobshop Lot Streaming With Routing Flexibility, Sequence-Dependent Setups, Machine Release Dates And Lag Time	Algoritmo genético e *Lot streaming*.
Nguyen e Bao (2016)	Procedia - Procedia Computer Science	An Efficient Solution To The Mixed Shop Scheduling Problem Using A Modified Genetic Algorithm	Utiliza algoritmo genético na resolução de *job shop schedule* misto.
Kundakc e Kulak (2016)	Computers & Industrial Engineering	Hybrid Genetic Algorithms For Minimizing Makespan In Dynamic Job Shop Scheduling Problem	apresenta híbrido de algoritmo genético na resolução de problemas *job shop schedule*.
Hoorn (2018)	Journal of Scheduling	The Current State Of Bounds On Benchmark Instances Of The Job-Shop Scheduling Problem	Compilação e atualização dos limites

Fonte: Elaborado pelo autor (2018)

3 METODOLOGIA DE PESQUISA

Este capítulo apresenta a análise metodológica referente à caracterização do roteiro que guiou o desenvolvimento da pesquisa.

3.1 Classificação da Pesquisa

Segundo Oliveira (1997), a metodologia é o estudo do conjunto de processos que torna possível a interpretação adequada de uma realidade, produzindo determinado objeto ou desenvolvendo certos procedimentos ou comportamentos. Assim sendo, o método identifica a forma pela qual pode se alcançar determinado objetivo. Para Jung (2010), uma pesquisa científica é classificada segunda sua tipologia referente à natureza, forma de abordagem do problema, objetivos, procedimentos e local de realização.

No que se refere a natureza da forma de abordagem do problema, esta pesquisa pode ser caracterizada como aplicada, pois há geração de novos conhecimentos resultantes do processo de pesquisa, proporcionando aplicação prática de modelos de otimização no escalonamento da produção (JUNG, 2010).

Do ponto de vista da forma de abordagem do problema, esta pesquisa caracteriza-se como pesquisa quantitativa, pois utiliza-se de parâmetros de modelagem matemática e computacional para qualificação de sua aplicabilidade. Entretanto, a flexibilidade da ferramenta desenvolvida permitirá análise crítica de dados qualitativos. Logo, é possível fazer ajustes nos dados dos resultados obtidos quantitativamente por intermédio da análise crítica qualitativa (MARCONI; LAKATOS, 2003) .

Quanto ao objetivo ou propósito, este trabalho é de cunho preditivo, isto é, parte da hipótese de que um modelo quantitativo pode ser desenvolvido para explicar ou predizer o comportamento de um sistema produtivo em função dos parâmetros de entrada do sistema. Utiliza-se de um modelo abstrato, descritos em linguagem matemática e computacional para calcular valores numéricos das propriedades de um sistema de planejamento, podendo ser usado para analisar os resultados de diferentes ações possíveis, resultando em um levantamento de uma série de pontos relevantes para o planejamento e condução de um estudo (GANGA, 2012).

Referente aos procedimentos, segundo Jung (2010), este estudo pode ser classificado como um estudo experimental, pois há a elaboração de um modelo matemático implementado a uma rotina computacional.

3.2 Abordagens de Algoritmos

A literatura classifica o problema *Job Shop Scheduling* como NP-difícil. Isto implica que esse é um problema para o qual não existe algoritmos que o resolva em tempo polinomial. Trata-se de um problema de otimização combinatória. Empregam-se especificações explicítas ou implícitas de possíveis alternativas em busca de soluções satisfatórias ou ótimas, conforme o algoritmo implementado para otimização (FURINI; LJUBIĆ; SINNL, 2017; NAGATA; ONO, 2017). Assim, três abordagens devem ser levadas em consideração para elaboração de algoritmos de otimização:

- **A Abordagem Matemática:** enfatiza obter resultados ótimos empregando parâmetro de desempenho. O parâmetro pode ser a minimização dos tempos de produção ou a maximização do uso dos recursos. Dependendo da complexidade do problema obtenção da solução ótima pode ser inviabilizada pelo tempo gasto;
- **A Abordagem Ponto-a-Ponto/Gulosos:** assume a estratégia de resolver o problema fazendo a escolhas localmente ótimas, em cada fase da resolução na tentativa que isto resulte num ótimo global. Porém, existe grande probabilidade dessa estratégia retornar um ótimo local como solução;
- **A abordagem Heurística:** fundamenta-se em regras indutivas de escalonamento que buscam obter soluções satisfatórias, aproximadamente ótimas. No entanto, não se tem a garantia da solução ótima.

Neste tipo de problema, costuma-se sacrificar a obtenção da solução ótima aplicando métodos heurísticos. Assim, objetiva-se uma solução sub-otima. Apesar de não garantir que a solução encontrada seja a ótima, o resultado apresentado por uma abordagem heurística costuma ser aproximado ao ótimo, a um tempo computacional admissível (ÇALI; BULKAN, 2015; KUNDAKC; KULAK, 2016; KURDI, 2016).

3.3 Generalização Formal do Modelo Matemático

A generalização matemática formal aplicada neste trabalho foi proposta por Baptiste, Flamini e Sourd (2008). Consiste em especificar que um conjunto de n *jobs*, $J = \{J_1, J_2, \ldots, J_n\}$, deve ser processado por um conjunto de m máquinas, $M = \{M_1, M_2, \ldots, M_m\}$. Cada *job*, J_i, possui uma sequência ordenada de operações, $O_i = \{O_i^1, O_i^2, \ldots, O_i^m\}$, sendo O_i^k a k-ésima operação do *job*, J_i. Com cada operação O_i^k tendo uma data de entrega d_i^k e uma instância de processamento P_i^k vinculada a uma máquina específica, $M(O_i^k)$. Assim, o índice da operação é representado por k e o índice do *job* é representado por i.

O objetivo é calcular um sequenciamento com finalizações, C_i^k, em tempos viáveis. Portanto, para se avaliar uma solução, cada operação, O_i^k, tem dois coeficientes de pena-

lidade, α_i^k para o adiantamento e β_i^k para o atraso. Isto posto o adiantamento é dado por $E_i^k = max(0, d_i^k - C_i^k)$ e o atraso por $T_i^k = max(0, C_i^k - d_i^k)$. Desse modo, uma operação está adiantada se $E_i^k > 0$ ou atrasada se $T_i^k > 0$. Em que a função objetivo minimiza a somas das penalidades de atraso e adiantamento, como apresentado na Equação 3.1.

$$Min \sum_{i=1}^{n} \sum_{k=1}^{m} (\alpha_i^k E_i^k + \beta_i^k T_i^k) \tag{3.1}$$

Sujeito às seguintes restrições:

$$\begin{cases} C_1^k - P_i^1 \geq 0 & \boxed{\text{I}} \\ C_i^k \leq C_i^{k+1} - P_i^{k+1} & \boxed{\text{II}} \\ C_i^k \geq C_j^h + P_i^k \text{ ou } C_j^h \geq C_i^k + P_j^h & \boxed{\text{III}} \end{cases}$$

I A primeira operação começa após o tempo 0, isto é para qualquer i em $\{1, \ldots, n\}$;

II Para que uma solução seja factível os critérios de precedências devem ser obedecidos, cada par de operações O_i^{k-1} não pode começar antes O_i^k para $i = 1, \cdots, n$ e $k = 1, \cdots, m$,;

III Restrições de recursos (também conhecidas como restrições disjuntivas): duas operações de *jobs* distintas O_i^k e O_j^h que precisam ser processadas na mesma máquina não podem ser processadas simultaneamente, ou seja para $M(O_i^k) = M(O_j^h)$ e $i, j = 1, \cdots, n$ obrigatoriamente $i \neq j$.

Cada *job* possui um roteiro de fabricação para com as máquinas, ou seja, uma lista de etapas a serem executadas de forma sequencialmente distinta; uma vez que produtos de famílias distintas compõem uma lista ordenada de operações próprias, sendo esta definida pela sucessão de máquinas requeridas e pelos tempos de processamentos em cada operação. Desse modo, é válido salientar que interrupções das operações não são permitidas e inexistem restrições quanto às precedências entre operações de *jobs* distintos ou operações executadas por uma mesma máquina.

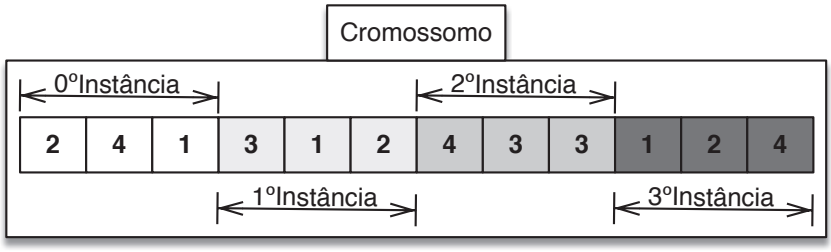
Sendo o roteiro de fabricação fixo para cada *job*, o problema a ser resolvido consiste em determinar a sequência de operações para cada máquina segundo o roteiro de fabricação de cada *job*, de modo que o tempo total de produção entre o iniciar do primeiro *job* até o término do último seja o mínimo possível.

O modelo matemático utilizado para a generalização formal do problema de escalonamento de *jobs*, ordens de produção ou lotes, segue o objetivo de otimização conhecido na literatura por *makespan*, onde o objetivo é encontrar a solução com menor tempo de finalização. Embora não seja a única abordagem desse problema, é a mais largamente abordada na literatura (GÜÇEMIR; SELIM, 2017; SOBEYKO; MÖNCH, 2016).

3.4 Codificação do AG

O modelo matemático foi a base para codificar uma rotina computacional responsável por emular o comportamento de um AG. Para esta proposta de algoritmo genético, cada solução factível é representada em um cromossomo, codificada num vetor de $i \times k$. Dessa forma, cada gene terá uma representação matemática dada por $M(O_i^k)$, onde k representa o índice da operação e i índice do *job*, cada i aparece k vezes no vetor. A Figura 11 exemplifica um problema envolvendo 4 *jobs* e 3 máquinas.

Figura 11 – Solução Exequível, Codificado em um Cromossomo



Fonte: Elaborado pelo autor (2017)

O primeiro gene assume o valor 2. Assim, a 1° operação do *job* 2 é alocada na máquina $M(O_2^1)$ começando na instância P_2^0. O segundo gene tem o valor 4, representado matematicamente por $M(O_4^2)$ com instância P_4^0. Portanto, o valor de k é determinado pela sequência de vezes que o *job* se repete dentro do vetor cromossomo.

O número de *jobs* do problema determina a quantidade de instâncias a qual o cromossomo é subdivido e o número de máquinas determina a extensão de cada instância. Assim, tem-se o escalonamento das operações nas respectivas máquinas segundo o roteiro requerido por cada *job*. Obedecendo as restrições de precedência, cada operação é iniciada de imediato, sempre que possível. Desta forma, os *jobs* 2 e 1 serão concluídos nas instâncias P_2^3 e P_3^2 respectivamente. Considerando hipoteticamente os roteiros de fabricação e tempos de processamento na Tabela 2:

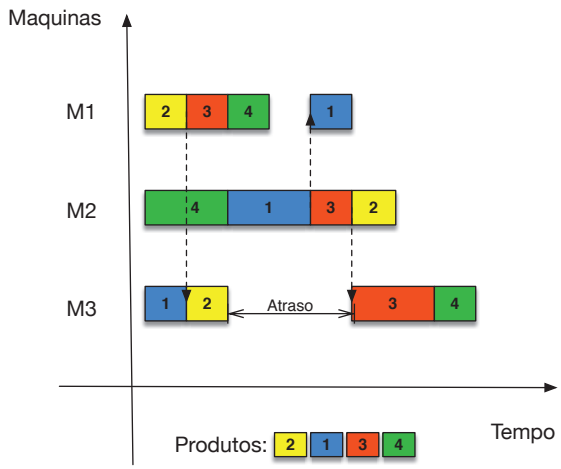
Tabela 2 – Roteirização Hipotetica.

Jobs	Roteiro de Fabricação	Tempo de Processamento
Produto1	(3,2,1)	(1,2,1)
Produto2	(1,3,2)	(1,1,1)
Produto3	(1,2,3)	(1,1,2)
Produto4	(2,1,3)	(2,1,1)

Fonte: Elaborada pelo autor (2017).

Ao aplicar a solução factível apresentada na Figura 11 aos dados hipotéticos da Tabela 2, obtém-se um possível planejamento da escala de produção, representado pelo gráfico de Gannt da Figura 12.

Figura 12 – Solução Factível Representada em um Gráfico de Gantt



Fonte: Elaborado pelo autor (2017)

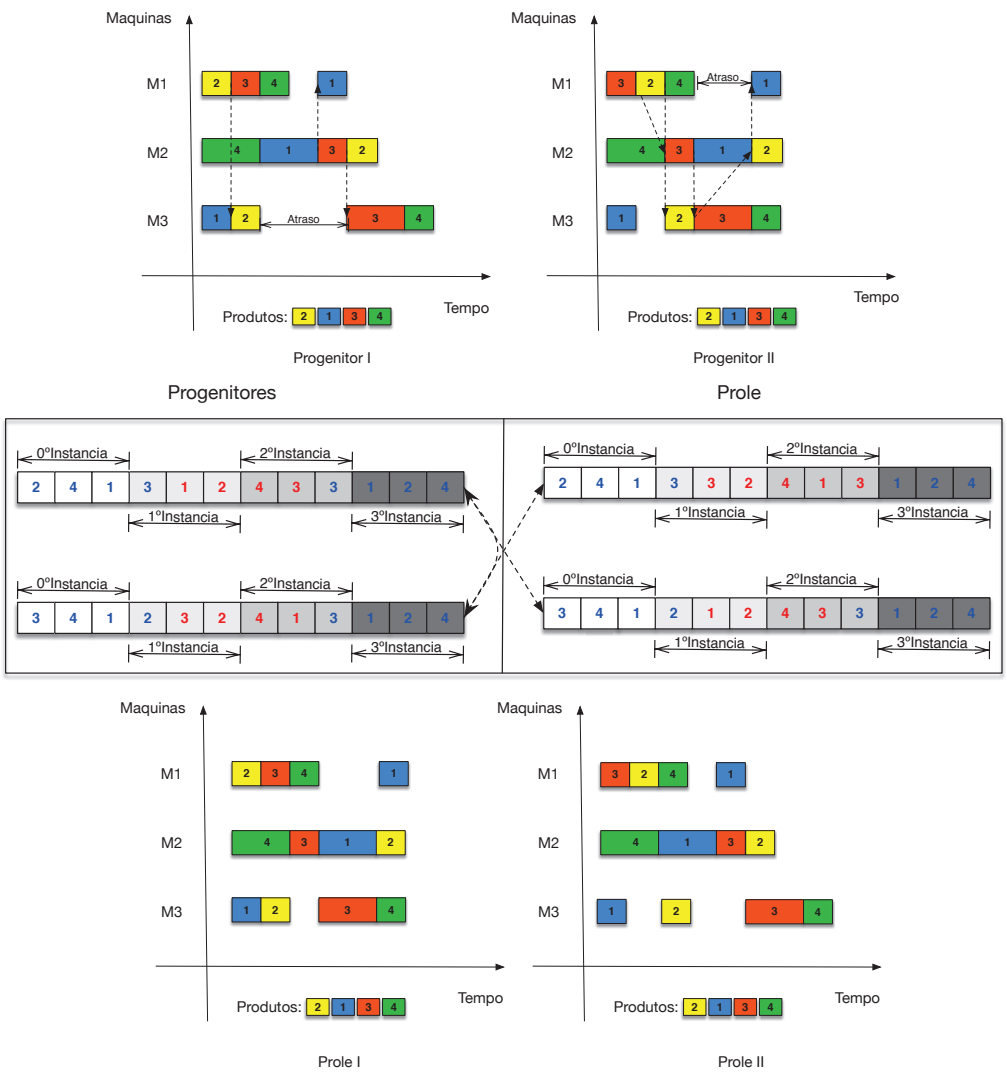
Para 4 *jobs* e 3 máquinas há um total de $3!^4 = 81$ possíveis soluções. Porém, caso seja necessário escalonar 10 *jobs*, o número de soluções a serem exploradas se eleva para $3!^{10} = 59.049$. As possibilidades de planejamento são elevadas, até mesmo para poucas máquinas, pois havendo produção em escala, existe datas de entregas geralmente inflexíveis. Dessa forma, é necessário selecionar, dentre as soluções exequíveis aquelas com menor tempo de finalização, *makespan*, e assim garantir soluções ótimas ou sub-ótimas.

3.4.1 Operadores de Seleção e Crossover

Para seleção dos cromossomos contidos na população aplicou-se o método da roleta viciada, onde os escolhidos para formar a próxima geração são selecionados através de um sorteio de roleta. Neste método, cada cromossomo tem representado na roleta uma fatia proporcional a sua aptidão. Dessa forma, aqueles cromossomos com maior porção na roleta possui maior probabilidade de serem selecionados para formar a próxima geração. Os cromossomos selecionados da população são combinados por um operador de crossover, que utiliza características específicas do problema tratado, da seguinte forma: é sorteado um índice i de operação, existem 3 opções nesta exemplificação, supondo que seja escolhido o índice 2, então substitui-se entre os dois cromossomos todas as operações com os índice $i = 2$, exemplificando Figura 13.

Toda a prole continua a ser factível, cada cromossomo representa uma solução exequível, pois como via de regra, cada *job* continua a ter uma quantidade de operações

Figura 13 – Crossover



Fonte: Elaborado pelo autor (2017)

sempre igual ao número de máquinas. Portanto, todas as possibilidade de escalonamento pertencentem ao universo factível de soluções.

3.4.2 Operadores de Mutações

O operador de mutação faz se necessário para preservar a diversidade genética da população pois ao decorrer de gerações a populacão tende a convergir para um ponto ótimo local. Dessa forma, o operador de crossover torna se incapaz de retirar a populacão da estagnação, sendo necessário a ocorrência de mutações. Zhang, Rao e Li (2008) propõe um conceito de mutação com dois operadores, descritas da seguinte forma:

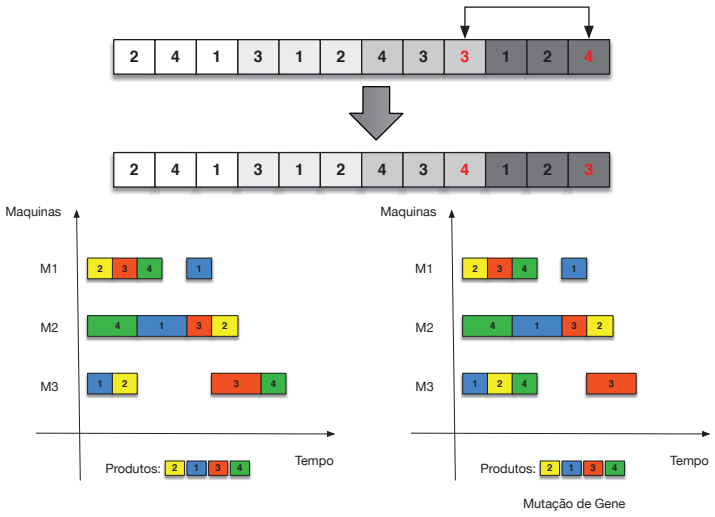
1. **Mutação de inversão:** inverte duas posições aleatoriamente selecionadas;

2. **Mutação de inserção:** recorta-se dois elementos vetoriais de forma aleatória, inserindo-os na segunda posição do vetor.

A mutação de inversão serve para manter a diversidade da população. A mutação de inserção é usada não só para produzir pequenas perturbações, mas também para realizar pesquisas intensivas para encontrar uma prole melhorada. Os operadores de mutação implementados fundamentaram-se nas diretrizes de Zhang, Rao e Li (2008). No entanto houve algumas personalizações realizadas na codificação, com objetivo de potencializar a diversidade cromossômica da população bem como estabelecer um significado na manipulação aleatória dos genes com o problema real, caracterizada da seguinte forma:

1. **Mutação de Gene:** A mutação de gene consiste em trocar de posição dois genes aleatoriamente selecionados em uma mesma instâncias cromossômicas ou não, conforme ilustrado Figura 14. Ao contrário do re-sequenciamento unitário proposto por Zhang, Rao e Li (2008), o número de vezes que esse re-sequenciamento genético é realizado, é arbitrado de modo aleatório. Deste modo, a mutação de gene pode se manifestar eventualmente de forma intensa ou amena.

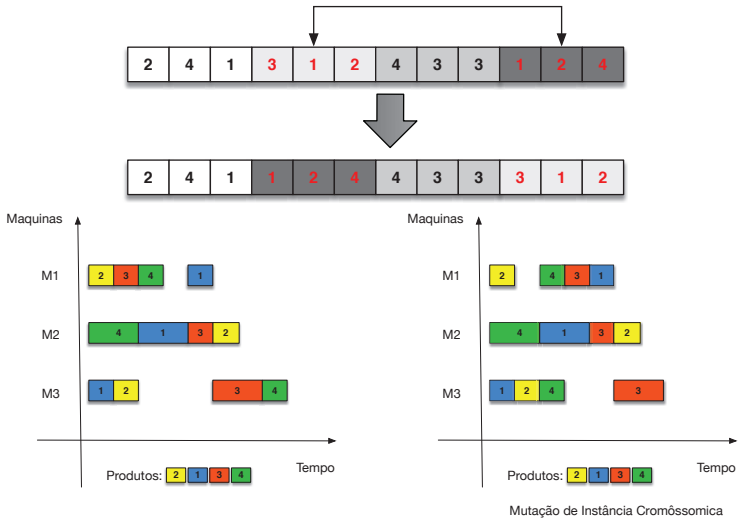
Figura 14 – Mutação de Gene



Fonte: Elaborado pelo autor (2018)

2. **Mutação de Instância Cromossômica:** A mutação de instância cromossômica consiste em selecionar aleatoriamente duas instâncias cromossômicas e invertê-las de posição, conforme ilustrado na Figura 15. O objetivo deste operador é proporcionar pequenas perturbações direcionais, invertendo instâncias cromossômicas possivelmente otimizadas, aumentando o espaço de busca de forma não tão aleatória.

Figura 15 – Mutação de Instância Cromossômica



Fonte: Elaborado pelo autor (2018)

3.5 Matrizes de Resultados

As matrizes de resultados apresentam o sequenciamento dos *jobs*, solução factível, tendo esta sequência sido otimizada ou não, em formato matricial. As iterações computacionais entre estas matrizes de resultados, constituem um processo de controle para os dados agregados a solução integral relativa ao valor da função objetivo do algoritmo. Sendo elas três matrizes apresentadas:

1. **Matriz de Operações Nas Máquinas**

$$MOM = \begin{bmatrix} M_{11}(O_i^k) & M_{12}(O_i^k) & M_{13}(O_i^k)\cdots & M_{1n}(O_i^k) \\ M_{21}(O_i^k) & M_{22}(O_i^k) & M_{23}(O_i^k)\cdots & M_{2n}(O_i^k) \\ M_{31}(O_i^k) & M_{32}(O_i^k) & M_{33}(O_i^k)\cdots & M_{3n}(O_i^k) \\ \vdots & \vdots & \vdots\ddots & \vdots \\ M_{m1}(O_i^k) & M_{m2}(O_i^k) & M_{m3}(O_i^k)\cdots & M_{mn}(O_i^k) \end{bmatrix}$$

2. **Matriz Tempo de Operações**

$$MTO = \begin{bmatrix} M_{11}(t_i^k) & M_{12}(t_i^k) & M_{13}(t_i^k)\cdots & M_{1n}(t_i^k) \\ M_{21}(t_i^k) & M_{22}(t_i^k) & M_{23}(t_i^k)\cdots & M_{2n}(t_i^k) \\ M_{31}(t_i^k) & M_{32}(t_i^k) & M_{33}(t_i^k)\cdots & M_{3n}(t_i^k) \\ \vdots & \vdots & \vdots\ddots & \vdots \\ M_{m1}(t_i^k) & M_{m2}(t_i^k) & M_{m3}(t_i^k)\cdots & M_{mn}(t_i^k) \end{bmatrix}$$

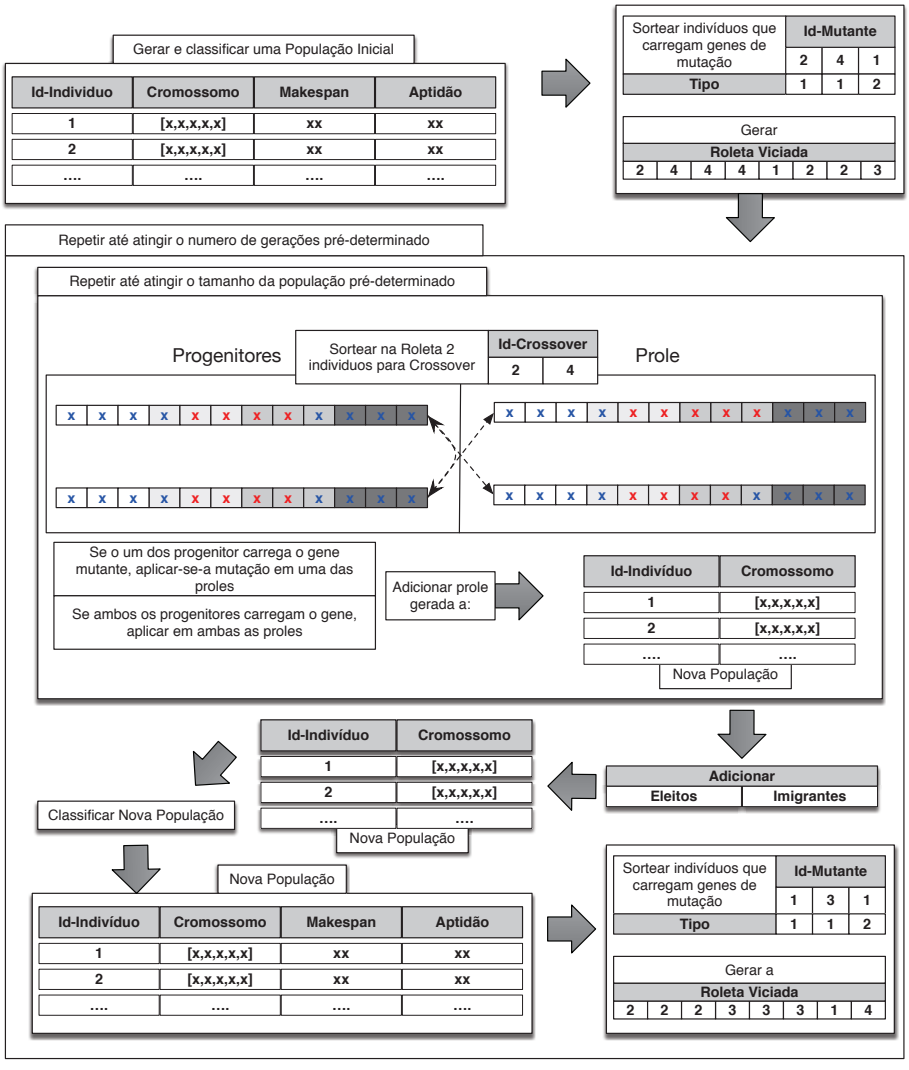
3. **Matriz de *Makespans***

$$MCn = \begin{bmatrix} M_{11}(C_i^k) & M_{12}(C_i^k) & M_{13}(C_i^k)\cdots & M_{1n}(C_i^k) \\ M_{21}(C_i^k) & M_{22}(C_i^k) & M_{23}(C_i^k)\cdots & M_{2n}(C_i^k) \\ M_{31}(C_i^k) & M_{32}(C_i^k) & M_{33}(C_i^k)\cdots & M_{3n}(C_i^k) \\ \vdots & \vdots & \vdots\ddots & \vdots \\ M_{m1}(C_i^k) & M_{m2}(C_i^k) & M_{m3}(C_i^k)\cdots & M_{mn}(C_i^k) \end{bmatrix}$$

3.6 Fluxo do Algoritmo Genético

O algoritmo foi elaborado subdividido em blocos de construção. Isto propiciou uma composição facilitada do fluxo de funcionamento do algoritmo genético. Onde cada objeto simula um comportamento inspirado em uma entidade real (exemplificado na Figura 16).

Figura 16 – Fluxo do Algoritmo Genético



Fonte: Elaborado pelo autor (2018)

As soluções encontradas a partir da elaboração de modelos para cálculos de otimização são geralmente estáticas, estas encontram-se limitadas a um conjunto de equações relacionadas ao modelo matemático.(YAZDANI *et al.*, 2017). Isto posto, é valido salientar que nos *benchmark sets* utilizados para elaboração e experimentação do modelo, inexiste uma data de entrega d_i^k para cada operação O_i^k. Assim sendo, adiantamentos e atrasos são respectivamente nulos ($E_i^k = 0$, $T_i^k = 0$). Portanto, foi necessário uma adaptação da função objetivo no fluxo de execução do algoritmo genético. O objetivo adaptado é determinar C_i^k para cada operação em um escalonamento exequível, minimizando o intervalo de tempo entre o início da primeira operação O_i^k e a última operação O_m^n, Equação 3.2, sujeito ás mesmas restrições da Equação 3.1.

$$Min \sum_{i=1}^{m} \sum_{k=1}^{n} (C_i^k) \tag{3.2}$$

3.7 Aplicabilidade e Análise dos Resultados

O *job shop schedule* recebeu uma atenção considerável na última década. Existindo disponível para comparar a qualidade dos diferentes métodos e abordagens, vários conjuntos de *benchmark sets* como referência. No esforço de evitar que possíveis publicações referenciem erroneamente o estado atual dos limites de otimização existentes. Hoorn (2018) compilou uma visão geral exata do estado atual para oito conjuntos de *benchmark sets*, sendo eles:

- `ft` (Muth, Thompson e Winters (1963))
- `la` (Lawrence (1984))
- `abz` (Adams, Balas e Zawack (1988))
- `orb` (Applegate e Cook (1991))
- `swv` (Storer, Wu e Vaccari (1992))
- `yn` (Yamada, T.; Nakano (1992))
- `ta` (Taillard (1993))
- `dmu` (Demirkol, Mehta e Uzsoy (1998))

Ao concluir o desenvolvimento, a rotina computacional possibilitou a racionalização de todo processo de escalonamento da produção. Sendo esta validação feita pela análise comparativa de um *benchmark sets* e seu limite atual com os limites obtidos através do algoritmo da aplicação. Hoorn (2018) mantém *benchmark sets* e seus limites atualizados em <http://jobshop.jjvh.nl>.

3.8 Plataforma de Desenvolvimento

Referente a plataforma de concepção, três tecnologias foram consideradas: i) Excel (VBA); ii) Linguagem R; iii) Python.

O VBA possui integração nativa com Excel, viabilizando uma visualização facilitada das entradas e saídas. Este ainda vem como padrão no Office 2016, e é uma alternativa única para se criar nativamente macros automatizada no pacote Office. Esta peculiaridade a faz ideal para o desenvolvimento de objetos de automatizações de "Office Applications". Porém, além de não ser uma plataforma *OpenSource*, a última atualização significativa na linguagem VBA, ocorreu em meados de 1999, tornando-a inapta quanto ao desenvolvimento de "algoritmos robustos", uma vez que esta linguagem não acompanhou os avanços científicos e tecnológico por mais de 17 anos.

Em se tratando de acompanhar os avanços científicos, R e Python são duas das linguagens mais integrada a comunidade científica. Porém, Python ao contrário do R é uma linguagem de multiparadigma: i) Orientação a objetos; ii) Programação imperativa; iii) Programação funcional; de propósito geral, com interpretador de script mais otimizado para o desenvolvimento de aplicações finais.

Seguindo esta abordagem, a rotina computacional deste trabalho foi desenvolvida integrando Python e Flask (pequeno *framework web* escrito em Python que facilita o desenvolvimento de aplicativos *web*). Desse modo, todo arcabouço científico de Python para análise de dados interativo como Jupyter Notebooks, NumPy, Pandas etc, pode ser aproveitado e implementado por meio de uma metodologia acessível desenvolvimento.

3.8.1 Linguagem de Modelagem Unificada

Na construção de rotinas computacionais onde o grau de complexidade se eleva, é desejável representar de forma visual o projeto de concepção. Neste sentido, a UML, Linguagem de Modelagem Unificada (do inglês, UML – *Unified Modeling Language*), foi criada com o propósito de estabelecer uma linguagem padrão para modelagem visual, com semântica e sintaxe normatizados, na concepção estrutural de projetos de software. Sendo uma linguagem para visualização, especificação, construção e documentação. Tem aplicações em fluxos do processo que vão além do desenvolvimento de software tendo sido empregada em diversas áreas como serviços financeiros, transportes e mapeamento de processos, telecomunicações e vendas.

A UML é composta por diagramas estruturais que compreende a existência e o posicionamento de entidades como classes, interfaces, colaborações e componentes. Segue abaixo as descrições mais comuns são o diagrama de caso de uso, diagrama de classes, diagrama de objetos, diagrama de colaboração, diagrama de sequência, diagrama de atividades, diagrama de estados, diagrama de componentes, diagrama de depuração, diagrama

de pacotes.

O paradigma orientado a objetos foi aplicado na concepção das rotinas computacionais deste trabalho. Com o objetivo de obter uma representação estrutural e visual das classes implementadas, elaborou-se o diagrama de classes para o projeto do AG fundamentado nas especificações de Booch, Rumbaugh e Jacobson (2006).

4 RESULTADOS E DISCUSSÕES

Este capítulo apresenta a implementação, validação e aplicação do modelo desenvolvido em instâncias de benchmark.

4.1 Elaboração da Rotina Computacional

Nesta etapa foram criadas as rotinas de solução para as partes fragmentadas do problema. Constituiu-se em uma etapa importante, pois foram realizados testes das rotinas, verificação da existência de erros, confiabilidade e precisão dos métodos.

Por se tratar de um modelo genérico foi necessário criar uma interação com o usuário para receber via teclado os parâmetros do modelo (número de indivíduos na população inicial, número de gerações, porcentagem de indivíduos na elite, porcentagem de indivíduos a sofrerem as mutações de cada tipo) e a seleção da instância a ser processada. Após o processamento dos dados disponibilizou-se na tela um gráfico de Gantt com o sequenciamento dos jobs, valor do *makespan* e gráficos do processo de evolução para permitir ao usuário proceder a análise da solução.

4.2 Variáveis

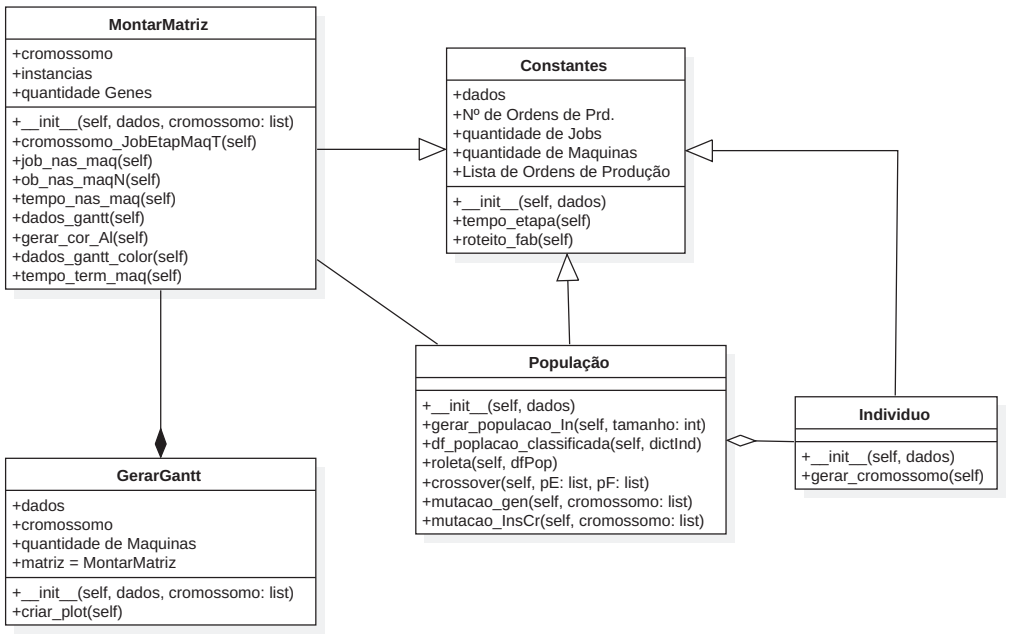
Os valores desconhecidos referentes ao problema proposto constituíram as variáveis de decisão. No caso desse problema, o sequenciamento de produção dos jobs, segundo as restrições do roteiro de fabricação. As variáveis de entrada armazenaram os parâmetros de cálculo fornecidos pelo usuário para otimização de um dos *benchmark sets* selecionado, sendo elas: matriz de dados; tamanho população; percentual de imigrantes; número de gerações; percentual de eleitos; percentual de mutações.

4.2.1 Algoritmo da Aplicação

Com intuito de conceber uma generalização computacional para o problema, utilizou-se de POO[1], um paradigma suportado e amplamente difundido na linguagem Python. O emprego desta lógica de concepção esta exemplificado na modelagem UML[2] da Figura 17. Este paradigma de programação fundamenta-se na composição e interação entre módulos codificados, denominados de objetos. A POO praticamente elimina a complexidade de modelar e representar o mundo real em componentes de software. Esta singularidade a fez popularmente reconhecida como a melhor forma de se eliminar o "*gap* semântico".

[1] **POO**: Programação Orientada a Objetos.
[2] **UML**: Linguagem de Modelagem Unificada.

Figura 17 – UML do Modelo Proposto



Fonte: Elaborado pelo autor (2018)

A Figura 16 apresenta as classes Constantes, MontarMatriz, Indivíduo e GerarGantt. São os blocos de construção do sistema, ou seja, é a estrutura que abstrai o conjunto de objetos com características similares do modelo. Cada classe em questão define o comportamento e o estado de seus respectivos objetos, através de métodos e atributos. Os objetos gerados a partir da classe MontarMatriz têm como funcionalidade interpretar o vetor cromossomo adjunto à matriz de dados, obtendo os *makespans* e matrizes sequências.

4.2.2 Implementação do algoritmo

Como salientado por Colin (2013), os algoritmos específicos são mais eficientes, por se beneficiarem das peculiaridades dos problemas para os quais foram concebidos. Dessa forma, cada classe possui operadores concebidos segundo as especificações do problema em questão. A implementação do `AG` em rotina computacional objetivou encontrar um conjunto de instruções que pudesse automatizar o processo de permutação dos dados de entrada bem como tratar aspectos de restrições, sequências e procedimentos computacionais.

4.2.3 Matriz de Dados

Os objetos gerados a partir da classe Constantes, têm como funcionalidade interpretar a matriz de dados no formato `.xlsx`[3], gerando as condições necessárias para que rotinas venham a ser processadas pelo computador. A definição do arquivo de entrada segue a configuração matricial adotada por Taillard (1993), apresentada como:

$$Matriz = \begin{bmatrix} T_{J1E1} & T_{J1E2} & T_{J1E3}\cdots & T_{J1En} \\ T_{J2E1} & T_{J2E2} & T_{J2E3}\cdots & T_{J1En} \\ T_{J3E1} & T_{J3E2} & T_{J3E3}\cdots & T_{J1En} \\ \vdots & \vdots & \vdots\ddots & \vdots \\ T_{JmE1} & T_{JmE2} & T_{JmE3}\cdots & T_{JmEn} \\ M_{J1E1} & M_{J1E2} & M_{J1E3}\cdots & M_{J1En} \\ M_{J2E1} & M_{J2E2} & M_{J2E3}\cdots & M_{J1En} \\ M_{J3E1} & M_{J3E2} & M_{J3E3}\cdots & M_{J1En} \\ \vdots & \vdots & \vdots\ddots & \vdots \\ M_{JmE1} & M_{JmE2} & M_{JmE3}\cdots & M_{JmEn} \end{bmatrix}$$

A matriz de dados é subdividida em tempos (T_{JmEn}) e máquinas (M_{JmEn}). Nesta matriz: T, representa o tempo da operação; J, o *job*, E a etapa do roteiro. Essa configuração permite uma interpretação compacta do roteiro de fabricação para cada *job* e seus respectivos tempos de processamento.

4.2.4 População Inicial e Roleta

Os operadores genéticos foram implementados na classe População que agrega a classe Indivíduo, em um relacionamento de associação com a classe MontarMatriz e de generalização com a classe Constantes, seguindo as especificações de concepção de publicações encontradas na literatura. No entanto, as implementações aplicadas na codificação dos operadores de seleção para crossover (roleta) e geração de indivíduos factíveis foram simplificações específicas para este trabalho. A implementação do método para gerar indivíduos factíveis seguiu três premissas: i) numero de posições vetoriais é igual à quantidade de jobs multiplicado pela quantidade de máquinas; ii) cada *job* deve repetir-se segundo o número de máquinas; iii) embaralha-se o vetor gerado. Deste modo, uma população inicial é gerada a partir de sucessivas repetições desse método. De forma semelhante, se deu a inserção de imigrantes em uma população. O fragmento de código a seguir foi utilizado para implementar a geração da população.

[3] `.xlsx`: Formato de Arquivo do Excel.

```
import numpy as np

def gerar_cromossomo():
        cromossomo = []
        for i in range(qtdMaq):
            cromossomo += list(qtdJobs)
        np.random.shuffle(cromossomo)
        return cromossomo
```

A implementação do método de seleção de crossover se deu por meio de uma roleta viciada, porém os indivíduos competiram entre si para determinar a maior fatia na roleta. A lógica empregada foi relativamente simples: i) obteve-se o *makespan* de todos os cromossomos; ii) classificou-se e ordenou-se do menor para o maior; iii) Obteve-se uma lista na qual cada posição representou o valor de aptidão em relação à lista classificatória. O fragmento de código a seguir foi utilizado para implementar a seleção para crossover.

```
import numpy as np

dicInd = {i: Indivíduo.gerar_cromossomo for i in range(tamanho)}
roletaDados = [max(makespans) - makespans[i] for i in range(tamanho)]

def roleta(roletaDados):
        listRoleta = []
        for i in range(len(dicInd)):
            qtd = roletaDados[i]
            for j in range(qtd):
                listRoleta += roletaDados.index[i]
        return listRoleta
# sorteios dos indivíduos para crossover
roleta = roleta(roletaDados)
sorteio1, sorteio2 = np.random.choice(roleta), np.random.choice(roleta)
```

4.3 Desenvolvimento da aplicação *Web*

Tendo como ponto de partida o modelo matemático do algoritmo genético e suas respectivas variáveis de entrada, através da integração entre Python e Html5 foi possível desenvolver uma interface simples porém amigável e de fácil utilização, cabendo ao usuário apenas a inserção dos parâmetros de entrada e seleção da instância a ser processada. A página *web* criada possibilita a inserção e consulta de dados, geração de relatório de performance do algoritmo, Gráfico de Gantt com a ocupação e folga das máquinas, além de gráficos e tabelas complementares para a análise da solução.

4.3.1 Interface da página *Web*

Ao acessar a página na url <http://iproductionscheduling.com> o usuário encontra as diretrizes e pequenos tutoriais que permitem a configuração de novas experimentações, além das apresentadas ao decorrer deste trabalho. Todo o algoritmo encontra-se implementado no núcleo desta página *web*, sendo possível escolher um *benchmark sets* desejado, e configurar os parâmetros de entrada para realizar cálculos. As telas da página são apresentadas no Apêndice A.

4.3.2 *Dashboard* dos Resultados

A página *web* exibe os resultados obtidos por meio de um *dashboard* [4] composto por um gráfico de linha, dois gráficos de dispersão e um gráfico de mapeamento de estado relativo por pixel. Os gráficos foram denominados por: *Fitness Plot*; Dispersão População Inicial; Dispersão População Final; Mapa Genético.

4.3.2.1 Gráfico Fitness Plot

Gráficos *Fitness Plot* é um gráfico de linhas que objetiva exibir tendências ao longo das gerações. O indivíduo com melhor *makespan* é plotado em função das gerações ao longo dos eixos Gerações e *Makspan*. Dessa forma, é possível analizar o processo de otimização e possibilidade de convergência do `AG`.

4.3.2.2 Gráfico de Dispersão População Inicial e Dispersão População Final

Estes gráficos objetivam pontuar dados em um eixo vertical (número do indivíduo) e horizontal (valor do *makespan*), com a intenção de exibir a diversidade populacional e o processo de convergência. Assim, o Gráfico Dispersão População Inicial demonstra uma população de soluções factíveis antes do processo de otimização e o Gráfico Dispersão População Final apresenta os indivíduos factíveis pós-otimização.

4.3.2.3 Gráfico Mapa Genético

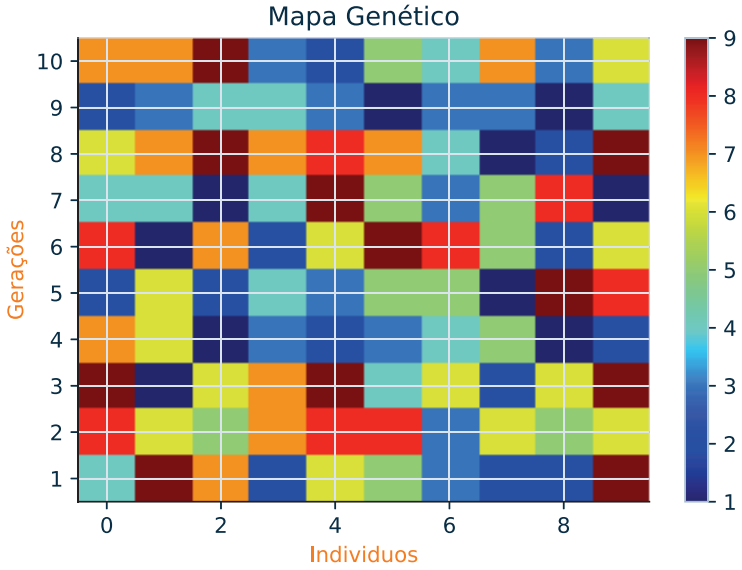
O histórico matricial de cálculo do `AG` resulta numa matriz de vetores empilhados, onde cada geração tem sua população representada por um `vetor` de n posições. Sendo n determinado pelo tamanho da população. Cada posição vetorial recebe o valor de aptidão, *makespan*, de um respectivo indivíduo. Isto posto, para fins de exemplificação, considere uma `matriz` 10×1 onde cada linha é um `vetor` de 10 posições com valores variando entre 1 e 10:

[4] ***dashboard*:** Painel de Indicadores

$$\texttt{matriz} = \begin{bmatrix} [7,7,9,3,2,5,4,7,3,6] \\ [2,3,4,4,3,1,3,3,1,4] \\ [6,7,9,7,8,7,4,1,2,9] \\ [4,4,1,4,9,5,3,5,8,1] \\ [8,1,7,2,6,9,8,5,2,6] \\ [2,6,2,4,3,5,5,1,9,8] \\ [7,6,1,3,2,3,4,5,1,2] \\ [9,1,6,7,9,4,6,2,6,9] \\ [8,6,5,7,8,8,3,6,5,6] \\ [4,9,7,2,6,5,3,2,2,9] \end{bmatrix}$$

O mapa genético tem por objetivo mostrar graficamente a posição relativa dos indivíduos ao decorrer das gerações em pixels tonalizados. Assim, classificam-se os valores contidos na `matriz` por meio de um espectro de cores que varia do azul ao vermelho. Se o valor é elevado a tonalidade do pixel tende ao vermelho, se o valor é diminuto sua tonalidade tende ao azul. O exemplo está demonstrado na Figura 18.

Figura 18 – Mapa Genético



Fonte: Elaborado pelo autor (2018)

Este tipo de gráfico permite uma análise detalhada das soluções concebidas a cada ciclo de cálculo `AG`. O mapeamento de estado do histórico de cada população, ao longo das gerações, torna possível a constatar possíveis impactos que cada operador genético exerce no processo de otimização.

4.4 Experimentação e Comportamento

Sabe-se que cada parâmetro de entrada impõe ao problema uma espécie de deslocamento comportamental no processo de otimização. Assim, é importante analisar os aspectos de influência para designá-los em conformidade às determinações do problema e dos recursos computacionais disponíveis. Foram realizadas exaustivas simulações com o código fonte da aplicação, considerando de cada vez um dos parâmetros.

Para execução do modelo utilizou-se um computador com processador Intel (R). Core (TM) i5-3210M CPU @2.30GHz, memória RAM 8,00 GB com sistema operacional de macOs High Sierra 64 bits. Diante desta perspectiva, foram executados escalonamentos de alguns *benchmark sets*. A seguir são apresentados os cenários e a respectiva análise comparativa dos resultados gerados.

4.4.1 Tamanho da População

O tamanho da população determina a cobertura do espaço de busca do problema, de modo a afetar desempenho global do algoritmo. Uma população pequena fornece uma cobertura pouco representativa do espaço de busca. Uma população grande para além de melhor representatividade no espaço de busca, previne convergências prematuras. No entanto, à medida em que se eleva o tamanho populacional, eleva-se também a exigência de recursos computacionais. Liden (2012) sustenta que:

> "A maioria dos trabalhos publicados tem uma fixação quase fetichista no número 100, talvez porque seja redondo, talvez por motivos históricos."

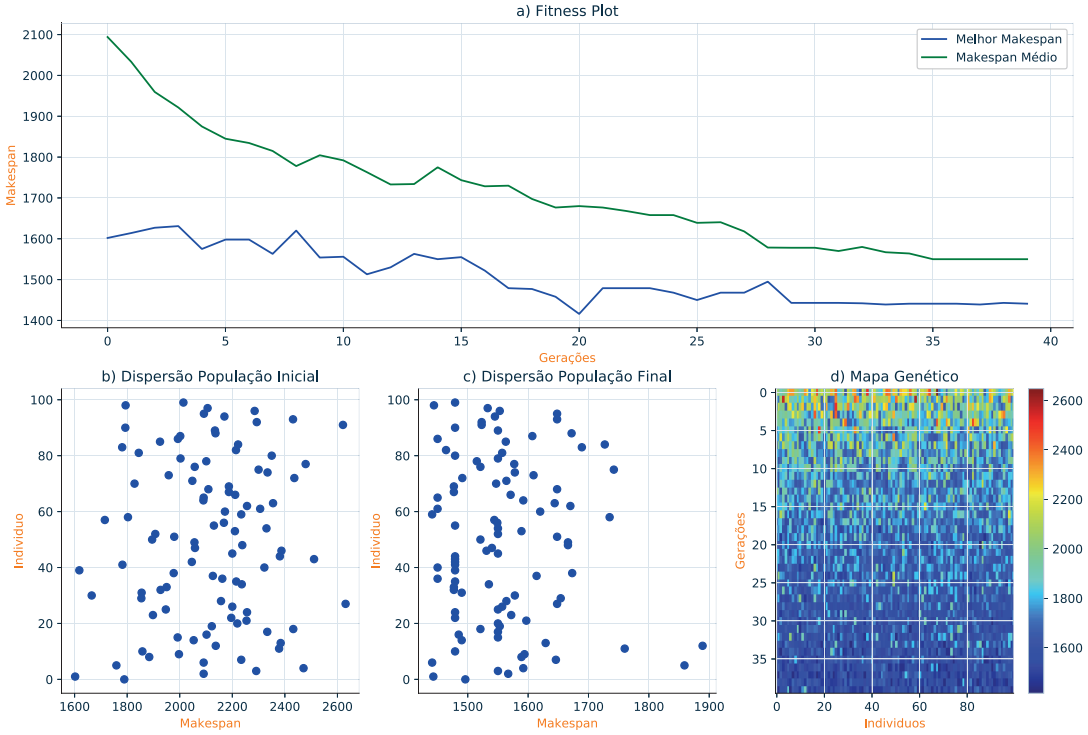
Assim, com o objetivo de avaliar o impactos dos demais parâmetros, a saber: elitismo, imigrantes, mutações, no comportamento da aplicação, adotou-se uma população de 100 indivíduos.

4.4.2 Taxa de Crossover

A taxa de crossover não é uma parâmetro de entrada, sendo esta automaticamente determinada pela competição entre os indivíduos como demonstrado na subseção 4.2.4. Para avaliar o impacto que o crossover tem no processo de otimização os seguintes parâmetros foram adotado: i) População = 100; ii) Imigrantes = 0; iii) Gerações = 40; iv) Eleitos = 0; iv) Mutação de Gene = 0; v) Mutação de Instância Cromossômica = 0. A matriz de dados adotada foi o *benchmark sets* `abz5` 10 $jobs \times 10$ máquinas. O comportamento pode ser observado na Figura 19.

Na Figura 19 a) *Fitness Plot* pode ser observado a curva de minimização do melhor *makespan* da população, bem como o valor de *makespan* médio populacional a cada

Figura 19 – Operador de Crossover



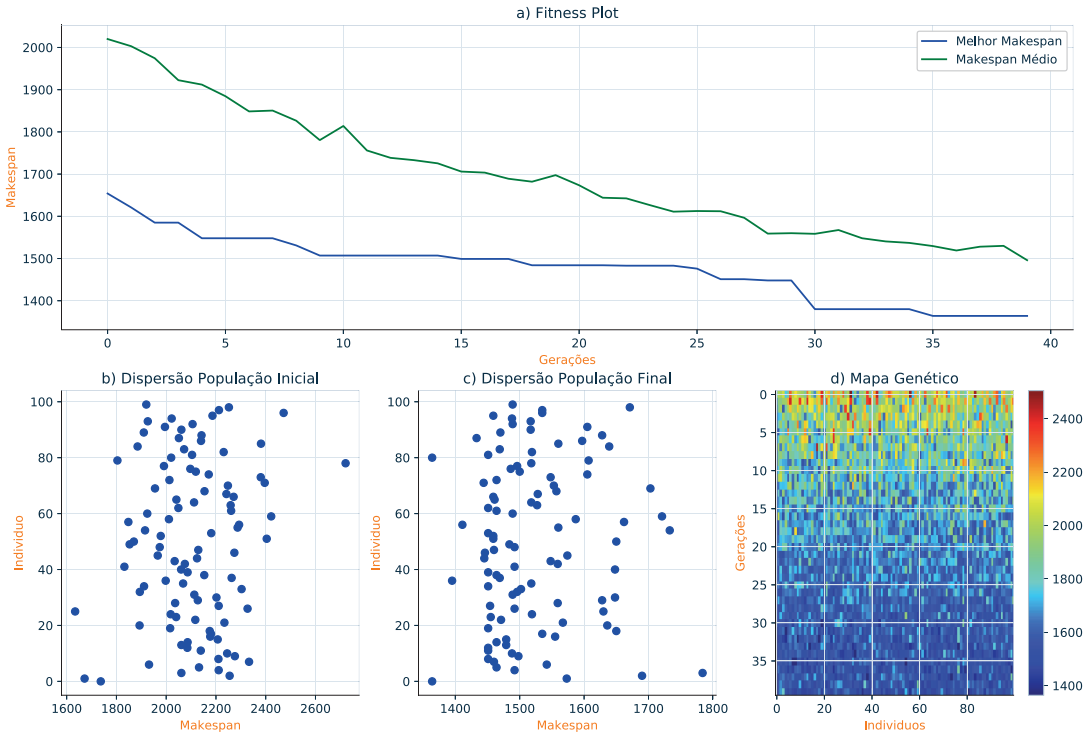
Fonte: Elaborado pelo autor (2018)

geração. Nota-se picos locais de elevação do *makespan*. Porém, globalmente , ao longo das gerações verifica-se a tendência de minimização. Os dois gráficos de dispersão apresentados na Figura 19 b) – c), mostram claramente uma população inicial diversicada com variados valores de *makespan*, e um população com valores de *makespan* que se tornam mais uniformes e menores no m do processo. O degradê no Mapa Genético , Figura 19 d), demostra que há o surgimento de melhores indivíduos a cada geração.

4.4.3 Taxa de Elitismo

O elitismo impede que a melhor solução encontrada a cada geração se perca. Para avaliar o impacto que o elitismo tem sob o processo de otimização, os seguintes parâmetros foram adotados: i) População = 100; ii) Imigrantes = 0 %; iii) Gerações = 40; iv) Eleitos = 1 %; iv) Mutação de Gene = 0 %; v) Mutação de Instância Cromossômica = 0 %. A matriz de dados adotada foi o *benchmark sets* `abz5` 10 *jobs* × 10 máquinas. O comportamento pode ser observado nas imagem, Figura 20.

Figura 20 – Taxa Elitismo



Fonte: Elaborado pelo autor (2018)

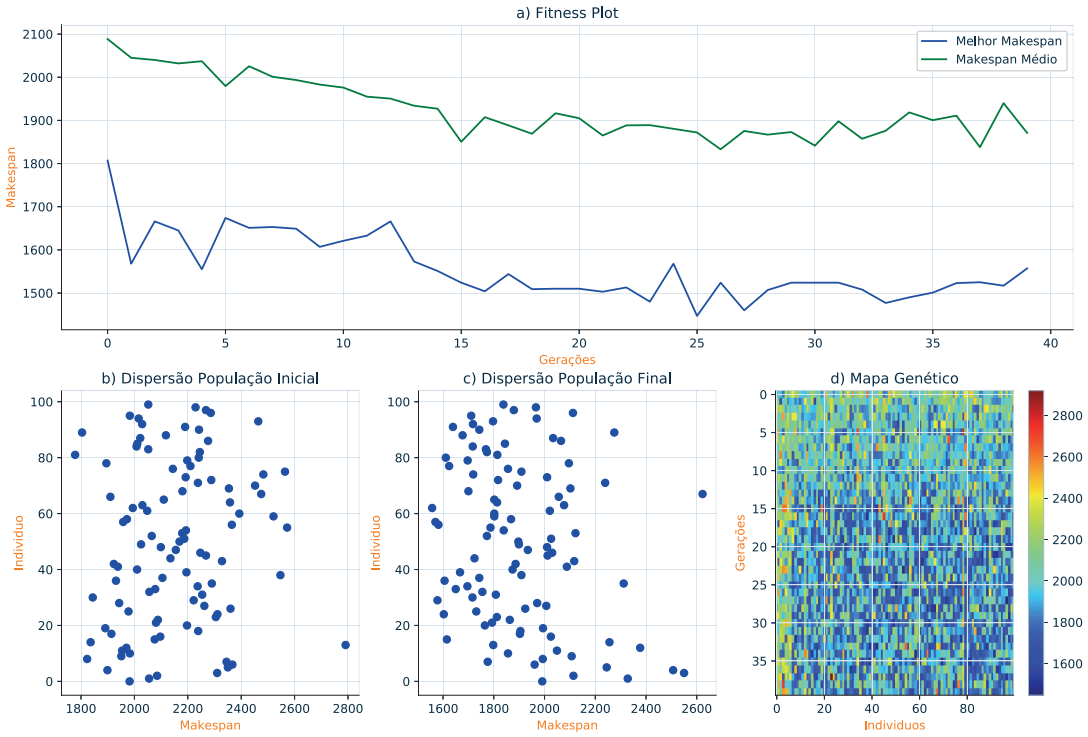
O comportamento pode ser observado na Figura 20 a). Observa-se que os picos de elevação local do *makespan* desapareceram. Isto é justicado pelo fato que o melhor indivíduo de cada geração é transportado para a geração subsequente. Isto implica, que uma taxa elevada de elitismo ocasionará uma convergência prematura. Os demais indicadores apresentam comportamento similar ao descrito subseção 4.4.2.

4.4.4 Taxa de Imigrantes

Para avaliar o impacto que a taxa de imigrantes tem sob o processo de otimização os seguintes parâmetros foi adotados: i) População = 100; ii) Imigrantes = 15 %; iii) Gerações = 40; iv) Eleitos = 0 %; iv) Mutação de Gene = 0 %; v) Mutação de Instância Cromossômica = 0 %. A matriz de dados adotada foi o *benchmark sets* `abz5` 10 *jobs* × 10 *maquinas*. O comportamento pode ser observado nas imagen, Figura 21.

O comportamento pode ser observado na Figura 21 a). Observa-se no gráfico que a aproximação do *makespan* médio em relação ao melhor *makespan* é praticamente inexis-

Figura 21 – Taxa Imigrantes



Fonte: Elaborado pelo autor (2018)

tente. Porém, continua a existir um fator de minimização e uniformização populacional, podendo isto ser comprovado pelos gráficos da Figura 21 b) – c), respectivamente. O Mapa Genético, Figura 21 d), apresenta em detalhes a presença dos imigrantes, introduzidos a cada nova geração, aparecendo separados do degradê de minimização gerado pelo operador de crossover.

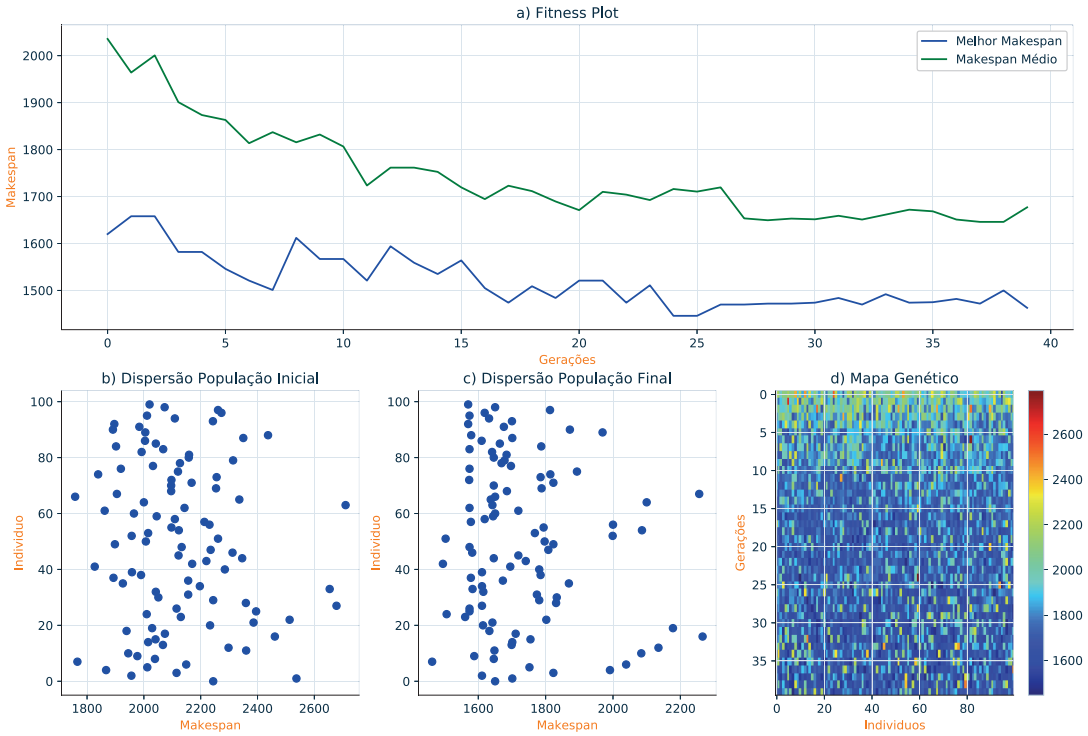
A cada nova geração novos indivíduos sempre hão de herdar as características de seus progenitores propiciando a uniformização populacional. Concluiu-se então, que taxa de imigrantes garantiu a renovação e variabilidade de indivíduos nas gerações.

4.4.5 Taxa de Mutação de Gene

Para avaliar o impacto que a taxa de mutação de gene tem sob o processo de otimização os seguintes parâmetros foi adotados: i) População = 100; ii) Imigrantes = 0 %; iii) Gerações = 40; iv) Eleitos = 0 %; iv) Mutação de Gene = 15 %; v) Mutação de Instância Cromossômica = 0 %. A matriz de dados adotada foi o *benchmark sets* `abz5`

10 *jobs* × 10 máquinas. O comportamento pode ser observado na imagem, Figura 22.

Figura 22 – Taxa de Mutação de Gene



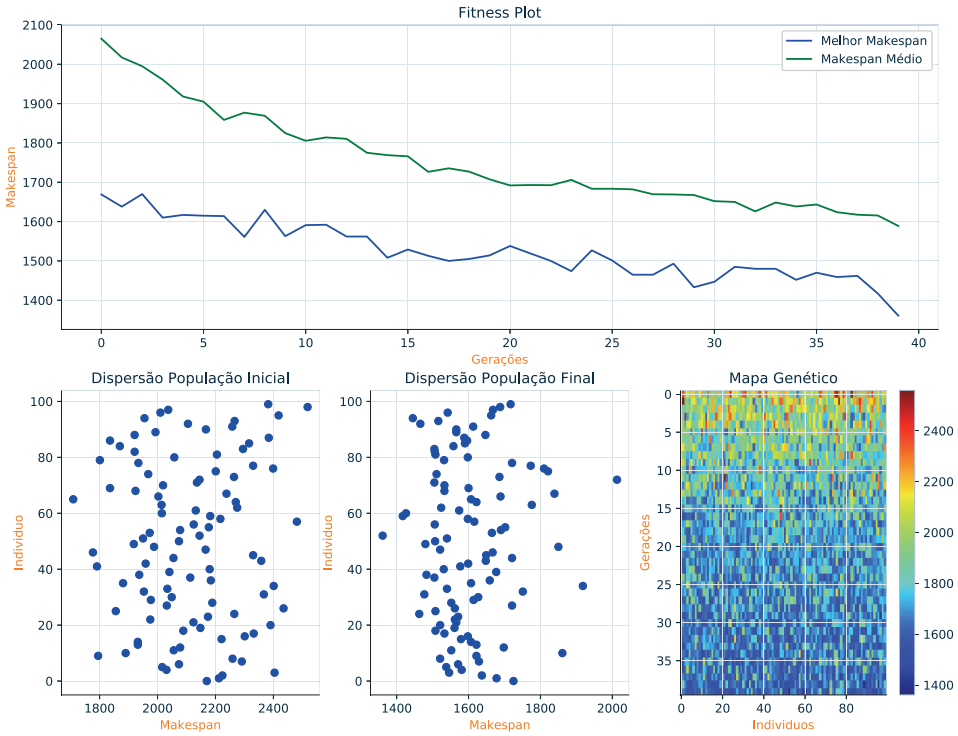
Fonte: Elaborado pelo autor (2018)

O comportamento pode ser observado na Figura 22. A Figura 22 a) demonstra que a aproximação do *makespan* médio em relação ao melhor *makespan* é praticamente inexistente. A Figura 22 b) – c) demonstra que a uniformização populacional foi afetada como esperado. No Mapa Genético, Figura 22 d), ainda pode ser notado a existência de um degradê, indicando que apesar de se ter populações menos uniforme ao decorrer das gerações, a uniformização continua a se fazer presente, mesmo em menor escala.

4.4.6 Taxa de Mutação de Instância Cromossômica

Para avaliar o impacto que a taxa de mutação de instância cromossômica tem sob o processo de otimização os seguintes parâmetros foram adotados: i) População = 100; ii) Imigrantes = 0 %; iii) Gerações = 40; iv) Eleitos = 0 %; iv) Mutação de Gene = 0 %; v) Mutação de Instância Cromossômica = 15 %. A matriz de dados adotada foi o *benchmark sets* `abz5` 10 *jobs* × 10 máquinas. O comportamento pode ser observado nas imagem, Figura 23.

Figura 23 – Taxa de Mutação da Instancia Cromossômica



Fonte: Elaborado pelo autor (2018)

O comportamento pode ser observado na Figura 23. Notam-se picos locais de elevação do *makespan*. Porém, globalmente, ao longo das gerações, o valor do *makespan* tende a minimizar-se. Os s de dispersão na Figura 23 b) – c) demonstram que a população inicial apresenta valores de *makespan* diversificados e que são minimizados e se tornam um pouco mais uniformes ao m do processo.

A uniformização da população continua a acontecer de forma acentuada ao contrário do efeito de amenização provocado pela mutação de gene, a mutação de instancia cromossômica tende apenas a retardar uma uniformização prematura. Comprovadamente o degradê no Mapa Genético, Figura 23 d), aparece de forma mais acentuada.

4.4.7 Otimização do *Benchmark Sets* `abz5` 10×10

Não há garantia que as soluções encontradas por um algoritmo genético sejam de fato ótimas. O princípio de concepção do modelo desenvolvido neste trabalho objetivou encontrar soluções aproximadas. Isto posto, a eciência de `AG` aqui proposto foi dimensionada a partir de experimentações comparativas entre o valor encontrado e o atual limite

ótimo *benchmark sets* conhecido. Caso fosse realizada uma busca exaustiva para 10 tarefas e 10 máquinas haveria um total de $10!^{10} = 3,95940866^{65}$ possíveis soluções. Os dados de entrada da instância *benchmark sets* `abz5` são apresentados a seguir:

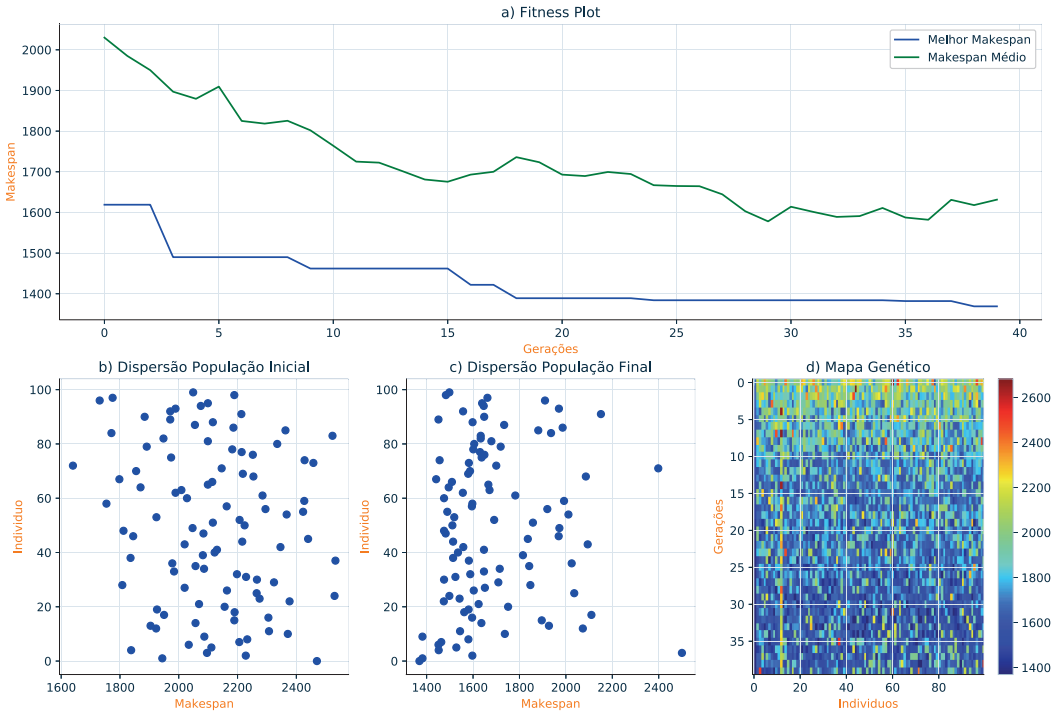
$$\texttt{abz5} = \begin{bmatrix} 88 & 68 & 94 & 99 & 67 & 89 & 77 & 99 & 86 & 92 \\ 72 & 50 & 69 & 75 & 94 & 66 & 92 & 82 & 94 & 63 \\ 83 & 61 & 83 & 65 & 64 & 85 & 78 & 85 & 55 & 77 \\ 94 & 68 & 61 & 99 & 54 & 75 & 66 & 76 & 63 & 67 \\ 69 & 88 & 82 & 95 & 99 & 67 & 95 & 68 & 67 & 86 \\ 99 & 81 & 64 & 66 & 80 & 80 & 69 & 62 & 79 & 88 \\ 50 & 86 & 97 & 96 & 95 & 97 & 66 & 99 & 52 & 71 \\ 98 & 73 & 82 & 51 & 71 & 94 & 85 & 62 & 95 & 79 \\ 94 & 71 & 81 & 85 & 66 & 90 & 76 & 58 & 93 & 97 \\ 50 & 59 & 82 & 67 & 56 & 96 & 58 & 81 & 59 & 96 \\ 5 & 9 & 7 & 6 & 2 & 3 & 10 & 8 & 1 & 4 \\ 6 & 4 & 7 & 5 & 3 & 9 & 1 & 2 & 8 & 10 \\ 10 & 9 & 1 & 2 & 7 & 6 & 8 & 5 & 3 & 4 \\ 8 & 3 & 2 & 5 & 4 & 7 & 6 & 1 & 10 & 9 \\ 4 & 5 & 10 & 9 & 1 & 3 & 7 & 6 & 8 & 2 \\ 2 & 5 & 6 & 7 & 9 & 3 & 8 & 10 & 4 & 1 \\ 8 & 2 & 5 & 4 & 1 & 9 & 3 & 6 & 7 & 10 \\ 5 & 7 & 4 & 3 & 2 & 6 & 8 & 1 & 9 & 10 \\ 1 & 7 & 4 & 8 & 2 & 3 & 5 & 6 & 9 & 10 \\ 4 & 1 & 2 & 9 & 8 & 10 & 7 & 5 & 6 & 3 \end{bmatrix}$$

Foi realizado a calibração das variáveis de entrada com os seguintes parâmetros: i) População = 100; ii) Imigrantes = 1 %; iii) Gerações = (40, 100, 200 300, 500); iv) Eleitos = 1 %; iv) Mutação de Gene = 2 %; v) Mutação de Instância Cromossômica = 10 %. Os resultados obtidos pelo algoritmo em comparação como os limites deste *benchmark sets* pode ser observado na Tabela 3. Com a variação do número de gerações os valores encontrados pelo `AG` são aproximados ao estado atual do limite ótimo.

A análise dos elementos da Figura 24 a) – d) sugeriu que uma quantidade de gerações superior a 40 potencializaria as possibilidades de encontrar uma solução mais aproximada ao atual limite. Desta maneira, a cada nova experimentação aumentou-se o número de gerações com o objetivo de melhorar as soluções encontradas pelo `AG`. A hipótese de que um número superior de gerações é uma garantia de melhores resultados se mostrou verdadeira. Isto deve-se ao fato de que apesar das gerações subsequente serem concebidas de forma aleatória, esta aleatoriedade é direcional e privilegia indivíduos com *makespans* diminutos. Assim sendo, convém elevar a quantidade de gerações até que haja

a estagnação do processo de melhoramento genético. Entretanto, conforme observado na Tabela 3, os valores encontrados forma apenas aproximados.

Figura 24 – Performance com *Benchmark Sets* `abz5` 40 Gerações



Fonte: Elaborado pelo autor (2018)

Tabela 3 – Resultados Para o *Benchmark Sets* `abz5`

Gerações	Tempo	Valor Encontrado	Eficiência
40	1,28 minutos	1397	88,33 %
100	4,83 minutos	1321	93,41 %
200	8,17 minutos	1294	95,58 %
300	12,73 minutos	1285	96,03 %
500	59,09 minutos	1260	97,93 %

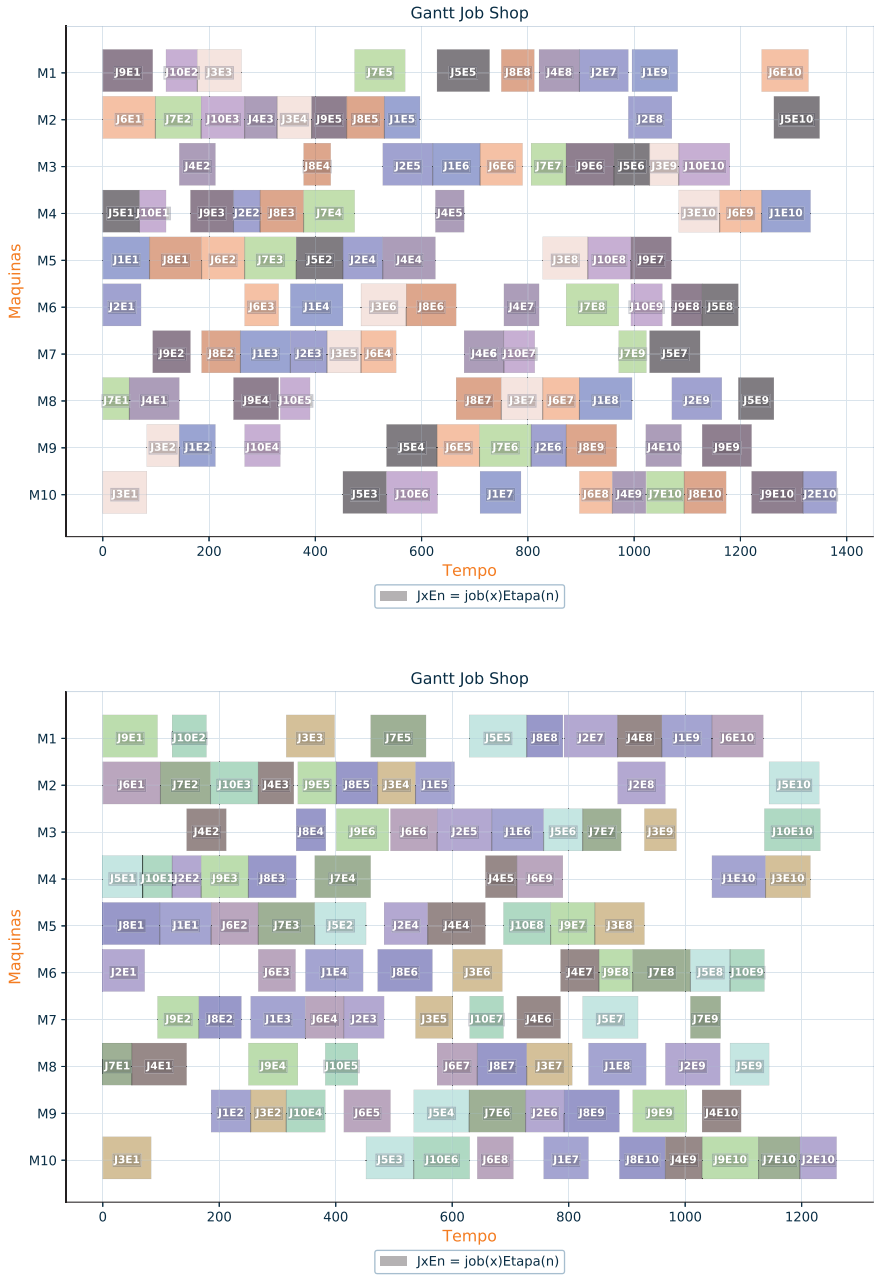
Fonte: Elaborada pelo autor (2018).

Nota: Limite ótimo é igual 1234, Hoorn (2018).

Muito embora o algoritmo não tenha encontrado um valor igual ao limite hoje existente, o mesmo cumpre o propósito de sua concepção, pois as soluções obtidas são

aproximadas, havendo um refinamento do valor encontrado à medida em que foram ofertados mais recursos computacionais. Algumas operações que necessitam de mais tempo para sua conclusão, podem causar atrasos cascata em muitos *jobs*. O efeito cumulativo exerce uma influência sob o comportamento do sequenciamento, dificultando o processo de otimização. A Figura 25 apresenta comparativamente os resultados s com 40 e 500 Gerações.

Figura 25 – Gantt *Benchmark Sets* `abz5` 40 e 500 gerações



Fonte: Elaborado pelo autor (2018)

4.4.8 Teste de precisão com o *Benchmark Sets* `ft06` 6×6

Esta etapa objetivou verificar a precisão do algoritmo, em repetidas simulações com mesmos parâmetros de entrada. Portanto, adotou-se uma instancia com menor exigência de recurso computacional, *benchmark sets* `ft06`. Caso fosse realizada uma busca para 6 tarefas e 6 máquinas há um total de $6!^6 = 1,39314070^{17}$ possíveis soluções.

$$\texttt{ft06} = \begin{bmatrix} 1 & 3 & 6 & 7 & 3 & 6 \\ 8 & 5 & 10 & 10 & 10 & 4 \\ 5 & 4 & 8 & 9 & 1 & 7 \\ 5 & 5 & 5 & 3 & 8 & 9 \\ 9 & 3 & 5 & 4 & 3 & 1 \\ 3 & 3 & 9 & 10 & 4 & 1 \\ 3 & 1 & 2 & 4 & 6 & 5 \\ 2 & 3 & 5 & 6 & 1 & 4 \\ 3 & 4 & 6 & 1 & 2 & 5 \\ 2 & 1 & 3 & 4 & 5 & 6 \\ 3 & 2 & 5 & 6 & 1 & 4 \\ 2 & 4 & 6 & 1 & 5 & 3 \end{bmatrix}$$

Sendo o algoritmo genético uma técnica de busca aproximada ao ótimo, um bom teste de precisão consiste em verificar o quão próximo as soluções encontradas circundam o limite ótimo. Nas simulações os seguintes parâmetros foram adotados: i) População = 100; ii) Imigrantes = 3 %; iii) Gerações = 40; iv) Eleitos = 1 %; iv) Mutação de Gene = 7 %; v) Mutação de Instância Cromossômica = 3 %. Um total de 10 replicações consecutivas foi realizadas, não havendo nenhum tipo de alteração nos parâmetros de entrada. Os resultados obtidos em comparação ao estado atual do limite do *Benchmark Sets* `ft06`, estão dispostos na Tabela 4. O desempenho de uma das replicações pode ser analisado com base na Figura 26 a) – d).

Observa-se mesmo encontrando o limite ótimo repetidas vezes, não necessariamente a solução encontrada é a mesma. Neste tipo de problema vários são os caminhos que levam a um mesmo limite. Apesar de se utilizar os mesmos parâmetros de entrada nas 10 replicações, cada uma das replicações iniciou com uma população única, e portanto com diferentes universos de possibilidades. A Figura 27 apresenta graficamente os resultados de duas soluções que alcançaram *makespan* igual a 55. Ambas as soluções encontraram o limite ótimo. No entanto como o de *gantt* demonstra são soluções com sequenciamento distinto.

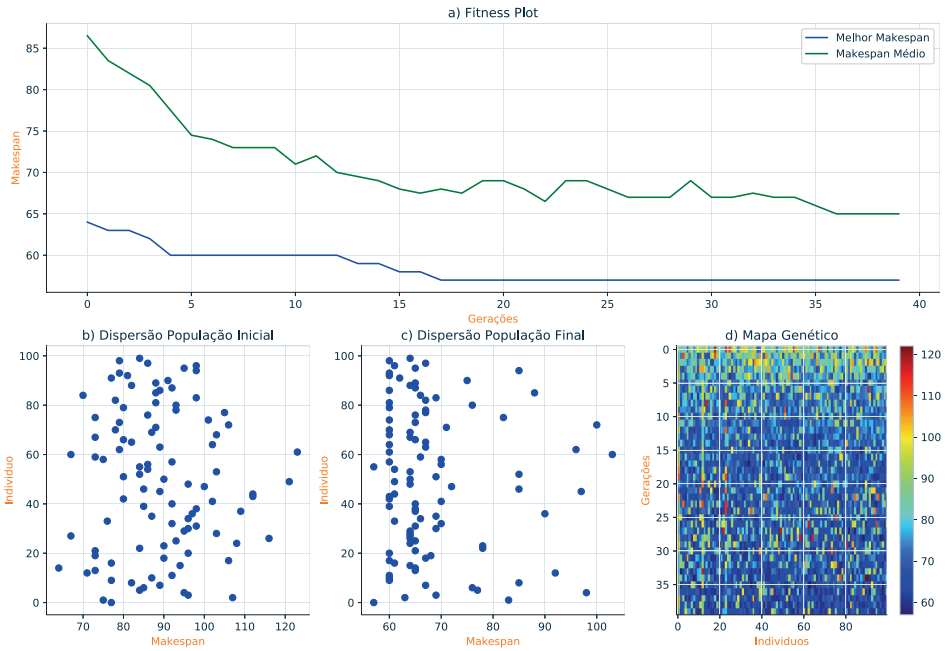
Tabela 4 – Resultados Para o *Benchmark Sets* `ft06`

Gerações	Tempo	Valor Encontrado	Eficiência
40	1,32 minutos	55	100,00 %
40	1,48 minutos	55	100,00 %
40	1,31 minutos	57	96,49 %
40	1,31 minutos	57	96,49 %
40	1,33 minutos	55	100,00 %
40	1,38 minutos	55	100 %
40	1,34 minutos	55	100,00 %
40	1,32 minutos	58	94,82 %
40	1,36 minutos	58	94,82 %
40	1,25 minutos	57	96,49 %

Fonte: Elaborada pelo autor (2018).

Nota: Limite ótimo é igual 55, Hoorn (2018)..

Figura 26 – Performance com *Benchmark Sets* `ft06` 40 Gerações



Fonte: Elaborado pelo autor (2018)

Figura 27 – Gantt *Benchmark Sets* `ft06` 40 Gerações

Fonte: Elaborado pelo autor (2018)

4.5 Considerações Finais

Ao decorrer da etapa de validação e experimentação, observou-se que a aplicação apresentou um comportamento estável, atingindo o propósito de sua concepção. Os resultados obtidos a partir do núcleo da rotina computacional obteve um desfecho com-

parável aos encontrado na literatura, evidenciando a correta modelagem e programação das rotinas.

É valido ressaltar que o objetivo primordial deste trabalho residiu na concepção de uma plataforma de aprendizagem metodológica de um modelo heurístico com base no `AG` para o planejamento da programação de sistemas de manufatura do tipo *job shop*. Isto posto, não foi portanto apresentado neste trabalho as análises comparativas e aprofundadas do desempenho do modelo `AG` proposto em contraposição aos ótimos encontrados para as instâncias de *Benchmark Sets* disponibilizados na página *web*.

Os experimentos apresentados e discutidos no texto tiveram a pretensão de parametrizar o comportamento dos operadores genéticos, bem como avaliar a capacidade do algoritmo em circundar os limites ótimos já publicados na literatura. Foram testadas apenas duas instâncias. No entanto, indagações referente às replicações dos experimentos aqui apresentados, bem como possíveis novas experimentações com as instâncias disponibilizadas na plataforma podem ser realizadas acessando <http://iproductionscheduling.com>.

É fato que ao se elevar o número de *jobs* diculta-se a viabilidade de se obter um escalonamento ótimo, ou seja, o arranjo dos trabalhos nas máquinas de modo a satisfazer às restrições de precedência cujo processamento deve ocorrer no menor tempo possível.

5 CONCLUSÃO

O *job shop* é um problema estabelecido sob as instâncias tático/operacional do planejamento de produção o e tem como propósito alocar os recursos produtivos. Sendo assim, o número elevado de variáveis, bem como restrições impostas a esse tipo de problema, exige um alto grau de conhecimento técnico específico do fluxo de trabalho, de técnicas de otimização e de aparatos computacionais, para a implementação de uma metodologia de solução. No entanto, a curva de aprendizagem costuma ser longa, pois não se encontra na literatura relatos de plataformas de aprendizado para experimentações de otimizações simuladas de escalas de produção.

Neste trabalho, conforme planejado, foi desenvolvido um aplicativo *web* tendo um algoritmo heurístico em seu núcleo, com a capacidade de realizar experimentações simuladas para otimização do *makespan*, à partir de uma interação com o usuário para seleção do *benchmark set* e parâmetros do algoritmo. Assim, buscou-se durante o desenvolvimento do trabalho fazer a apresentação da modelagem matemática do problema, para em seguida fazer a elaboração do algoritmo e sua correspondente transcrição em modelagem em UML e para a linguagem Python que viabilizou o projeto da página *web*.

As experimentações realizadas com dois *benchmark sets* (`ft6 6x6 e abz5 10x10`) comprovaram a eficácia do algoritmo em ter resultados que se aproximam dos limites ótimos, comprovando uma correta modelagem de concepção. Portanto, este trabalho pode auxiliar na curva de aprendizado de estudos do escalonamento da produção com fluxo de trabalho *job shop*, à partir de experimentações simuladas no ambiente disponibilizado com operação online.

Diante deste contexto, este trabalho deixa como contribuição o aplicativo *web* que está disponível em <http://iproductionscheduling.com> para que as pessoas interessadas no problema possam fazer experimentações *online* com os *benchmark sets* compilados do trabalho de Hoorn (2018) para o problema *Job Shop*. O resultado da otimização é mostrado explicitando o cromossomo da melhor solução que é apresentada em um gráfico de Gantt. Além disto, outros gráficos são disponibilizados para um melhor entendimento do funcionamento do algoritmo genético implementado no núcleo da aplicação. O usuário tem informações sobre a instância processada, limites ótimos, valor do *makespan* da melhor solução, podendo acompanhar o tempo para finalização do processamento. O usuário também pode consultar os dados das diferentes instancias que poderá usar em suas experimentações.

5.1 Possíveis Melhorias

Na Pesquisa Operacional a hipótese da certeza sobre os valores dos coeficientes é denominada análise de sensibilidade. O presente trabalho pode ser melhorado adotando uma análise de sensibilidade sob às diretrizes dos parâmetros de entrada com os operadores genético.

5.2 Sugestões Para Trabalhos Futuros

Uma dos pressupostos em problemas de *job shop schedule* é o tempo variável no processamento das etapas do roteiro de fabricação. Com o objetivo de amenizar esta hipótese realiza-se uma análise pós-otimização verificando as possíveis variações para cima e para baixo destes valores, observando o impacto das possíveis variações na solução otimizada. Assim, sugere-se estender metodologia apresentada neste trabalho ao problema *job shop* flexível e tempo variável, tendo como ponto de partida os seguintes trabalhos:

- Lei (2010)
- Abdullah e Abdolrazzagh-Nezhad (2014)
- Liu, Fan e Liu (2015)
- Gao *et al.* (2016)

REFERÊNCIAS

ABDULLAH, S.; ABDOLRAZZAGH-NEZHAD, M. Fuzzy job-shop scheduling problems: A review. *Inf. Sci. (Ny).*, Elsevier Inc., v. 278, p. 380–407, 2014. ISSN 00200255. Disponível em: <http://dx.doi.org/10.1016/j.ins.2014.03.060>. Citado na página 78.

ADAMS, J.; BALAS, E.; ZAWACK, D. The Shifting Bottleneck Procedure for Job Shop Scheduling. *Manage. Sci.*, v. 34, n. 3, p. 391–401, mar 1988. ISSN 0025-1909. Disponível em: <http://pubsonline.informs.org/doi/abs/10.1287/mnsc.34.3.391>. Citado na página 55.

APPLEGATE, D.; COOK, W. A Computational Study of the Job-Shop Scheduling Problem. *ORSA J. Comput.*, v. 3, n. 2, p. 149–156, 1991. ISSN 0899-1499. Disponível em: <http://pubsonline.informs.org/doi/abs/10.1287/ijoc.3.2.149>. Citado na página 55.

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. *Pesquisa Operacional*. [s.n.], 2007. 523 p. ISBN 8535251936. Disponível em: <http://books.google.com/books?id=sB___Fi8rprEC&pgis=1>. Citado 3 vezes nas páginas 29, 30 e 35.

BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. *Handbook of Evolutionary Computation*. Institute of Physics Pub., 1997. (Computational intelligence library, v. 1). ISBN 9780750303927. Disponível em: <https://books.google.com.br/books?id=kgqGQgAACAAJ>. Citado na página 38.

BAPTISTE, P.; FLAMINI, M.; SOURD, F. Lagrangian bounds for just-in-time job-shop scheduling. *Comput. Oper. Res.*, v. 35, n. 3, p. 906–915, 2008. ISSN 03050548. Disponível em: <https://doi.org/10.1016/j.cor.2006.05.009>. Citado 2 vezes nas páginas 45 e 47.

BEYER, H.-G. *The Theory of Evolution Strategies*. Springer Berlin Heidelberg, 2001. 400 p. ISSN 1063-6560. ISBN 9783662043783. Disponível em: <https://link.springer.com/book/10.1007%2F978-3-662-04378-3>. Citado na página 39.

BEYER, H.-G.; BEYER, H.-G.; SCHWEFEL, H.-P.; SCHWEFEL, H.-P. Evolution strategies A comprehensive introduction. *Nat. Comput.*, v. 1, n. 1, p. 3 – 52, 2002. ISSN 1572-9796. Disponível em: <http://link.springer.com/10.1023/A:1015059928466>. Citado 2 vezes nas páginas 39 e 41.

BIERWIRTH, C.; KUHPFAHL, J. Extended GRASP for the job shop scheduling problem with total weighted tardiness objective. *Eur. J. Oper. Res.*, Elsevier B.V., v. 261, n. 3, p. 835–848, 2017. ISSN 03772217. Disponível em: <http://dx.doi.org/10.1016/j.ejor.2017.03.030>. Citado 3 vezes nas páginas 16, 26 e 28.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *UML guia do usuario*. [S.l.]: Elsevier, 2006. ISBN 9788535217841. Citado na página 57.

ÇALI, B.; BULKAN, S. A research survey: review of AI solution strategies of job shop scheduling problem. *J. Intell. Manuf.*, v. 26, n. 5, p. 961–973, 2015. ISSN 15728145. Citado na página 47.

CARVALHO, M.; SILVA, F. O.; FERNANDES, C. O planejamento da manufatura: práticas industriais e métodos de otimização. *Gestão e Produção*, v. 5, n. 1, p. 34–59, apr 1998. ISSN 0104-530X. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X1998000100003&lng=pt&tlng=pt>. Citado na página 23.

CHEN, J. C.; CHEN, Y.-Y.; CHEN, T.-L.; LIN, J. Z. Comparison of simulated annealing and tabu-search algorithms in advanced planning and scheduling systems for TFT-LCD colour filter fabs. *Int. J. Comput. Integr. Manuf.*, v. 3052, n. March, p. 1–19, 2016. ISSN 0951-192X. Disponível em: <http://www.tandfonline.com/doi/full/10.1080/0951192X.2016.1145805>. Citado na página 44.

COLIN, E. C. *Pesquisa operacional: 170 aplicaçoes em estratégia, finanças, logística, produçao, marketing e vendas*. [S.l.]: Livros Técnicos e Científicos, 2013. 502 p. ISBN 978852161190. Citado 9 vezes nas páginas 29, 30, 31, 32, 33, 34, 35, 37 e 59.

CORREA, H. L.; GIANESI, I. G. N.; CAON, M. *Planejamento, programação e controle da produção: MRP II/ERP : conceitos, uso e implantaçâo*. São Paulo: [s.n.], 2014. Citado na página 22.

DÁVALOS, R. Uma abordagem do ensino de pesquisa operacional baseada no uso de recursos computacionais. *XXII Encontro Nac. Eng. Produção*, p. 1–8, 2002. Disponível em: <http://www.abepro.org.br/biblioteca>. Citado na página 29.

DAVIS, L. Job Shop Scheduling with Genetic Algorithms. In: *Proc. 1st Int. Conf. Genet. Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985. p. 136–140. ISBN 0-8058-0426-9. Disponível em: <http://dl.acm.org/citation.cfm?id=645511.657084>. Citado na página 44.

DEFERSHA, F. M.; CHEN, M. Jobshop lot streaming with routing flexibility, sequence-dependent setups, machine release dates and lag time. *Int. J. Prod. Res.*, v. 50, n. 8, p. 2331–2352, 2012. ISSN 00207543. Citado na página 45.

DELLAGI, S.; CHELBI, A.; TRABELSI, W. Joint integrated production-maintenance policy with production plan smoothing through production rate control. *J. Manuf. Syst.*, v. 42, p. 262–270, 2017. ISSN 02786125. Disponível em: <https://doi.org/10.1016/j.jmsy.2016.12.013>. Citado na página 23.

DEMIRKOL, E.; MEHTA, S.; UZSOY, R. Theory and Methodology Benchmarks for shop scheduling problems. *Eur. J. Oper. Res.*, v. 109, p. 137–14, 1998. ISSN 03772217. Citado na página 55.

ELMI, A.; TOPALOGLU, S. Multi-degree cyclic flow shop robotic cell scheduling problem: Ant colony optimization. *Comput. Oper. Res.*, v. 73, n. August, p. 67–83, 2016. ISSN 03050548. Disponível em: <http://dx.doi.org/10.1016/j.cie.2017.08.005>. Citado na página 25.

EROL, S.; JÄGER, A.; HOLD, P.; OTT, K.; SIHN, W. ScienceDirect Tangible Industry 4.0: a scenario-based approach to learning for the future of production. *Procedia CIRP*, v. 54, p. 13–18, 2016. ISSN 2212-8271. Disponível em: <www.sciencedirect.com>. Citado na página 15.

EROL, S.; SIHN, W. Intelligent production planning and control in the cloud towards a scalable software architecture. *Procedia CIRP*, v. 62, p. 571–576, 2017. ISSN 2212-8271. Disponível em: <www.sciencedirect.com>. Citado na página 15.

FAÉ, C. S.; ERHART, A. Desafios e tendencias na aplicacao de sistemas APS no Brasil. *Rev. Mundo Logística*, p. 197–206, 2009. Citado 2 vezes nas páginas 42 e 43.

FEOFILOFF, P. Algoritmos de programaçao linear. *Ed. da Univ. Sao Paulo*, 1999. Disponível em: <http://www.ime.usp.br/$\sim$pf/prog-lin/>. Citado 2 vezes nas páginas 37 e 38.

FERNANDES, F. C. F.; FILHO, M. G. *Planejamento e controle da produção: dos fundamentos ao essencial*. [S.l.: s.n.], 2010. 275 p. ISBN 8522458715, 9788522458714. Citado 5 vezes nas páginas 20, 21, 22, 23 e 29.

FREITAG, M.; HILDEBRANDT, T. Automatic design of scheduling rules for complex manufacturing systems by multi-objective simulation-based optimization. *CIRP Ann. - Manuf. Technol.*, CIRP, v. 65, n. 1, p. 433–436, 2016. ISSN 17260604. Disponível em: <http://dx.doi.org/10.1016/j.cirp.2016.04.066>. Citado na página 24.

FUCHIGAMI, H. Y.; RANGEL, S. A survey of case studies in production scheduling: Analysis and perspectives. *J. Comput. Sci.*, Elsevier B.V., 2017. ISSN 18777503. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1877750317300741>. Citado na página 15.

FURINI, F.; LJUBIĆ, I.; SINNL, M. An Effective Dynamic Programming Algorithm for the Minimum-Cost Maximal Knapsack Packing Problem. *Eur. J. Oper. Res.*, 2017. ISSN 03772217. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0377221717302928>. Citado na página 47.

GANGA, G. M. D. *Trabalho de Conclusão de Curso (TCC) na Engenharia de Produção: um Guia Prático de Conteúdo e Forma*. [S.l.: s.n.], 2012. 384 p. ISBN 9788522471164. Citado na página 46.

GAO, K. Z.; SUGANTHAN, P. N.; PAN, Q. K.; TASGETIREN, M. F.; SADOLLAH, A. Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. *Knowledge-Based Syst.*, Elsevier B.V., v. 109, p. 1–16, 2016. ISSN 09507051. Disponível em: <http://dx.doi.org/10.1016/j.knosys.2016.06.014>. Citado na página 78.

GIROTTI, L. J.; MESQUITA, M. A. Simulação e estudos de caso no ensino de planejamento e controle da produção: um survey com professores da engenharia de produção Palavras-chave. v. 26, n. 1, p. 176–189, 2016. ISSN 01036513. Disponível em: <http://dx.doi.org/10.1590/0103-6513.145013>. Citado na página 44.

GOLDBERG, D. E. *Genetic Algorithms*. Pearson Education, 2006. ISBN 9788177588293. Disponível em: <https://books.google.com.br/books?id=6gzS07Sv9hoC>. Citado na página 40.

GRAHAM, R.; LAWLER, E.; LENSTRA, J.; KAN, A. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. In: *Ann. Discret. Math.* Elsevier, 1979. v. 5, n. C, p. 287–326. ISBN 9780080867670. Disponível em: <http://www.sciencedirect.com/science/article/pii/S016750600870356Xhttp://

linkinghub.elsevier.com/retrieve/pii/S016750600870356X>. Citado 2 vezes nas páginas 23 e 25.

GÜÇEMIR, H.; SELIM, H. Customer centric production planning and control in job shops: A simulation optimization approach. *J. Manuf. Syst.*, v. 43, n. Part 1, p. 100–116, 2017. ISSN 0278-6125. Disponível em: <http://10.0.3.248/j.jmsy.2017.02.004%0Ahttp://search.ebscohost.com/login.aspx?direct=true&db=edselp&AN=S0278612517300171&site=eds-live>. Citado 3 vezes nas páginas 15, 24 e 48.

GUERRINI, F. M.; BELHOT, R. V.; JÚNIO, W. A. *Planejamento e Controle da Produçao: Projeto e operações de Sistema*. ELSEVIER EDITORA, 2014. 264 p. ISBN 9788535268072. Disponível em: <https://books.google.com.br/books?id=tilLvgAACAAJ>. Citado na página 23.

HAM, A. Flexible job shop scheduling problem for parallel batch processing machine with compatible job families. *Appl. Math. Model.*, Elsevier Inc., v. 45, p. 551–562, may 2017. ISSN 0307904X. Disponível em: <http://dx.doi.org/10.1016/j.apm.2016.12.034http://linkinghub.elsevier.com/retrieve/pii/S0307904X17300094>. Citado na página 15.

HAO, X.; GEN, M.; LIN, L.; SUER, G. A. Effective multiobjective EDA for bi-criteria stochastic job-shop scheduling problem. *J. Intell. Manuf.*, v. 28, n. 3, p. 833–845, 2017. ISSN 15728145. Disponível em: <http://link.springer.com/10.1007/s10845-014-1026-0>. Citado na página 26.

HARJUNKOSKI, I.; MARAVELIAS, C. T.; BONGERS, P.; CASTRO, P. M.; ENGELL, S.; GROSSMANN, I. E.; HOOKER, J.; MÉNDEZ, C.; SAND, G.; WASSICK, J. Scope for industrial applications of production scheduling models and solution methods. *Comput. Chem. Eng.*, Elsevier Ltd, v. 62, p. 161–193, 2014. ISSN 00981354. Disponível em: <http://dx.doi.org/10.1016/j.compchemeng.2013.12.001>. Citado 2 vezes nas páginas 14 e 15.

HOORN, J. J. van. The Current state of bounds on benchmark instances of the job-shop scheduling problem. *J. Sched.*, Springer US, v. 21, n. 1, p. 127–128, feb 2018. ISSN 1094-6136. Disponível em: <http://link.springer.com/10.1007/s10951-017-0547-8>. Citado 8 vezes nas páginas 15, 16, 44, 45, 55, 71, 74 e 77.

JAIN, S.; FOLEY, W. J. Dispatching strategies for managing uncertainties in automated manufacturing systems. *Eur. J. Oper. Res.*, Elsevier Ltd., v. 248, n. 1, p. 328–341, 2016. ISSN 03772217. Disponível em: <http://dx.doi.org/10.1016/j.ejor.2015.06.060>. Citado na página 42.

JUNG, C. F. Elaboração de projetos de pesquisa aplicados a engenharia de producao. *Faccat*, 2010. Disponível em: <http://www.jung.pro.br/moodle/index.php?www.metodologia.net.br>. Citado na página 46.

KECHADI, M.-T.; LOW, K. S.; GONCALVES, G. Recurrent neural network approach for cyclic job shop scheduling problem. *J. Manuf. Syst.*, The Society of Manufacturing Engineers, v. 32, n. 4, p. 689–699, 2013. ISSN 02786125. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0278612513000174>. Citado na página 15.

KUNDAKC, N.; KULAK, O. Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. *Comput. Ind. Eng.*, v. 96, p. 31–51, 2016. ISSN 03608352. Disponível em: <https://doi.org/10.1016/j.cie.2016.03.011>. Citado 3 vezes nas páginas 16, 45 e 47.

KURDI, M. An effective new island model genetic algorithm for job shop scheduling problem. *Comput. Oper. Res.*, Elsevier, v. 67, p. 132–142, 2016. ISSN 03050548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054815002361>. Citado na página 47.

LACHTERMACHER, G. *Pesquisa operacional na tomada de decisões*. [S.l.]: Pearson Prentice Hall, 2013. ISBN 8576050935. Citado 4 vezes nas páginas 30, 31, 32 e 33.

LAWRENCE, S. *Resource Constrained Project Scheduling. An Experimental Investigation of Heuristic Scheduling Techniques (Supplement)*. [S.l.]: Carnegie-Mellon University, 1984. Citado na página 55.

LEI, D. Fuzzy job shop scheduling problem with availability constraints. *Comput. Ind. Eng.*, Elsevier Ltd, v. 58, n. 4, p. 610–617, 2010. ISSN 03608352. Disponível em: <http://dx.doi.org/10.1016/j.cie.2010.01.002>. Citado na página 78.

LIAO, C.-J.; YOU, C.-T. An Improved Formulation for the Job-Shop Scheduling Problem. *J. Oper. Res. Soc.*, v. 43, n. 11, p. 1047–1054, nov 1992. ISSN 0160-5682. Disponível em: <http://link.springer.com/10.1057/jors.1992.162>. Citado na página 25.

LIDEN, R. *ALGORITMOS GENETICOS*. 3. ed. [S.l.]: CIENCIA MODERNA, 2012. ISBN 9788539901951. Citado 3 vezes nas páginas 40, 41 e 64.

LIU, B.; FAN, Y.; LIU, Y. A fast estimation of distribution algorithm for dynamic fuzzy flexible job-shop scheduling problem. *Comput. Ind. Eng.*, Elsevier Ltd, v. 87, p. 193–201, 2015. ISSN 03608352. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0360835215002077>. Citado na página 78.

LIU, C. H. Lot streaming for customer order scheduling problem in job shop environments. *Int. J. Comput. Integr. Manuf.*, v. 22, n. 9, p. 890–907, 2009. ISSN 0951192X. Citado na página 45.

LUSTOSA, L.; MESQUITA, M.; OLIVEIRA, R.; QUELHAS, O. *Planejamento E Controle Da Produção (Pcp) Coleção Campus-ABEPRO engenharia de produção*. [S.l.: s.n.], 2013. 376 p. ISBN 8535251901, 9788535251906. Citado 3 vezes nas páginas 19, 20 e 22.

MACCARTHY, B. L.; LIU, J. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *Manuf. Eng.*, n. 786636650, p. 37–41, 1993. Disponível em: <http://dx.doi.org/10.1080/00207549308956713>. Citado 2 vezes nas páginas 15 e 23.

MANNE, A. S. On the Job-Shop Scheduling Problem. n. February 2015, 1960. Citado na página 15.

MARCONI, M.; LAKATOS, E. *Fundamentos de metodologia científica*. [S.l.: s.n.], 2003. 310 p. ISSN 9788522457588. ISBN 8522433976. Citado na página 46.

MARTINS, P. G.; LAUGENI, F. P. *Administração da produção*. 3. ed. São Paulo: Saraiva, 2015. 561 p. ISBN 978-8502-61835-0. Citado na página 14.

MUTH, J. F.; THOMPSON, G. L.; WINTERS, P. R. *Industrial scheduling*. Englewood Cliffs, NJ: Prentice-Hall, 1963. Citado na página 55.

NAGATA, Y.; ONO, I. A Guided Local Search with Iterative Ejections of Bottleneck Operations for the Job Shop Scheduling Problem. *Comput. Oper. Res.*, Elsevier Ltd, 2017. ISSN 03050548. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0305054817302460>. Citado na página 47.

NGUYEN, V.; BAO, H. P. An Efficient Solution to the Mixed Shop Scheduling Problem Using a Modified Genetic Algorithm. *Procedia - Procedia Comput. Sci.*, The Author(s), v. 95, p. 475–482, 2016. ISSN 1877-0509. Disponível em: <http://dx.doi.org/10.1016/j.procs.2016.09.324>. Citado na página 45.

NOGUEIRA, F. Modelagem e Simulação - Modelos de Previsão. p. 1–51, 2010. Disponível em: <http://www.ufjf.br/epd042/files/2009/02/previsao1.pdf>. Citado na página 21.

OLIVEIRA, S. L. *Tratado de metodologia científica: projetos e pesquisa. TCI, TCC, Monografias, Dissertações e Teses*. [S.l.]: Pioneira, 1997. ISBN 8522100705. Citado na página 46.

PINEDO, M. L. *Scheduling*. [s.n.], 2012. 1–671 p. ISSN 0278-6125. ISBN 978-1-4614-1986-0. Disponível em: <http://link.springer.com/10.1007/978-1-4614-2361-4>. Citado 4 vezes nas páginas 23, 26, 27 e 28.

Shen, Liji; Dauzère-pérès, S. Solving the Flexible Job Shop Scheduling Problem with Sequence-Dependent Setup Times. n. 2017, p. 1–24, 2017. ISSN 03772217. Citado na página 15.

SILVA, F. O. S.; CEZARINO, W.; RATTO, J. Planejamento agregado da produção: modelagem e solução via planilha Excel e Solver. *Rev. Produção Online*, v. 9, n. 3, p. 572–599, 2009. ISSN 16761901. Disponível em: <http://producaoonline.org.br/rpo/article/view/173>. Citado na página 29.

SLACK, N. *Administração Da Produção*. 3. ed. [S.l.: s.n.], 2009. 703 p. ISBN 978-85-224-5353-5. Citado 4 vezes nas páginas 19, 20, 21 e 22.

SOBEYKO, O.; MÖNCH, L. Heuristic approaches for scheduling jobs in large-scale flexible job shops. *Comput. Oper. Res.*, Elsevier, v. 68, p. 97–109, 2016. ISSN 03050548. Disponível em: <http://dx.doi.org/10.1016/j.cor.2015.11.004>. Citado 2 vezes nas páginas 26 e 48.

STORER, R. H.; WU, S. D.; VACCARI, R. New Search Spaces for Sequencing Problems with Application to Job Shop Scheduling. *Manage. Sci.*, v. 38, n. 10, p. 1495–1509, 1992. ISSN 0025-1909. Citado na página 55.

SULLIVAN, M. *Matematica Finita - Uma Abordagem Aplicada*. 11. ed. Rio de Janeiro: [s.n.], 2013. 692 p. ISBN 978-85-216-2358-8. Citado 2 vezes nas páginas 30 e 31.

TAHA, H. a. *Pesquisa Operacional*. 8. ed. [S.l.: s.n.], 2008. 359 p. ISBN 8576051508, 9788576051503. Citado 5 vezes nas páginas 29, 33, 34, 35 e 36.

TAILLARD, E. Benchmarks for basic sheduling problems. *Eur. J. Oper. Res.*, v. 64, p. 278–285, 1993. Citado 3 vezes nas páginas 15, 55 e 60.

TUBINO, D. F. *Planejamento e controle da produção: teoria e prática*. 2. ed. São Paulo: [s.n.], 2009. 190 p. ISBN 8522456941, 9788522456949. Citado 4 vezes nas páginas 19, 20, 21 e 23.

VIDONI, M. C.; VECCHIETTI, A. R. A systemic approach to define and characterize Advanced Planning Systems (APS). *Comput. Ind. Eng.*, Elsevier Ltd, v. 90, p. 326–338, 2015. ISSN 03608352. Disponível em: <http://dx.doi.org/10.1016/j.cie.2015.10.006>. Citado 2 vezes nas páginas 43 e 44.

VIDONI, M. C.; VECCHIETTI, A. R. An intelligent agent for ERP's data structure analysis based on ANSI/ISA-95 standard. *Comput. Ind.*, Elsevier B.V., v. 73, p. 39–50, 2015. ISSN 01663615. Disponível em: <http://dx.doi.org/10.1016/j.compind.2015.07.011>. Citado 2 vezes nas páginas 42 e 44.

WAGNER, H. M. An Integer Linear Programming Model for Machine Scheduling. *Nav. Res. Logist. Q.*, v. 6, n. 2, p. 131–140, 1959. ISSN 00281441. Disponível em: <http://onlinelibrary.wiley.com/wol1/doi/10.1002/nav.3800060205/abstract>. Citado na página 25.

YAMADA, T.; NAKANO, R. Genetic Algorithms for Job-Shop Scheduling Problems. *Proc. Mod. Heuristic Decis. Support*, n. March, p. 67–81, 1997. Citado na página 45.

Yamada, T.; Nakano, R. A genetic algorithm applicable to large-scale job-shop instances. *Parallel instance solving from Nat.*, v. 2, n. January 1992, p. 10, 1992. Disponível em: <http://dblp.uni-trier.de/db/conf/ppsn/ppsn1992.html#YamadaN92>. Citado 2 vezes nas páginas 45 e 55.

YAZDANI, M.; ALETI, A.; KHALILI, S. M.; JOLAI, F. Optimizing the sum of maximum earliness and tardiness of the job shop scheduling problem. *Comput. Ind. Eng.*, v. 107, p. 12–24, 2017. ISSN 03608352. Disponível em: <http://dx.doi.org/10.1016/j.cie.2017.02.019>. Citado na página 55.

ZHANG, C.; RAO, Y.; LI, P. An effective hybrid genetic algorithm for the job shop scheduling problem. *Int. J. Adv. Manuf. Technol.*, v. 39, n. 9-10, p. 965–974, 2008. ISSN 02683768. Disponível em: <https://doi.org/10.1007/s00170-007-1354-8>. Citado 3 vezes nas páginas 45, 51 e 52.

ZHANG, G.; YANG, S.; GAO, L. A genetic algorithm and tabu search for solving flexible job shop schedules. *Proc. 2008 Int. Symp. Comput. Intell. Des. Isc. 2008*, v. 1, p. 369–372, 2008. Citado na página 45.

Apêndices

APÊNDICE A – PÁGINA *WEB*

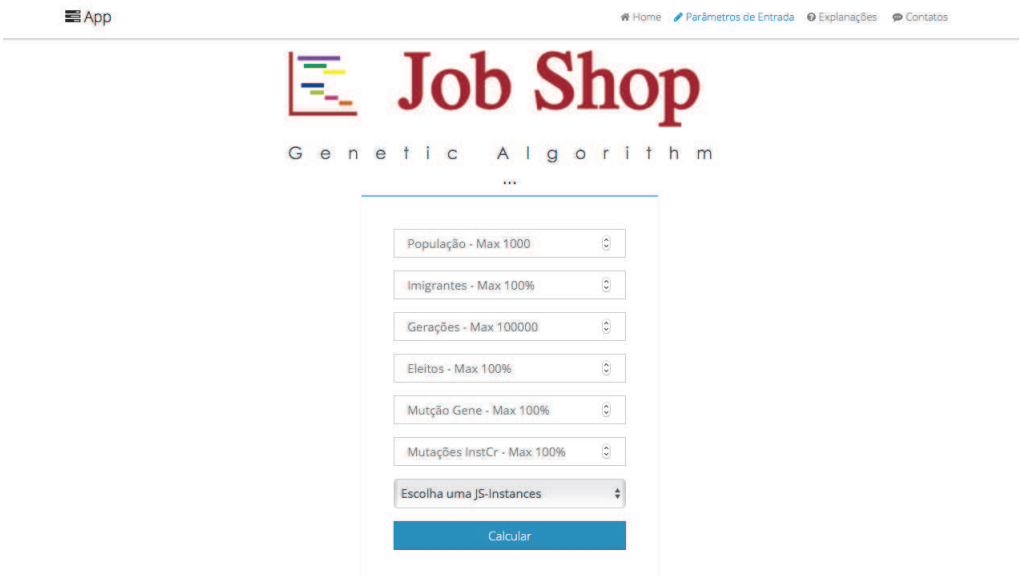
A.1 Explanações

Figura 28 – Página de Explanações

Fonte: Elaborado pelo autor (2018)

A.2 Parametrização dos Dados de Entrada

Figura 29 – Página de Parâmetros



Fonte: Elaborado pelo autor (2018)

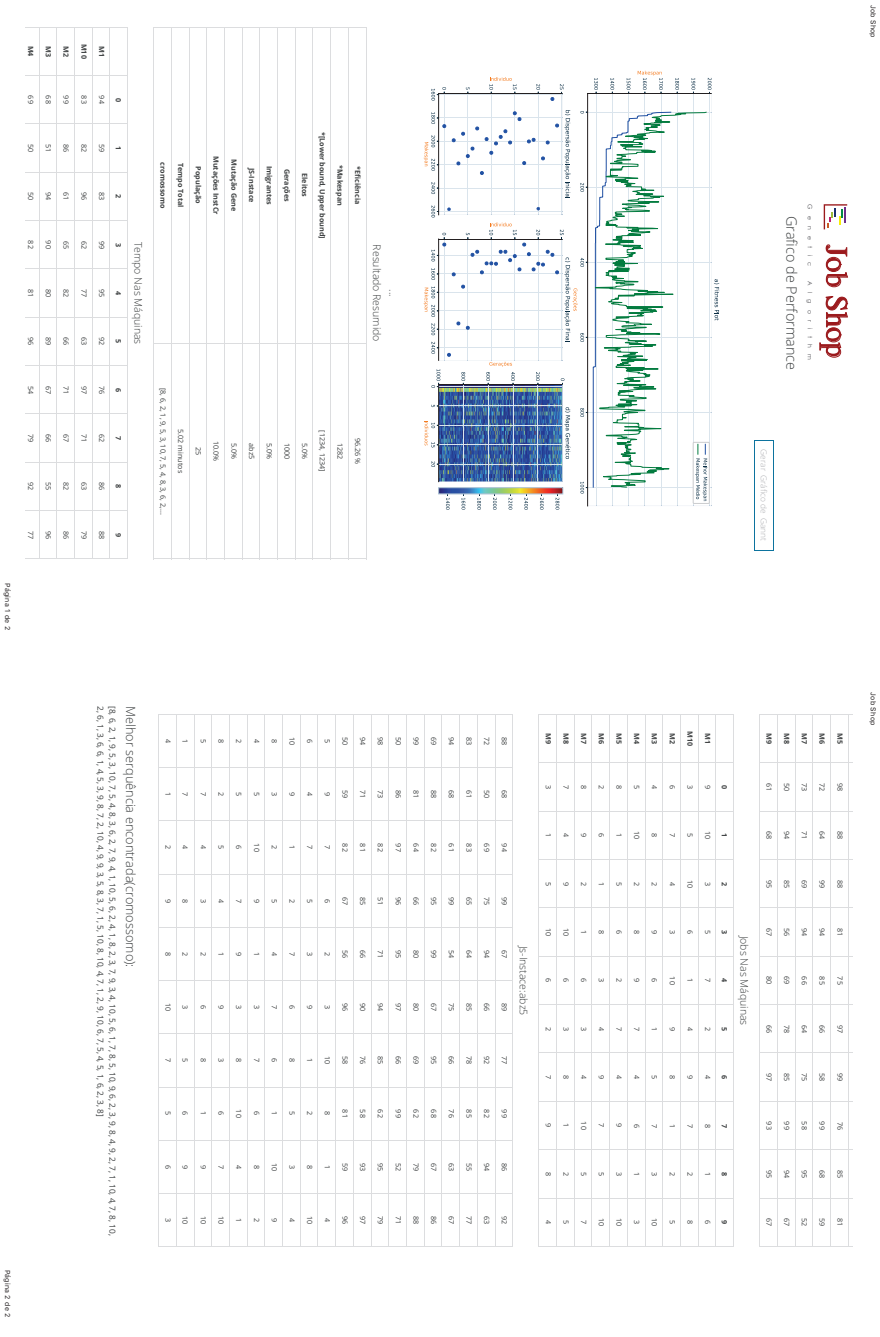
A.3 Execução do Cálculo

Figura 30 – Página de Calculo em Processo

Fonte: Elaborado pelo autor (2018)

A.4 Resultado e Desempenho do AG

Figura 31 – Página de Resultados



Fonte: Elaborado pelo autor (2018)

A.5 Resultado da Matriz Gráfica – Gráfico de Gantt

Figura 32 – Página de Resultados – Gantt

Fonte: Elaborado pelo autor (2018)