**Universidade Estadual de Maringá**
**Centro de Tecnologia**
**Departamento de Engenharia de Produção**

# Aplicação de Meta-Heurísticas para Resolução do Problemas de Sequenciamento de Máquinas Paralelas Não-Relacionadas com Tempo de Preparação Dependente da Sequência

*Everton Tozzo*

**TCC-EP-2014**

**Maringá - Paraná**
**Brasil**

Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Engenharia de Produção

# Aplicação de Meta-Heurísticas para Resolução do Problemas de Sequenciamento de Máquinas Paralelas Não-Relacionadas com Tempo de Preparação Dependente da Sequência

*Everton Tozzo*

**TCC-EP-2014**

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Produção, do Centro de Tecnologia, da Universidade Estadual de Maringá.
Orientadora: Profa. MSc. Gislaine Camila Lapasini Leal

**Maringá - Paraná**
**2014**

# AGRADECIMENTOS

Gostaria de agradecer aos meus pais, Aparecido e Elza, por me apoiarem durante toda a minha trajetória e por me inspirarem na busca de novos desafios. Agradeço pelos valores transmitidos e por estarem presentes em todos os momentos da minha vida.

À minha irmã Ana Paula e ao meu cunhado Francis, por me ajudarem sempre que necessário, pelo incentivo e pelo carinho.

À todos os meus familiares que, apesar da distância, sempre se mostraram na torcida pelo alcance dos meus objetivos.

À minha orientadora Camila pela confiança depositada, pelo conhecimento e pela experiência passada, por sua flexibilidade e apoio considerando os diferentes desafios enfrentados nessa etapa final do curso.

Ao professor Ademir, não somente por ter aceitado o convite para participar da minha banca, mas também pelo constante incentivo e participação durante o desenvolvimento de minhas atividades acadêmicas.

Aos meus professores de modo geral que, além de contribuírem para a minha formação profissional, se mostraram amigos e dispostos a ajudar sempre que necessário.

Finalmente à todos os meus amigos, junto aos quais tive a oportunidade de compartilhar essa intensa jornada, e que me ajudaram, de alguma forma, a chegar até aqui.

# RESUMO

Este trabalho propõe a avaliação de duas meta-heurísticas aplicadas na resolução do Problema de Sequenciamento de Máquinas Paralelas Não-Relacionadas com Tempo de Preparação Dependente da Sequência (PSMPNTPDS). As máquinas são ditas não-relacionadas devido a inexistência de uma relação entre o tempo de processamento das tarefas e cada uma das máquinas que tomam parte do problema. Além disso, o tempo de preparação entre a execução de duas tarefas é dependente tanto da sequência das tarefas quanto da máquina a qual está associado. A medida de desempenho utilizada para avaliar as soluções obtidas para o modelo é o tempo máximo de conclusão do sequenciamento, também conhecida como *makespan*. As meta-heurísticas algoritmos genéticos e VNS-VND foram escolhidas para resolução do problema devido à grande diferença entre suas características e peculiares métodos de resolução. Ao final, os resultados obtidos por ambas meta-heurísticas são comparados diretamente considerando o desempenho de cada uma delas ao tentar reduzir o *makespan* para o sequenciamento.

**Palavras-chave:** Máquinas Paralelas. *Makespan*. Algoritmos Genéticos. VNS-VND.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE QUADROS

# 1 INTRODUÇÃO

Segundo Macedo *et al*. (2012), ao longo dos últimos anos, o mundo vem sofrendo constantes mudanças, onde tem-se cada vez menos tempo para aprender e reagir aos novos desafios que são apresentados. Por conta disso, o julgamento e a tomada de decisão se tornaram etapas críticas do processo de gerenciamento. Muitos são os fatores e aspectos a serem considerados no momento de uma escolha, porém, de maneira antagônica, é cada vez menor o tempo para se pensar e escolher a melhor opção. Por isso, se faz necessário a aplicação de técnicas e de processos de tomada de decisão estruturados, que possam de maneira ágil responder às questões gerenciais.

Nesse contexto, a Pesquisa Operacional tem sofrido um recente e positivo impacto na administração das empresas. Com a aplicação de diversos métodos científicos, esta consolidou-se como fator diferencial presente em suas estratégias e como ferramenta auxiliar para a resolução de problemas complexos e tomadas de decisão importantes. Tanto o número quanto a variedade de suas aplicações continuam a crescer rapidamente. Algumas de suas técnicas envolvem ideias sofisticadas nos mais variados campos de aplicação, como nas ciências políticas, matemáticas, econômicas, teoria da probabilidade e estatística; razão pela qual têm sido utilizadas amplamente em todos os tipos de organizações, principalmente na indústria (DA SILVA, 2013).

A otimização combinatória é uma área da Pesquisa Operacional que se mostra de extrema importância para as empresas, pois permite a análise e otimização de casos reais, que muitas vezes requerem uma investigação mais detalhada e matematicamente complexa para a obtenção de uma solução. O objetivo dos problemas de otimização combinatória é encontrar o melhor conjunto/combinação de elementos que compõem uma solução válida. Nesses problemas, a complexidade cresce exponencialmente à medida que cresce o número de variáveis a serem otimizadas. Em alguns casos, quando há um grande número de variáveis, o tempo necessário para encontrar a melhor solução por meio de uma busca exaustiva pode ser grande o suficiente para ser considerado inviável. Técnicas de otimização são então utilizadas para resolver essa classe de problemas. Essas técnicas normalmente não garantem que a melhor solução seja encontrada, entretanto, garantem uma boa solução, considerada

satisfatória levando-se em conta as características do problema apresentadas inicialmente (VANNIMENUS & MÉZARD *apud* CONTI & ROISENBERG, 2011).

As heurísticas ou algoritmos heurísticos destacam-se como importante técnica de otimização combinatória, alvo de grande atenção acadêmica e responsável pelo desenvolvimento de considerável pesquisa nos últimos anos. Souza (2012) define heurística como uma técnica que procura boas soluções (próximas da solução ótima) a um custo computacional razoável, sem, no entanto, estar capacitada a garantir a solução ótima, bem como garantir quão próxima uma determinada solução está da solução ótima.

A grande desvantagem das heurísticas reside na dificuldade de fugir de ótimos locais, o que deu origem à outra metodologia chamada de meta-heurística, permitindo a busca em regiões mais promissoras. O desenvolvimento e a aplicação de meta-heurísticas constituem o foco deste trabalho, onde pretende-se enfatizar a importância do estudo e conhecimento das mesmas como ferramenta para o profissional da Engenharia de Produção por meio da comparação dos resultados obtidos por duas meta-heurísticas na resolução de um problema clássico da pesquisa operacional, o problema de sequenciamento de máquinas paralelas não-relacionadas com tempo de processamento dependente da sequência.

## 1.1 Justificativa

Do ponto de vista acadêmico, todos os problemas que podem ser resolvidos por algoritmos exatos são classificados pela literatura em três tipos, sendo eles: problemas de decisão, problemas de localização e problemas de otimização. Dentre os três tipos, os problemas de otimização são os que mais intrigam os pesquisadores, pois, normalmente, requerem técnicas específicas para superar as limitações computacionais. Esse tipo de problema tem impulsionado o surgimento de novas alternativas para se projetar algoritmos, como por exemplo, algoritmos aproximativos e as meta-heurísticas.

Em Engenharia de Produção, a tomada de decisão deve ser pautada em um sistema organizado, com a utilização de modelos matemáticos estruturados. O problema de sequenciamento de máquinas paralelas tem sido amplamente estudado nas últimas décadas, entretanto o caso envolvendo máquinas paralelas não-relacionadas tem sido muito menos especulado. Mais ainda, considerar o tempo de preparação dependente da sequência entre duas tarefas não era aplicável até recentemente (VALLADA & RUIZ, 2011).

Este constitui um cenário muito comum nos sistemas de produção atuais, especialmente nas indústrias têxtil, química, eletrônica, na execução de serviços e manutenção industrial. Na prática, esse tipo de problema é altamente complexo, devido não somente as suas dimensões, mas principalmente às características peculiares advindas da situação a ser resolvida (YING *et al.*, 2012).

Essa dificuldade é um incentivo para a investigação de algoritmos heurísticos que forneçam boas soluções em tempo computacional aceitável. Esse tipo de problema fomenta a investigação de novas técnicas computacionais, proporcionando o avanço dos modelos algorítmicos e comprovando a sua eficiência, principalmente no que tange a utilização das meta-heurísticas.

## 1.2 Definição e Delimitação do Problema

Este trabalho pretende investigar a importância da utilização de meta-heurísticas no contexto da Engenharia de Produção para resolução de problemas de sequenciamento de máquinas paralelas. Para isso, foi realizada uma pesquisa bibliográfica de modo a construir uma base teórica necessária para o desenvolvimento de dois algoritmos para a resolução do problema. Cada algoritmo corresponde à implementação de uma única meta-heurística. Os resultados obtidos por cada um deles foram posteriormente comparados de modo a verificar sua eficiência.

## 1.3 Objetivos

### 1.3.1 Objetivo geral

Aplicação de duas meta-heurísticas para a resolução do problema de sequenciamento de máquinas paralelas não-relacionadas com tempo de preparação dependente da sequência, considerando a redução do tempo máximo de conclusão do sequenciamento (*makespan*) como medida de desempenho.

### 1.3.2 Objetivos específicos

Como objetivos específicos, foram identificados:

- Revisar a literatura, introduzindo os principais conceitos sobre os problemas de otimização combinatória e meta-heurísticas;
- Caracterizar o problema de escalonamento de máquinas paralelas e as duas meta-heurísticas identificadas para sua resolução;
- Implementar as meta-heurísticas;
- Realizar testes computacionais;
- Comparar os resultados obtidos por ambas meta-heurísticas na resolução do mesmo problema;

## 1.4 Metodologia

Com base em Silva e Menezes (2005), quanto à natureza deste trabalho, ele é considerado uma pesquisa básica, uma vez que objetiva gerar conhecimentos novos úteis para o avanço da ciência sem aplicação prática prevista, envolvendo verdades e interesses universais. Do ponto de vista da abordagem, a pesquisa é quantitativa, ou seja, pode ser traduzida em números, opiniões e informações para classificá-las e analisá-las. O estudo é realizado por meio da aplicação de recursos e técnicas estatísticas para uma melhor análise dos resultados obtidos a partir da implementação dos algoritmos propostos.

Quanto aos objetivos, a pesquisa é exploratória, ou seja, visa proporcionar maior familiaridade com o problema de modo a torná-lo explícito ou a construir hipóteses, envolvendo levantamento bibliográfico e análise de exemplos que estimulem a compreensão. Do ponto de vista dos procedimentos técnicos, o trabalho é uma pesquisa experimental, pois determina um objeto de estudo, por meio da seleção das variáveis que seriam capazes de influenciá-lo, definindo-se as formas de controle e de observação dos efeitos que a variável produz no objeto.

A metodologia utilizada é dividida basicamente em seis etapas: investigação da literatura, caracterização do problema, delimitação das meta-heurísticas, implementação, experimentação e comparação dos algoritmos. De modo a operacionalizar esses métodos e consolidá-los com os objetivos do trabalho, foram utilizados os seguintes procedimentos:

1- Realizar um levantamento bibliográfico dos trabalhos mais recentes, por meio das mais variadas fontes de pesquisa, sobre os principais conceitos dos problemas de otimização combinatória e meta-heurísticas;

2- Apresentar os principais aspectos teóricos sobre o Problema de Sequenciamento de Máquinas Paralelas, considerando o caso específico para máquinas não-relacionadas em que o tempo de preparação é dependente da sequência;
3- Escolher duas meta-heurísticas para resolver o problema proposto. Nesse caso, devem ser levados em consideração a aplicabilidade do problema, suas características e as características das próprias meta-heurísticas;
4- Definir a estratégia de implementação para as meta-heurísticas escolhidas e implementar dois algoritmos independentes. Cada algoritmo corresponde a aplicação de uma única meta-heurística na resolução do problema proposto cujo objetivo é a redução do *makespan* do sequenciamento;
5- Realizar experimentos/testes com dados de entrada obtidos a partir da formulação de problemas fictícios, explorando-se diferentes características no que tange a quantidade de máquinas e tarefas a serem processadas;
6- Comparar e analisar os resultados obtidos por ambos algoritmos, apresentando-se as vantagens e desvantagens de cada uma das meta-heurísticas utilizadas.

## 1.5 Estrutura do Texto

Este trabalho está estruturado em cinco capítulos. O capítulo 2 discorre a revisão bibliográfica considerando-se a temática desenvolvida. São detalhados os principais conceitos relacionados à teoria da otimização e aos métodos heurísticos, dando-se especial atenção a estrutura das meta-heurísticas Algoritmo Genético e VNS. Também são apresentadas as principais características relacionadas ao Problema de Sequenciamento de Máquinas Paralelas (PSMP), e as peculiaridades decorrentes de uma das classes desse problema, com máquinas não-relacionadas cujo tempo de preparação é dependente da sequência.

No capítulo 3 são descritos os métodos propostos para resolução do problema de escalonamento das máquinas paralelas, isto é, são detalhados os principais aspectos de implementação das meta-heurísticas algoritmo genético e VNS. Considera-se também a forma de representação dos dados de entrada e da solução gerada para ambos algoritmos.

O capítulo 4 apresenta os aspectos para construção das instâncias e experimentação dos algoritmos propostos, considerando diferentes quantidades de máquina e tarefas a serem processadas. O mesmo capítulo apresenta os resultados computacionais e a análise dos dados

obtidos a partir da execução dos experimentos, comparando de forma direta o desempenho obtido por ambas meta-heurísticas. Por fim, as considerações finais são apresentadas no capítulo 5.

# 2 REVISÃO DA LITERATURA

## 2.1 O Processo de Modelagem

Para Rangel (2005), um modelo não é equivalente à realidade, mas suficiente similar para que as conclusões obtidas por meio de sua análise e/ou operação possam ser estendidas à realidade. Hillier & Lieberman (2006) complementam que os modelos, também chamados de representações ideais, são partes integrantes do dia-a-dia. Exemplos comuns são os modelos de aviões, os retratos, os globos e assim por diante. Os modelos desempenham um importante papel na ciência e no mundo dos negócios, pois são inestimáveis na abstração da essência da matéria de investigação, mostrando inter-relacionamentos e facilitando a análise do problema.

Dependendo do propósito do modelo, faz-se necessária a definição de prioridades, uma vez que os modelos podem aumentar em complexidade de acordo com a quantidade de detalhes a ele adicionados (BISSCHOP, 2000). Desse modo, em geral, ao se formular um modelo, simplificações razoáveis do sistema ou problema real precisam ser consideradas, sendo a sua validação dependente da solução do modelo ser coerente ou não com o contexto original. Um modelo deve ser suficientemente detalhado para captar os elementos essenciais do problema, entretanto suficientemente enxuto para que seja possível a aplicação dos métodos de resolução. Nesse sentido, filtrar e estruturar a informação constitui uma importante contribuição para o processo de modelagem (ARENALES *et al*., 2007).

Diversas situações podem ser estudadas de forma mais abrangente se representadas por meio de modelos que capturem seus principais elementos. A Figura 1 representa os níveis de abstração que caracterizam o desenvolvimento de um modelo. O processo de modelagem abstrai o mundo real considerado, concentrando-se nas variáveis dominantes que controlam o comportamento desse sistema. Assim, o modelo expressa de maneira tratável as relações matemáticas que representam o comportamento do mundo real considerado (TAHA, 2008).

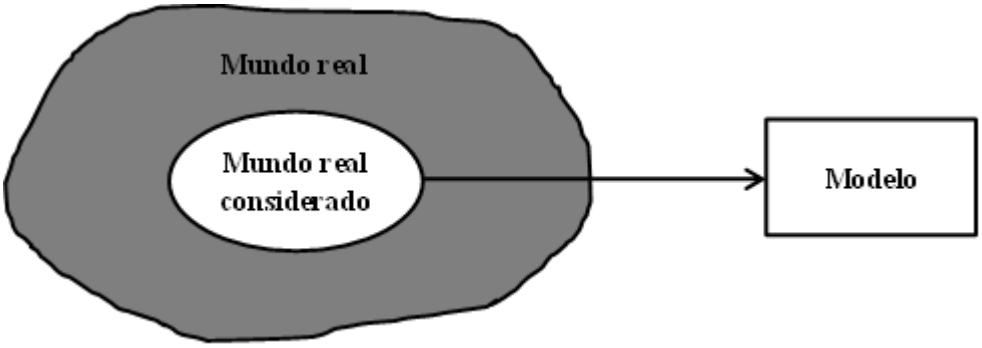


**Figura 1: Níveis de abstração no desenvolvimento de um modelo.**
**Fonte: Taha, 2008.**

São várias as razões para a construção de um modelo, dentre as quais destacam-se: análise da situação estudada, a proposta de soluções não aparentes e a experimentação de cenários simulativos impossíveis de serem alcançados na realidade ou mesmo não recomendáveis. Além disso, um modelo funciona como uma ferramenta de aprendizado. No processo de construção de modelos torna-se imprescindível a concentração em todos os seus elementos, em suas funções, em seus detalhes. Dessa forma, aprende-se a respeito da relação existente entre os elementos que o constituem, o que permite a identificação da inter-relação entre variáveis que dificilmente seriam visíveis de forma intuitiva (WILLIAMS, 2013).

Rangel (2005) apresenta que para a construção de um modelo que represente um determinado problema faz-se necessária a identificação de quais são os elementos conhecidos, geralmente associados à interpretação prévia do problema, e quais são os elementos desconhecidos, associados ao que queremos determinar ao solucionar o problema. Essa fase inicial é muitas vezes realizada de forma conjunta por meio de reuniões com o pessoal envolvido na resolução do problema e envolve o conhecimento da situação estudada. Nem sempre a identificação desses elementos é imediata, sendo a facilidade de obtenção dessas informações diretamente dependente do grau de organização da situação estudada.

Goldbarg & Luna (2000) propõem um resumo das etapas do processo de modelagem ou de construção de modelos sobre a ótica operacional, de acordo com os passos sugeridos pelo fluxograma da Figura 2.

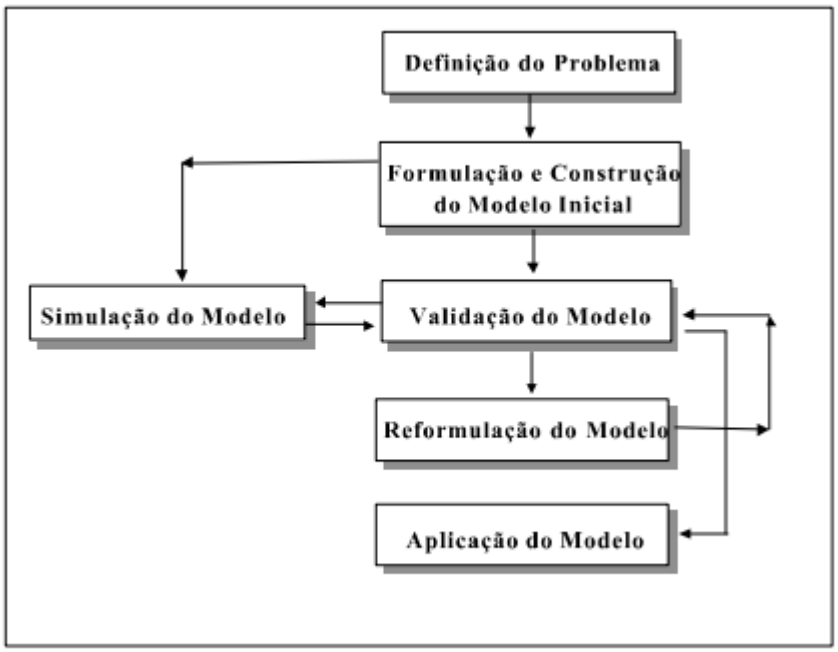


**Figura 2: O processo de construção de modelos.**
**Fonte: Goldbarg & Luna, 2000.**

Na fase de formulação do modelo de otimização são definidos os tipos de variáveis a serem utilizadas na representação, bem como o nível apropriado de agregação dessas variáveis. Devem ser representadas as restrições do problema, tanto as quantitativas quanto as de natureza lógica. O modelo deverá ser adequado a natureza dos dados de entrada e de saída, bem como ser capaz de expressar a função de desempenho, também denominada função objetivo. A formulação será complementada com o estabelecimento das hipóteses de representação que irão orientar a escolha e a possível utilização de modelos já existentes e de técnicas para a solução do problemas. Essas técnicas podem ser exatas ou heurísticas, sendo a escolha dependente da que melhor se adequa ao caso específico estudado (GOLDBARG & LUNA, 2000).

## 2.2 Teoria da Otimização

De acordo com Lopes *et al*. (2013), a Pesquisa Operacional é uma área do conhecimento fortemente interdisciplinar voltada ao desenvolvimento de modelos matemáticos e algorítmicos para a resolução de problemas reais complexos. Todos os problemas que podem ser resolvidos por algoritmos exatos são classificados pela literatura em três tipos: problemas de decisão, problemas de localização e problemas de otimização. Um problema de decisão é

aquele cuja solução é representada por uma resposta do tipo “sim” ou “não”. Em um problema de localização, procura-se identificar uma certa estrutura que satisfaça a um determinado conjunto de propriedades. No caso dos problemas de otimização, procura-se pela melhor estrutura que satisfaça a esse conjunto de propriedades.

A teoria da otimização é conhecida pelos matemáticos a séculos, entretanto a grande complexidade da maioria dos seus métodos para resolução de problemas dificultou a sua aplicação em muitas situações práticas. O desenvolvimento de computadores cada vez mais rápidos, não só tornou atrativos os velhos métodos, como também encorajou novas pesquisas nessa área (IZQUIERDO, 2000).

Os problemas de otimização envolvem a procura da melhor forma de realizar determinadas tarefas. Esses problemas são aplicados por profissionais nas mais diversas áreas de atuação como um fator diferencial para a tomada de decisão.

### 2.2.1 Problemas de otimização

De acordo com Harrel *et al*. (2000), a otimização é o processo de tentar diferentes combinações de valores para variáveis que podem ser controladas (variáveis independentes), objetivando-se a identificação de uma combinação que provê a saída mais desejada. Na maioria das vezes esse processo de tentar diferentes combinações para as variáveis se torna difícil ou mesmo impossível de ser feito em um sistema real, e por isso é feito por meio de modelos.

Um problema de otimização pode ser visto como constituinte de três elementos básicos: uma função objetivo a qual deseja-se minimizar ou maximizar, um conjunto de variáveis que afetam o valor da função objetivo e um conjunto de restrições que definem determinados valores que as variáveis podem assumir, e que formam um conjunto discreto (finito ou não) de soluções factíveis para o problema. A resposta para o problema de otimização, o ótimo global, consiste em encontrar valores para as variáveis que minimizem ou maximizem a função objetivo mantendo satisfeitas as restrições impostas (IZQUIERDO, 2000).

Na resolução de um problema de otimização, caracterizado como um problema de busca genérico, são considerados os seguintes aspectos (NIEVERGELT, 1995):

- *Espaço de soluções ou espaço de busca*: o problema de busca é formalizado pela definição de um conjunto apropriado de "estados" ou soluções que representam configurações que poderão ser encontradas durante o percurso do estado inicial do problema para a sua solução final. É um conjunto apropriado de estados que correspondem a uma dada instância de um problema. Projetar um espaço de soluções efetivo é o primeiro e principal passo para obter-se bons resultados. É importante lembrar que a construção de um espaço de soluções é uma oportunidade de conhecer os aspectos específicos de um problema e possibilita o uso eficiente de uma determinada técnica de busca. Uma solução para o problema é uma estrutura ou configuração candidata que obedece as regras estabelecidas pelas variáveis e restrições.
- *Estrutura de vizinhança*: é uma relação, usualmente incluída no espaço de soluções, que expressa restrições na forma de percorrer esse espaço, ou seja, expressa como ir de uma solução adjacente a partir de uma solução atual. Essa solução adjacente é dita vizinha da solução atual. Assim, o espaço de soluções é frequentemente modelado como um grafo com arestas direcionadas ou não. Essas arestas conectam as soluções vizinhas.
- *Meta ou critério de busca*: pode variar muito com relação a quantidade e qualidade: de uma simples solução desejada a uma enumeração exaustiva de todas as soluções do espaço. A meta faz parte do problema, no entanto, muitas vezes é realizada uma relaxação dessa meta, o que possibilita encontrar um objetivo parcial mais rapidamente. Em geral, é atribuída uma medida para cada solução do espaço e esta medida é usada como forma de avaliar as soluções estabelecendo um critério para alcançar a meta, seja de forma aproximada ou exata. Nos problemas de otimização a meta é maximizar ou minimizar uma função objetivo, sendo esta função objetivo a forma de avaliar a qualidade de cada solução.
- *Algoritmo de busca*: corresponde a uma estrutura que especifica a forma como explorar o espaço de soluções. A direção da busca pode variar de um estado inicial para um estado objetivo ou, no caso do estado objetivo ser conhecido mas o caminho até ele ser desconhecido, do estado objetivo em sentido ao estado inicial. A estratégia utilizada na resolução dos problemas de otimização depende de uma série de fatores, como finalidade teórica ou prática, tipo e abrangência da aplicação e frequência de uso da mesma.

- *Estrutura de dados*: utilizada para organizar o método de busca escolhido, de modo a memorizar quais partes do espaço de soluções foram visitadas, encontrar caminhos para regiões ainda não exploradas e armazenar as informações obtidas durante o processo de busca.

### 2.2.2 Otimização combinatória

Muitos problemas de otimização de importância tanto prática quanto teórica consistem na busca pela "melhor" configuração de um conjunto de variáveis de modo a se atingir determinado objetivo. Esses problemas dividem-se naturalmente em duas categorias: aqueles em que as soluções são codificadas com variáveis reais e aqueles cujas soluções são codificadas com variáveis discretas. Dentre os problemas nos quais as soluções são codificadas com variáveis discretas existe uma classe denominada Problemas de Otimização Combinatória. Para esses problemas, especificamente, busca-se por um objeto pertencente a um conjunto finito. Esse objeto é normalmente um número inteiro, um subconjunto, uma permutação ou uma estrutura em grafo (BLUM & ROLI, 2003).

A otimização combinatória é um importante campo da matemática aplicada, que combina técnicas da combinatória, programação linear e teoria dos algoritmos para resolver problemas de otimização em estruturas discretas. Ela desempenha um importante papel no processo de tomada de decisão, uma vez que a melhor decisão depende comumente de um combinação não trivial de vários fatores. De forma genérica, pode-se caracterizar os problemas de otimização combinatória como sendo o arranjo, agrupamento, ordenação ou seleção a partir de um conjunto discreto e finito de dados, do melhor subconjunto que satisfaça a determinados critérios pré-definidos (GOLDBARG & LUNA, 2000).

De modo formal, Brum & Roli (2003) definem um problema de otimização combinatória $P = (S,f)$ como:

- Um conjunto de variáveis $X = \{x_1,x_2,...,x_n\}$;
- Um conjunto de domínios para cada uma das variáveis $D_1,D_2,...,D_n$;
- Restrições dentre as variáveis;
- Uma fução objetivo $f$ a ser minimizada, onde $f: D_1 \text{ x } ... \text{ x } D_n \rightarrow \mathbb{R}^+$;

O conjunto de todas as possíveis atribuições factíveis é dado por:

$$S = \{s = \{(x_1,v_1),(x_2,v_2),...,(x_n,v_n)\} | v_i \in D_i, \text{ } s \text{ satisfazendo todas as restrições}\}.$$

Por terminologia, $S$ normalmente é chamado de espaço de busca ou solução, onde cada elemento do conjunto pode ser considerado uma solução candidata. Para resolver um problema de otimização combinatória deve-se encontrar uma solução $s^* \in S$ que minimiza o valor da função objetivo, isto é, $f(s^*) \leq f(s) \; \forall \; s \in S$. A solução $s^*$ é chamada de ótimo global de $(S,f)$ e o conjunto $S^* \subseteq S$ é chamado de conjunto da solução ótimo global.

Essa definição é construída objetivando-se determinar um valor mínimo para a função objetivo $f$. Entretanto, os mesmos princípios podem ser aplicados, sem perda de generalidade, para o caso em que o objetivo principal do problema consiste na maximização de resultados. Logo, maximizar o valor da função objetivo $f$ corresponde a minimizar o valor de $-f$, o que garante a completude da definição.

Mesmo a duas décadas atrás, Aardal *et al*. (1997) já consideram a otimização combinatória como uma poderosa área de interesses científicos e de grande relevância prática. De modo complementar, o desenvolvimento em outras áreas também tem estimulado a pesquisa em otimização combinatória. Por um lado, o contínuo aperfeiçoamento da capacidade computacional impulsionou a necessidade de algoritmos mais eficientes. Por outro lado, no campo de aplicações, houve um crescente confiança no potencial prático das técnicas de otimização.

De forma conjunta, como consequência do desenvolvimento dessas áreas, problemas reais complexos que até pouco tempo não podiam ser tratados agora podem ser resolvidos. Notáveis exemplos de aplicação da otimização combinatória podem ser verificados na resolução de problemas de escalonamento de tripulações aéreas, escalonamento de enfermeiros, atribuição de horários de aula em universidades, roteamento de veículos, desenvolvimento de redes de telecomunicações, entre vários outros.

## 2.3 Métodos Heurísticos

Técnicas de otimização são conhecidas a mais de um século e podem ser aplicadas em diversos campos da ciência. No entanto, estas técnicas apresentam algumas limitações, como: falta de continuidade das funções a serem otimizadas ou de suas restrições, funções não convexas, multimodalidade (vários pontos ótimos), existência de ruídos nas funções,

necessidade de se trabalhar com valores discretos para as variáveis, entre outros (SARAMAGO, 2003).

Além dessas limitações, o crescente aumento da dimensão e complexidade dos problemas reais práticos criaram um grande impacto na eficiência dos algoritmos de otimização exata para encontrar um resultado dentre um período de tempo considerado aceitável. Dessa forma, como alternativa às modelagens puramente matemáticas, têm surgido, ao longo das últimas décadas, algoritmos aproximados. Esses algoritmos são obtidos por meio dos procedimentos heurísticos e meta-heurísticos (VIANA, 2004).

Para Silva (2005), as técnicas heurísticas de otimização proporcionam soluções de boa qualidade dentro de um tempo hábil para a tomada de decisão, mas não garantem que as soluções encontradas sejam as ótimas. Assim, tais métodos, baseados na busca aleatória controlada por critérios probabilísticos, tiveram um importante desenvolvimento nos últimos anos, principalmente devido ao avanço dos recursos computacionais, uma vez que necessitam de um número elevado de avaliações da função objetivo.

### 2.3.1 Heurísticas

Segundo Neto *et al*. (2010), o nome heurística é derivado da palavra grega *heuriskein*, que significa descobrir e se vincula a suposta exclamação *eureka* de Arquimedes ao descobrir seu famoso princípio. Esse termo é usado para descrever um método que, baseado na experiência ou julgamento, parece conduzir a uma boa solução de um problema de acordo com os recursos utilizados. Esse termo também pode ser associado a um conhecimento circunstancial, não verificável matematicamente.

A ideia mais genérica associada ao termo heurístico corresponde à tentativa de resolver problemas reais utilizando, em sua essência, o conhecimento disponível a partir da natureza dos problemas. Especificamente em pesquisa operacional, o termo heurístico se aplica a um procedimento de resolução de problemas complexos para o qual se tem um alto grau de confiança e que encontra soluções de boa qualidade a um custo computacional consideravelmente baixo, mesmo sem a garantia de otimalidade e viabilidade durante sua aplicação (MELIÁN *et al*., 2003).

As heurísticas são métodos de busca que tiram proveito de características e informações do próprio problema a ser explorado, facilitando o encontro de um mínimo global no espaço de busca. O termo heurístico é utilizado em contraposição a exato, que se aplica aos procedimentos nos quais exige-se que a solução obtida seja ótima e factível. Os métodos heurísticos devem ser projetados e propostos para cada problema, utilizando toda a informação disponível e análise teórica do modelo.

As heurísticas são limitadas e fornecem sempre a mesma solução quando iniciadas de um mesmo ponto de partida. As pesquisas realizadas ao longo de décadas sobre o desempenho de métodos heurísticos e, em particular, sobre as características que conduzem ao êxito de tais métodos, levaram à elaboração de estratégias genéricas ou esqueletos de algoritmos para a construção de heurísticas. Mais tarde, essas estratégias passaram a ser chamadas de meta-heurísticas (MELIÁN *et al*., 2003).

Quando bem projetadas para um problema específico, as heurísticas frequentemente alcançam um rendimento significativamente mais alto do que as meta-heurísticas. Entretanto, as meta-heurísticas apresentam outros tipos de vantagens, como a simplicidade, adaptabilidade e robustez dos procedimentos. Sendo assim, as meta-heurísticas surgiram como uma tentativa de suprir uma deficiência das heurísticas e tem como objetivo principal explorar um espaço de pesquisa de forma inteligente, ou seja, encontrar soluções de alta qualidade movendo-se para áreas não exploradas, quando necessário (BLUM & ROLI, 2003).

### 2.3.2 Meta-heurísticas

Meta-heurística é um termo geralmente utilizado para descrever um subcampo conhecido como otimização estocástica, que corresponde a uma classe geral de algoritmos e técnicas que empregam um certo grau de aleatoriedade para a identificação da solução ótima (ou da mais ótima possível) para problemas considerados difíceis. Para Luke (2013), as meta-heurísticas são aplicadas a problemas ditos *I know it when I see it*, termo que pode ser traduzido para "eu sei quando verifico". Como justificativa, o autor complementa:

> "São algoritmos usados para encontrar respostas para problemas quando tem-se poucas informações para ajudá-lo: não se sabe de antemão qual é a aparência da solução ótima, não se sabe como encontrá-la de modo esquemático, tem-se poucas informações heurísticas para prosseguir e a utilização de pesquisa exaustiva está fora

> de cogitação, pois o espaço de busca é muito amplo. Todavia, dada uma solução candidata para o problema, pode-se testá-la e verificar quão boa ela é. Isto é, conhece-se uma solução boa quando a mesma é verificada."

As meta-heurísticas são técnicas de otimização combinatória, que não são dedicadas a solução de um problema específico. Segundo Ólafsson (2006), as meta-heurísticas são projetadas para resolver complexos problemas de otimização onde outros métodos de otimização fracassaram ao tentarem ser eficientes e/ou eficazes. Esses métodos tornaram-se reconhecidos como uma dos mais práticos modos de resolução de problemas complexos, o que é particularmente verdade para os diversos problemas reais práticos que são de caráter combinatório por natureza. As vantagens práticas das meta-heurísticas são consequências de sua efetividade e aplicabilidade genérica.

Até pouco tempo, heurísticas especializadas eram tipicamente desenvolvidas para resolver problemas complexos de otimização combinatória. Elas requeriam um novo método para cada problemas e as lições aprendidas em uma dada aplicação nem sempre eram bem generalizadas para outras classes de problemas. Dessa forma, com o desenvolvimento de estratégias de solução mais genéricas, isto é, estruturas algorítmicas gerais que podem ser aplicadas em diferentes problemas de otimização, as meta-heurísticas, o desafio principal se tornou a adaptação das meta-heurísticas para um problema específico ou para uma classe de problemas. Isso requer muito menos esforço do que o desenvolvimento de uma heurística especializada para cada aplicação, o que torna as meta-heurísticas uma escolha muito mais atraente para implementação em softwares genéricos (ÓLAFSSON, 2006).

Dentre os algoritmos classificados como meta-heurísticas, encontram-se: otimização por colônia de formigas (*ant colony optimization*), algoritmos genéticos (*genetic algorithm*), busca tabu (*tabu search*), recozimento simulado (*simulated annealing*), pesquisa em vizinhança variável (*variable neighboorhod search*), busca local iterativa (*iterated local search*), busca local guiada (*guided local search*), entre outros (CHAVES, 2009).

As meta-heurísticas são procedimentos de busca com capacidade de escapar de ótimos locais, focados na eficiência e em uma maior exploração do espaço de busca. As meta-heurísticas possuem algumas propriedades, dentre elas (BLUM & ROLI, 2003):

- São estratégias para guiar o processo de busca;

- Objetivam explorar eficientemente o espaço de busca, de modo a encontrar uma solução ótima próxima ao ponto atual;
- Estendem-se de simples procedimentos de busca local a complexos processos de aprendizagem;
- Constituem-se de algoritmos aproximados e geralmente não determinísticos.
- Incorporam mecanismos que evitam o aprisionamento em áreas restritas do espaço de busca;
- Fazem uso de um domínio específico de conhecimento na forma de uma heurística que é controlada por um nível de estratégia;
- Armazenam experiências de busca (encapsulada em memória), utilizando-as para guiar futuros processos de busca.

As meta-heurísticas são estratégias para se projetar procedimentos heurísticos. Dessa forma, os tipos de meta-heurísticas se estabelecem, em primeiro lugar, em função do tipo de procedimento a que se referem. Para Blum & Roli (2003), existem diversas formas de se classificar as meta-heurísticas, o que depende da característica selecionada para individualizar cada uma delas, como o resultado de diferentes pontos de vista. Os autores sintetizam as formas mais importantes de classificação das meta-heurísticas como:

- *Inspiradas na natureza ou não inspiradas na natureza*: talvez a forma mais intuitiva de se classificar as meta-heurísticas é baseada na origem de seus algoritmos. Existem algoritmos que são inspirados na própria natureza, em seu comportamento, no comportamento dos seres vivos e no modo de ocorrência dos fenômenos naturais, como é o caso dos Algoritmos Genéticos e da Otimização por Colônia de Formigas. Outros algoritmos não são inspirados nesse comportamento, como é o caso da Busca Tabu e da Busca Local Iterativa. Essa classificação não é muito expressiva pois muitos algoritmos híbridos atuais não se enquadram em nenhuma das duas classes. Além disso, muitas vezes, torna-se muito complicada a atribuição de um algoritmo para uma das duas classes. Por exemplo, pode-se indagar se o uso de memória na Busca Tabu é ou não inspirado na natureza;
- *Baseada em população ou busca pontual*: outra característica que pode ser baseada para classificação das meta-heurísticas é o número de soluções utilizadas ao mesmo tempo. Algoritmos que utilizam solução única são chamados de métodos de trajetória

e envolvem meta-heurísticas de busca local, como a Busca Tabu, Busca Local Iterativa e Pesquisa em Vizinhança Variável. Eles possuem a propriedade de descrever uma trajetória no espaço de busca durante sua execução. Meta-heurísticas baseadas em população, de modo contrário, executam um processo de busca que descreve a evolução de um conjunto de pontos no espaço de busca.

- *Função objetivo estática ou dinâmica*: pode-se também classificar as meta-heurísticas segundo a forma como a função objetivo é utilizada. Enquanto alguns algoritmos mantêm a função objetivo fixa na representação do problema, outros, como é o caso da Busca Local Guiada (GLS), modificam-na durante a busca. A ideia por traz desse método é escapar de mínimos locais por meio da modificação do panorama da busca. Dessa forma, a função objetivo é alterada na tentativa de incorporar informações coletadas durante o processo de busca.
- *Uma ou várias estruturas de vizinhança*: a maioria das meta-heurísticas constituem-se de apenas uma estrutura de vizinhança. Em outras palavras, o modo como é percorrido o espaço de busca não é alterado durante a execução do algoritmo. Outras meta-heurísticas, como a Pesquisa em Vizinhança Variável (VNS) utiliza um conjunto de estruturas de vizinhança que possibilitam uma diversificação na forma de exploração do espaço de busca.
- *Utilização ou não de memória*: um importante aspecto a ser considerado na classificação das meta-heurísticas é a utilização do histórico de busca, isto é, se existe a utilização de memória ou não. Os algoritmos que não utilizam memória executam um processo de Markov, uma vez que a informação utilizada para determinar a ação subsequente é o estado atual do processo de busca. Para os algoritmos que utilizam memória, esta pode ser classificada em memória de curto e memória de longo prazo. A memória de curto prazo geralmente mantém a trajetória dos movimentos recentemente executados pelo algoritmo, as soluções obtidas e as decisões tomadas. Já a memória de longo prazo é uma acumulação de parâmetros resumidos com relação a toda busca. Atualmente, a utilização de memória é reconhecida como um dos elementos fundamentais que compõem uma meta-heurística poderosa.

A estratégia de uma meta-heurística deve ser projetada com o objetivo de explorar o espaço de busca de forma eficiente e eficaz. A busca executada, de acordo com a estratégia utilizada pela meta-heurística, deve ser suficientemente hábil tanto para explorar intensivamente

regiões do espaço de busca com soluções de boa qualidade como também mover-se para regiões ainda não exploradas, quando necessário. De grande importância na aplicabilidade das meta-heurísticas, os conceitos para o alcance desses objetivos são conhecidos como intensificação e diversificação (NETO, 2010).

A intensificação, também chamada de exploração focada ou busca em profundidade, consiste em uma busca cuidadosa e intensiva ao redor de boas soluções já encontradas e de suas vizinhanças, em uma pequena região do espaço de busca. A diversificação, também conhecida por exploração diversificada ou busca em largura, por outro lado, estimula o processo de busca a examinar regiões ainda não exploradas considerando toda a extensão do espaço de busca, gerando soluções que diferenciam significantemente das encontradas anteriormente.

A intensificação e diversificação constituem duas forças para exploração do espaço de busca. Um dos desafios na aplicabilidade de uma meta-heurística é encontrar o equilíbrio ideal entre essas duas estratégias. Uma meta-heurística será bem sucedida em um dado problema de otimização combinatória se proporcionar um balanço na intensificação da experiência de busca acumulada e na diversificação do espaço de busca de modo a identificar regiões com soluções de boa qualidade (BLUM & ROLI, 2003).

### 2.3.3 Algoritmos genéticos

O Algoritmo Genético (AG) é uma meta-heurística inspirada nos mecanismos de evolução de população dos seres vivos. Esse método foi introduzido por John Holland em 1975 e popularizado por um de seus alunos, David Goldberg, em 1989. Esse algoritmo segue o princípio da seleção natural e sobrevivência do mais apto, declarado em 1859 pelo naturalista e fisiologista inglês Charles Darwin em seu livro *A Origem das Espécies*.

De acordo com Darwin “quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes”. De modo geral, a própria evolução natural implementa mecanismos adaptativos de otimização que, embora estejam longe de serem uma forma de busca aleatória, com certeza envolvem aleatoriedade. É esse tipo de busca inteligente, mas não determinística, que o AG tenta imitar (LACERDA & CARVALHO, 1999).

Os AGs são procedimentos iterativos que mantêm um grupo de soluções. Para cada iteração, um conjunto de indivíduos são selecionados, se reproduzem e sofrem mutação, gerando descendentes

com características semelhantes às de seus predecessores. Nesse contexto, indivíduos melhor adaptados tendem a reproduzir com maior frequência e a sobreviver por mais tempo. Após certo número de iterações, espera-se que a população de indivíduos convirja para uma geração mais apta ou melhor adaptada, que corresponde a uma solução ótima ou quase ótima para o problema a ser solucionado.

As aplicações dos AGs possuem um vocabulário muito específico e suas principais definições estão estreitamente associadas com termos da biologia (CASTRO, 2001):

- *Cromossomo ou indivíduo*: cadeia de caracteres representando alguma informação relativa às variáveis do problema. Cada indivíduo representa uma solução;
- *Gene*: unidade básica do indivíduo. Cada indivíduo possui um certo conjunto de genes, que descrevem as variáveis do problema;
- *Alelo*: representa os possíveis valores que um gene pode assumir. Se o gene for uma variável binária, os alelos serão os valores 0 e 1;
- *Loco*: posição de cada gene no cromossomo;
- *População*: conjunto de indivíduos ou soluções;
- *Geração*: o número de iterações que o algoritmo genético executa;
- *Aptidão*: valor da função-objetivo associada a um indivíduo.

#### 2.3.3.1 Estrutura de um algoritmo genético

Os AGs são aparentemente simples, entretanto são capazes de resolver problemas complexos de uma maneira muito elegante (REZENDE, 2005). Existe uma grande variedade de AGs, porém todos eles possuem uma estrutura em comum, conforme Quadro 1.

O AG inicializa com uma população inicial $P(0)$ de indivíduos, comumente gerados de modo aleatório. A avaliação é realizada por uma função de aptidão $f(i)$, que retorna um valor (medida de aptidão ou desempenho) para cada indivíduo. Após avaliar os candidatos, o algoritmo seleciona pares para a recombinação. Para isso, avalia a qualidade de cada solução por meio de uma comparação direta de sua medida de aptidão com a medida de aptidão dos demais indivíduos da população. Métodos de seleção são comumente probabilísticos, onde membros mais "fracos" recebem menores probabilidades de reprodução (LUGER, 2004). Diferentes métodos podem ser

aplicados para a seleção dos indivíduos, dentre os quais destacam-se o método da roleta, o método do torneio e o método da amostragem universal estocástica.

**Início;**
1. $t = 0$;
2. Gere população inicial $P(t = 0)$;
3. **Enquanto (Critério de parada não satisfeito) faça:**
4. Avalie a aptidão $f(i)$ de cada membro $i$ da população $P(t)$;
5. Selecione membros de $P(t)$;
6. Aplique operadores genéticos sobre os membros de $P(t)$;
7. Substitua indivíduos de $P(t)$ por esses descendentes;
8. $t = t + 1$;

**Fim.**

**Quadro 1: Pseudocódigo para o Algoritmo Genético.**

De acordo com Rezende (2005), após a seleção dos indivíduos, um conjunto de operadores genéticos são aplicados de modo a permitir a geração de populações sucessivas. Os principais operadores são o cruzamento (*crossover*) e a mutação. Eles são necessários para que a população se diversifique, enquanto mantem características de adaptação adquiridas pelas gerações anteriores.

O cruzamento (*crossover*) é o operador responsável pela recombinação de característica dos pais durante a reprodução, permitindo que as próximas gerações herdem essas características. Ele é considerado o operador genético predominante, sendo aplicado com probabilidade dada por uma taxa de cruzamento $P_C$ relativamente alta ($0{,}6 \leq P_C \leq 0{,}99$). De forma conjunta, o operador de mutação é necessário para a introdução e manutenção da diversidade genética da população, alterando arbitrariamente um ou mais componentes de uma estrutura escolhida, o que fornece meios para a introdução e diversificação dos novos elementos. O operador de mutação é considerado um operador secundário, sendo aplicado de acordo com uma taxa de mutação $P_M$ pequena ($0{,}001 \leq P_M \leq 0{,}1$) (REZENDE, 2005).

A partir da avaliação, seleção e aplicação dos operadores genéticos, o AG transforma a população por meio de sucessivas gerações, estendendo a busca até que um critério de parada seja satisfeito. Para prevenir que os melhores indivíduos não desapareçam da população, esses podem ser automaticamente colocados nas gerações sucessivas, o que é chamado de política elitista.

### 2.3.4 A meta-heurística VNS-VND

#### 2.3.4.1 Busca local VND

Segundo Freitas e Montané (2008), o método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*), também conhecido como VND, é uma estratégia de busca local que consiste na exploração do espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança, assumindo-se apenas a possibilidade de melhorias na solução inicial. Essas vizinhanças diversificadas procuram por soluções "mais distantes" da solução atual na tentativa de escapar de ótimos locais.

O método explora sequencialmente as vizinhanças e, para cada uma delas, procura pelo melhor vizinho. Quando o vizinho não apresentar melhoria com relação à solução corrente, passa-se para a vizinhança subsequente. Em caso de melhora nessa última situação, retorna-se novamente à primeira vizinhança da sequência. O algoritmo é interrompido quando não encontra melhoria em nenhuma das vizinhanças pesquisadas (CHAVES *et al*., 2007).

Dado um conjunto $r$ de estruturas de vizinhança $N_k$ ($k$ = 0, 1, ... , $r$ - 1), seja $s_0$ uma solução inicial. O procedimento da meta-heurística VND pode ser representado conforme Quadro 2.

| | |
|---|---|
| **Início;** | |
| 1. $s' = s_0$; | // Solução corrente |
| 2. $k = 0$; | // Tipo de estrutura de vizinhança |
| 3. **Enquanto ($k < r$) faça:** | |
| 4. Encontre o melhor vizinho s" $\in N_k$ a partir de *s'*; | |
| 5. **Se (custo da solução *s''* < custo da solução *s'*):** | |
| 6. Então *s'* = *s''*; $k$ = 0; | // Primeira vizinhança |
| 7. Senão $k = k + 1$; | // Próxima vizinhança |
| 8. Retorne *s'*; | |
| **Fim.** | |

**Quadro 2: Pseudocódigo para o procedimento VND.**

Na linha 4 do procedimento apresentado no Quadro 2, para encontrar o vizinho *s''* a partir de *s'*, pode-se aplicar um procedimento de busca local qualquer para se obter a melhor solução da estrutura de vizinhança que contém a solução *s'*.

#### 2.3.4.2 A meta-heurística VNS

O método *Variable Neighborhood Search* (VNS), também conhecido como Pesquisa em Vizinhança Variável, foi proposto por Hansen e Mladenovic em 1997. O VNS consiste em uma meta-heurística para resolver problemas de otimização global e combinatorial cuja ideia básica é a troca sistemática da vizinhança dentro de uma busca local. Segundo Hansen & Mladenovic (2009), o VNS possui alguns princípios básicos, dentre eles:

1- Um ótimo local com relação a uma vizinhança não necessariamente corresponde a um ótimo local com relação à outra vizinhança;
2- Um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança;
3- Para muitos problemas, os ótimos locais com relação a uma vizinhança são relativamente próximos.

A última observação implica que um ótimo local sempre fornece algum tipo de informação sobre o ótimo global. Sendo assim, um estudo sobre a vizinhança desse ótimo local pode ser realizado até que um provável melhor resultado seja encontrado.

O VNS não segue uma trajetória, mas sim explora vizinhanças gradativamente mais "distantes" da solução corrente e focaliza a busca em torno dessas vizinhanças. Este processo funciona como um mecanismo de diversificação da busca. Após o processo de perturbação da solução é executado um processo de busca local que é tipicamente um mecanismo de intensificação. Sempre que a busca local não consegue melhorar a solução corrente altera-se o mecanismo de perturbação utilizando-se outra estrutura de vizinhança. Pode-se considerar que a meta-heurística VNS é uma extensão da meta-heurística VND, mas admite soluções que pioram a solução corrente, o que permite escapar de ótimos locais (FREITAS & MONTANÉ, 2008).

Neste trabalho utiliza-se uma variante da meta-heurística VNS, conhecida também como meta-heurística VNS-VND, na qual o procedimento utilizado como busca local é o próprio VND. Seja $s_0$ uma solução inicial, dado um conjunto $r$ de estruturas de vizinhança $N_k$ ($k$ = 0, 1,..., $r$ - 1), o pseudocódigo para o método VNS-VND é apresentado no Quadro 3.

**Início;**
1. $s^* = s_0$; // Solução corrente
2. **Enquanto (Critério de parada não satisfeito) faça:**
3. $k = 0$; //Tipo de estrutura de vizinhança
4. **Enquanto ($k < r$) faça:**
5. Gere um vizinho qualquer s' $\in N_k$ a partir de $s^*$;
6. $s'' = VND(s')$; // Procedimento de busca local
7. **Se (custo da solução $s''$ < custo da solução $s^*$):**
8. Então $s^* = s''$; $k = 0$;
9. Senão $k = k + 1$;
10. Retorne $s^*$;

**Fim.**

**Quadro 3: Pseudocódigo para o procedimento VNS-VND.**

A solução *s'* gerada na linha 5 é um vizinho qualquer da estrutura $N_k$, ou seja, a solução s' não é necessariamente melhor que a solução corrente s*. Essa etapa do VNS realiza uma diversificação na busca e, com isso, aumenta a probabilidade de se escapar de ótimos locais.

## 2.4 Problema de Sequenciamento em Máquinas Paralelas

O Problema de Sequenciamento de Máquinas Paralelas (PSMP), do termo em inglês *Parallel Machine Scheduling Problem*, foi introduzido por McNaughton em 1959 e se tornou um problema extensivamente estudado devido à sua complexidade teórica e grande aplicabilidade em sistemas complexos de produção (MORAIS, 2008).

Um sistema de produção com máquinas paralelas é caracterizado pela disponibilidade de um conjunto $M = \{1, ..., m\}$ de $m$ máquinas, idênticas ou não, para a execução de um conjunto $N = \{1, ..., n\}$ de $n$ tarefas em um único estágio de produção. O objetivo do problema é determinar qual é a melhor sequenciamento para as tarefas, atentando-se não somente na identificação do conjunto de tarefas que devem ser alocadas em cada máquina, mas também na ordem com que essas tarefas serão distribuídas de modo a otimizar um determinado critério de desempenho (HADDAD, 2008). A Figura 3 representa um sistema genérico para o PSMP.

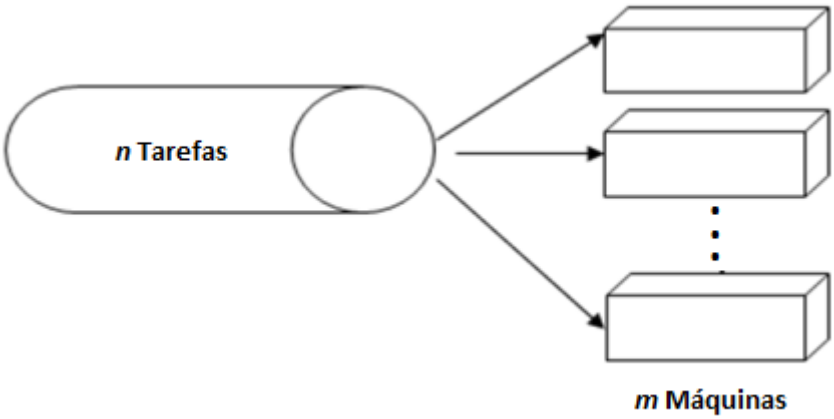


**Figura 3: Representação genérica do PSMP.**
**Fonte: Cantiere & Boiko, 2011.**

O PSMP pode ser classificado de acordo com a relação entre a velocidade de processamento das tarefas para cada uma das máquinas paralelas. Senthilkumar & Narayanan (2010) classificam o problema em três categorias distintas. Seja $p_{ij}$ o tempo de processamento definido para uma tarefa *j* (*j = 1, ... , n*) ser processada na máquina *i* (*i = 1, ..., m*), então:

1- Se $p_{ij} = p_{1j}$ para $\forall\, i,j$, tem-se o chamado Problema de Sequenciamento de Máquinas Paralelas Idênticas. Isso significa que todas as máquinas paralelas são idênticas em termos de velocidade e, nesse caso, toda tarefa terá o mesmo tempo de processamento para cada uma delas;
2- Se $p_{ij} = p_{1j}/s_i$ para $\forall\, i,j$, onde $s_i$ corresponde à velocidade de processamento da máquina $i$, tem-se o chamado Problema de Sequenciamento de Máquinas Paralelas Uniforme. Para esse problema, todas as máquinas possuem velocidades que diferem de forma constante. Normalmente assume-se $s_1, s_2, \dots, s_m$ conforme a relação $s_1 < s_2 < \cdots < s_m$, isto é, determina-se a máquina 1 como a mais devagar, a máquina $m$ como a mais rápida e estabelece-se uma relação de proporcionalidade $1/s_1 : 1/s_2 : \dots : 1/s_m$ para o processamento das tarefas;
3- Se $p_{ij}$ é arbitrário para $\forall\, i,j$, tem-se o chamado Problema de Sequenciamento de Máquinas Paralelas Não-Relacionadas. Nesse caso, não existe relação entre o tempo de processamento das tarefas para nenhuma das máquinas, o que pode ser uma consequência

das diferenças tecnológicas entre as máquinas, de características inerentes das tarefas, entre outros.

Diferentes características podem ser associadas às tarefas em processamento, tais como data de disponibilidade, tempo de preparação (*setup*), preempção, restrições de precedência, quebras de máquinas, restrições de elegibilidade, permutações, bloqueios, recirculações, entre outros. Essas características, somadas as peculiaridades advindas das três categorias do PSMP, são utilizadas para a criação de distintas classes do problema (PAULA, 2008).

Considerando-se a modelagem do PSMP, suas características e seus objetivos iniciais, são aplicadas diferentes medidas de desempenho para avaliação. Senthilkumar & Narayanan (2010) identificam que as principais medidas de desempenho incluem o tempo máximo de conclusão, também chamado de *makespan*, o tempo médio de fluxo, o atraso máximo para as tarefas, o total de atraso e a quantidade de tarefas em atraso.

O sequenciamento de máquinas paralelas é importante porque muitos sistemas reais envolvendo grupos de trabalho possuem mais de uma máquina. Além disso, esse problema enquadra-se no contexto de subproblemas de maior complexidade, tais como *flow shop* ou *job shop*, onde conjuntos de estações de trabalho contendo máquinas paralelas trabalham simultaneamente.

### 2.4.1 Problema de sequenciamento de máquinas paralelas não-relacionadas com tempo de preparação dependente da sequência

O Problema de Sequenciamento de Máquinas Paralelas Não-Relacionadas com Tempo de Preparação Dependente da Sequência (PSMPNTDS), do termo em inglês *Unrelated Parallel Machine Scheduling Problem With Sequence Dependent Setup Times*, constitui uma das classes de problemas originados do PSMP. Para tal, tem-se um conjunto $N = \{1, ..., n\}$ de $n$ tarefas e um conjunto $M = \{1, ..., m\}$ de $m$ máquinas não-relacionadas ($m < n$), com as seguintes características (HADDAD, 2012):

1- Cada tarefa deve ser processada exatamente uma vez por apenas uma máquina;
2- Cada tarefa $j$ possui um tempo de processamento $p_{ij}$ que depende da máquina $i$ na qual será alocada. É devido à essa característica que as máquinas são ditas não-relacionadas;

3- O tempo de preparação é dependente tanto da sequência das tarefas quanto da máquina associada. Seja $S_{ijk}$ o tempo de preparação da máquina *i* entre o processamento das tarefas *j* e *k*, nesta ordem. Então, o tempo de preparação da máquina *i* entre as tarefas *j* e *k* é diferente do tempo de preparação da máquina *i* entre as tarefas *k* e *j*, isto é, $S_{ijk} \neq S_{ikj}$. Além disso, o tempo de preparação entre as tarefas *j* e *k* na máquina *i* é diferente do tempo de preparação entre as tarefas *j* e *k* na máquina *h*, logo $S_{ijk} \neq S_{hjk}$.

Quando *n* tarefas são sequenciadas em *m* máquinas paralelas, cada máquina *i* terá um tempo diferente $C_i$ para finalizar o processamento de todas as tarefas a ela associadas. O tempo máximo de conclusão entre todas as máquinas é conhecido como o *makespan*, sendo representado por $C_{max} = max_{0 \leq i \leq m}\{C_i\}$. O *makespan* é o tempo de conclusão do sequenciamento, que será consumido pela máquina que concluir suas tarefas por último, também chamada de máquina gargalo (VALLADA & RUIZ, 2011).

O Problema de Sequenciamento de Máquinas Paralelas Não-Relacionadas com Tempo de Preparação Dependente da Sequência, para o qual o objetivo é a redução do tempo máximo de conclusão do sequenciamento, pode ser denotado por $R|S_{ijk}|C_{max}$ (PINEDO, 2008). Nessa representação $R$ indica as máquinas não-relacionadas, $S_{ijk}$ o tempo de preparação (que depende da máquina e da ordem de execução das tarefas) e $C_{max}$ o *makespan*.

# 3 ALGORITMOS PROPOSTOS

Para os algoritmos propostos para resolução do Problema de Sequenciamento de Máquinas Paralelas Não-Relacionadas com Tempo de Preparação Dependente da Sequência, conforme objetivos desse trabalho, optou-se pela utilização de duas meta-heurísticas: algoritmos genéticos e VNS-VND.

Essas meta-heurísticas foram escolhidas devido à grande diferença existente entre suas características. Enquanto o algoritmo genético é classificado como uma meta-heurística inspirada na natureza, baseada em população e que utiliza apenas uma estrutura de vizinhança, a meta-heurística VNS apresenta características completamente opostas, visto que não é inspirada na natureza e realiza uma busca pontual por meio de várias estruturas de vizinhança. Essas peculiaridades foram tomadas como estratégia pois possibilitam uma completa diversificação no método de resolução para um mesmo problema. Ao mesmo tempo, objetivando-se uma comparação justa e independente das diferenças existentes entre ambas meta-heurísticas, foram utilizadas estratégias aproximativas na forma de execução dos algoritmos e análise dos resultados por eles obtidos.

## 3.1 Representação do Problema

O Problema de Sequenciamento de Máquinas Paralelas Não-Relacionadas com Tempo de Preparação Dependente da Sequência possui os seguintes dados de entrada: a quantidade de tarefas, a quantidade de máquinas paralelas, o tempo de processamento das tarefas para cada máquina e o tempo de *setup* das máquinas entre a execução de duas tarefas consecutivas. Dado um conjunto de *m* máquinas paralelas e *n* tarefas, então o tempo de processamento de cada tarefa $p_{ij}$ pode ser representado por uma matriz quadrada $P = [p_{ij}]$ de dimensões $m \times n$. As linhas da matriz $P$ representam as máquinas e suas colunas correspondem a todas as tarefas a serem sequenciadas. O tempo de *setup* $s_{ijk}$, ou de preparação da máquina $i$ entre o processamento das tarefas $j$ e $k$, nesta ordem, pode ser representado por uma matriz cúbica $S = [s_{ijk}]$ de ordem $m \times n \times n$.

A Figura 4 apresenta a matriz $P = [p_{ij}]$ para o sequenciamento de oito tarefas ($n = 8$) em duas máquinas pararelas não-relacionadas ($m = 2$). O tempo de processamento estabelecido são todos números inteiros gerados por uma distribuição uniforme $U[10,30]$.

| | | Tarefa | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| Máquina | **1** | 29 | 26 | 10 | 25 | 13 | 28 | 25 | 29 |
| | **2** | 11 | 10 | 20 | 20 | 14 | 29 | 15 | 13 |

**Figura 4: Matriz tempo de processamento.**

Conforme pode ser verificado na Figura 4, o tempo de processamento de cada tarefa depende diretamente da máquina para a qual será associada. Por exemplo, o tempo de processamento da tarefa 7 para a máquina 1 ($p_{17} = 25$) é diferente do tempo de processamento da mesma tarefa para a máquina 2 ($p_{27} = 15$).

Ainda considerando o problema de sequenciamento com $m = 2$ e $n = 8$, pode-se exemplificar a representação dos dados da matriz cúbica dos tempos de preparação $S = [s_{ijk}]$ desmembrando-a em duas matrizes quadradas $S_1 = [s_{1jk}]$ e $S_2 = [s_{2jk}]$, ambas de ordem $n \times n$. Nesse caso, a matriz $S_1$ corresponde ao tempo de preparação da máquina 1 entre o processamento das tarefa $j$ e $k$, nessa ordem, e a matriz $S_2$ corresponde ao tempo de preparação entre o processamento das tarefa $j$ e $k$ para a máquina 2. Ambas matrizes são apresentadas na Figura 5, sendo o tempo de setup obtido por meio de uma distribuição inteira e uniforme U[1,10].

Para as matrizes da Figura 5, as linhas representam a tarefa que acabou de ser processada (tarefa j), enquanto as colunas representam a tarefa a ser processada na sequência (tarefa $k$). Os demais valores das matrizes representam o tempo necessário para o ajuste da máquina, considerando a sua configuração inicial e a próxima tarefa. Verifica-se que o tempo de *setup* para dois pares de tarefas é dependente da máquina associada. Por exemplo, o tempo de preparação entre as tarefas 4 e 3 para a máquina 1 ($s_{143} = 10$) é diferente do tempo de preparação entre as tarefas 4 e 3 para a máquina 2 ($s_{243} = 3$).

Tarefa *k*

| Tarefa *j* / $S_1$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 8 | 1 | 1 | 5 | 9 | 5 | 5 | 6 |
| 2 | 6 | 5 | 6 | 7 | 7 | 6 | 4 | 6 |
| 3 | 2 | 7 | 4 | 6 | 2 | 1 | 6 | 9 |
| 4 | 4 | 3 | 10 | 10 | 10 | 7 | 7 | 8 |
| 5 | 7 | 4 | 10 | 2 | 9 | 7 | 8 | 4 |
| 6 | 4 | 5 | 4 | 9 | 3 | 1 | 5 | 8 |
| 7 | 10 | 10 | 10 | 6 | 2 | 2 | 1 | 4 |
| 8 | 9 | 2 | 6 | 2 | 3 | 10 | 7 | 4 |

Tarefa k

| Tarefa *j* / $S_2$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 3 | 2 | 8 | 3 | 6 | 5 | 9 |
| 2 | 3 | 10 | 2 | 1 | 2 | 2 | 7 | 10 |
| 3 | 1 | 3 | 2 | 1 | 2 | 9 | 4 | 1 |
| 4 | 3 | 10 | 3 | 8 | 8 | 5 | 10 | 1 |
| 5 | 6 | 8 | 10 | 6 | 4 | 2 | 2 | 7 |
| 6 | 5 | 7 | 3 | 10 | 5 | 5 | 7 | 6 |
| 7 | 5 | 10 | 10 | 3 | 9 | 3 | 8 | 10 |
| 8 | 5 | 4 | 2 | 8 | 4 | 4 | 8 | 3 |

**Figura 5: Matriz tempo de preparação desmembrada.**

A partir dos dados de entrada do problema de sequenciamento de máquinas paralelas não-relacionadas com tempo de preparação dependente da sequência, uma solução pode ser gerada por meio da distribuição de todas as tarefas nas máquinas existentes, desde que o conjunto das tarefas atribuídas para uma mesma máquina tenha uma ordem de processamento. A Figura 6 ilustra um possível sequenciamento que considera os dados de entrada apresentados anteriormente. Nela, o tempo de preparação de cada máquina é representado por um retângulo branco entre linhas pontilhadas.

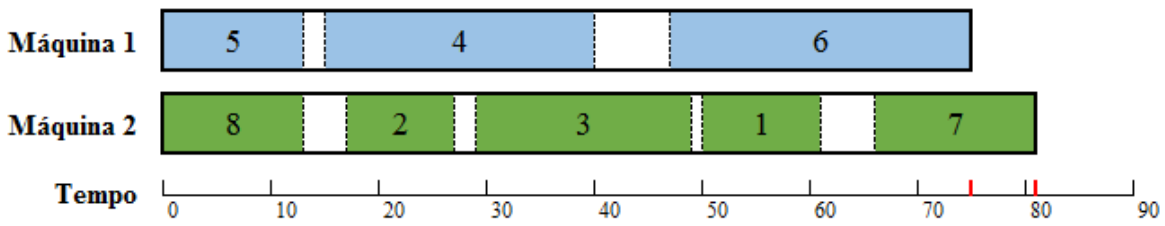


**Figura 6: Sequenciamento gerado com base nos dados de entrada.**

O tempo $C_1$ para a máquina 1 processar todas as suas tarefas, conforme verificado na Figura 6, pode ser calculado da seguinte forma:

$$C_1 = p_{15} + s_{154} + p_{14} + s_{146} + p_{16} = 13 + 2 + 25 + 7 + 28 = 75.$$

De modo semelhante, o tempo consumido para a máquina 2 processar todas as suas tarefas é $C_2 = 81$. Isso implica que a máquina 2 é a gargalo do processo, pois determina o tempo máximo para a conclusão do sequenciamento. Conclui-se então que o *makespan* obtido para a Figura 6 é igual a 81.

Para a representação computacional de uma solução do problema para os algoritmos implementados, optou-se pela utilização de uma matriz $E = [e_{ij}]$ de dimensões $m \times n$. Na matriz $E$, as linhas representam cada uma das $m$ máquinas paralelas. As tarefas de cada máquina são inseridas sequencialmente nas colunas da matriz seguindo a ordem de processamento estabelecida. Quando uma máquina possui $t$ tarefas ($t < n$), todas as colunas maiores que ($n - t$) para essa máquina são preenchidas com números 0 (zero), indicando a ausência de tarefas escalonadas.

Considerando a representação computacional adotada, o sequenciamento obtido na Figura 6 pode ser representado conforme a matriz da Figura 7.

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Máquina | 1 | 5 | 4 | 6 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 8 | 2 | 3 | 1 | 7 | 0 | 0 | 0 |

**Figura 7: Representação computacional para a solução obtida na Figura 6.**

Como metodologia alternativa, as soluções para o problema de sequenciamento também poderia ser representadas por um conjunto de $m$ listas independentes. Nesse caso, cada lista corresponde a uma máquina paralela cujos nós representam as tarefas a ela associadas.

## 3.2 Algoritmo Genético

As características do Problema de Sequenciamento de Máquinas Paralelas (PSMP) podem ser relacionadas ao vocabulário específico para aplicação do Algoritmo Genético (AG). Nesse problema, um indivíduo corresponde ao sequenciamento e distribuição de todas as tarefas nas máquinas existentes. Seus genes são cada uma de suas tarefas, que podem assumir valores

pertencentes ao conjunto $N = \{1, ..., n\}$. O valor da função de aptidão é representado pelo próprio valor do *makespan* de cada um dos indivíduos.

Todos os parâmetros adotados para o algoritmo genético foram obtidos por meio da realização de experimentos empíricos. Dentre esses parâmetros incluem-se: tamanho da população, a quantidade de pares de indivíduos selecionados em cada geração e a porcentagem de ocorrência de cada um dos operadores genéticos e dos métodos de busca local. Os valores estabelecidos para cada um desses parâmetros serão apresentados nos próximos tópicos dessa seção, sendo adotados aqueles cujos resultados obtidos foram os melhores para os experimentos empíricos executados.

Para a implementação do AG, considerou-se as etapas apresentadas na estrutura do fluxograma da Figura 8. Todas essas etapas são apresentadas com maior detalhamento nos próximos itens dessa seção.

### 3.2.1 Inicialização da população

É muito comum a geração de indivíduos aleatórios para constituir a população inicial de um AG, entretanto, incluir indivíduos considerados relativamente bons na população inicial pode parecer uma boa estratégia na tentativa de melhorar o resultado final a ser obtido pelo algoritmo.

Dessa forma, uma heurística construtiva foi aplicada na criação dos indivíduos pertencentes à população inicial do algoritmo proposto, de modo que a população fosse otimizada. Nessa heurística, para cada indivíduo, o conjunto de tarefas *n* a ser processadas nas máquinas paralelas são ordenadas aleatoriamente. Sequencialmente, cada uma das tarefas é testada em todas as posições de todas as máquinas (o que inclui as posições entre as tarefas já inseridas em uma mesma máquina) e são alocadas na posição que resultar no menor *makespan* para o escalonamento.

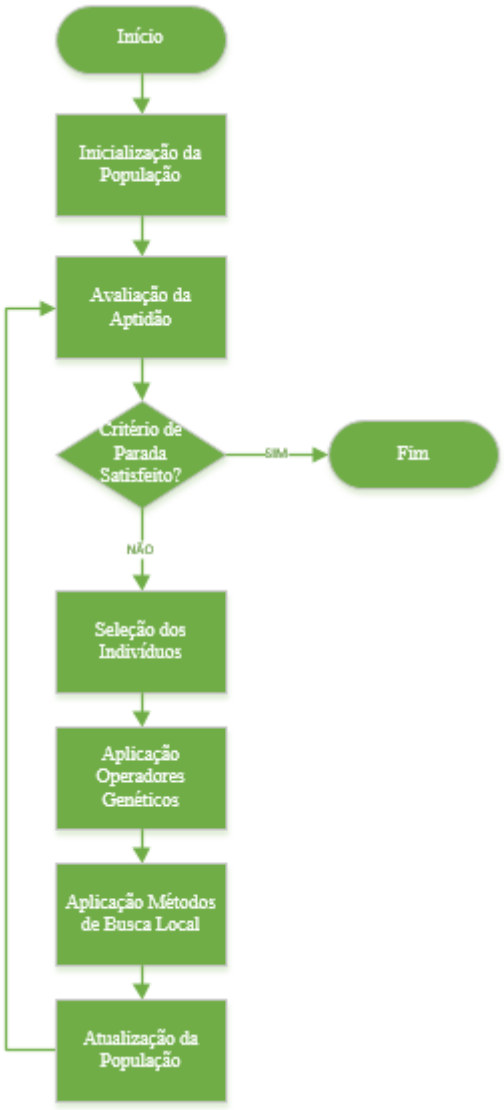


**Figura 8: Estrutura do algoritmo genético proposto.**

De acordo com essa metodologia, inicialmente nenhuma tarefa terá sido atribuída para as *m* máquinas paralelas, até que uma a uma cada tarefa será incluída na posição aparentemente mais favorável para o escalonamento, considerando o cenário resultante naquele momento.

Esse procedimento é realizado para cada indivíduo até que toda a população inicial seja construída de acordo com o tamanho da população do AG. O tamanho da população definido para o AG proposto é equivalente a 50 indivíduos.

### 3.2.2 Avaliação da aptidão e seleção dos indivíduos

Para a seleção dos indivíduos, optou-se pela utilização do método da roleta. Nesse método, para cada indivíduo da população é associada uma probabilidade de seleção diretamente proporcional a sua aptidão. O nome do método relaciona a probabilidade de seleção de um indivíduo com o tamanho da fatia de uma roleta que posteriormente seria girada para sorteio. Assim, indivíduos com melhor aptidão teriam uma porção maior da roleta, enquanto indivíduos com pior aptidão receberiam uma menor fatia da mesma.

Como citado anteriormente, no Problema de Sequenciamento de Máquinas Paralelas (PSMP), um dos métodos adotados para avaliação da aptidão de cada um dos indivíduos corresponde ao seu respectivo *makespan*. Este trabalho propõe a redução do *makespan* para uma das classes de problemas do PSMP, o que significa que os indivíduos com melhor aptidão serão aqueles que possuírem os menores valores de *makespan* dentre os demais. Desse modo, as "maiores fatias da roleta" deverão ser atribuídas aos indivíduos com menor valor associado. Para normalizar esse problema, seja $mak(i)$ a função aptidão que calcula o *makespan* de um indivíduo $i$ e seja $p$ o tamanho da população do AG. Então, a função $f(i)$ que determina a probabilidade de seleção de um indivíduo pode ser calculada de acordo com a Equação 1.

$$f(i) = \frac{{}^{1}/_{mak(i)}}{\sum_{j=1}^{p}\left({}^{1}/_{mak(j)}\right)} \qquad (1)$$

Obtida a probabilidade $f(i)$ para todos os indivíduos da população, o método da roleta pode ser aplicado normalmente, pois agora todos os indivíduos terão probabilidade de seleção inversamente proporcional ao valor de seu *makespan*. Para o AG proposto são selecionados 10 pares de indivíduos para cada geração em execução.

### 3.2.3 Operadores genéticos

Uma vez aplicado o processo de avaliação da aptidão e seleção de dois indivíduos, considerados indivíduos pais, inicia-se a etapa de aplicação dos operadores genéticos. O primeiro utilizado é o operador de cruzamento, também chamado de *crossover*. Esse operador é aplicado com uma probabilidade dada por uma taxa de cruzamento de 100% e foi baseado no modelo proposto por Vallada & Ruiz (2011).

Segundo os autores, o objetivo do operador de cruzamento é gerar dois bons indivíduos, também chamados de filhos, provenientes de ambos progenitores selecionados. Um dos operadores de cruzamento mais utilizados é o Cruzamento de Ordenação em Ponto Único (do termo em inglês *One Point Order Crossover*) adaptado para o caso das máquinas paralelas. Nesse cruzamento, para cada máquina um ponto *p* é aleatoriamente selecionado no pai 1. Toda tarefa localizada da primeira posição até a posição *p* são copiadas para a mesma máquina do filho 1. Já as tarefas localizadas da posição *p + 1* até a última posição são copiadas para a mesma máquina do filho 2. Na sequência, todas as tarefas do pai 2 que ainda não pertencem a um determinado filho são nele sequenciadas. Nesse processo, a máquina para qual determinada tarefa será associada no indivíduo filho respeitará a mesma máquina pertencente ao indivíduo pai. Além disso, as tarefas do pai 2 são inseridas na posição que resulta no menor tempo de processamento para a máquina pertencente ao indivíduo filho (o que pode reduzir o valor do *makespan* para o indivíduo, otimizando o processo).

A Figura 9 apresenta um exemplo do procedimento descrito para o operador de cruzamento considerando-se duas máquinas paralelas e oito tarefas. Para facilitar o entendimento, o tempo de preparação entre duas tarefas não é apresentado e todas as tarefas são representadas com o mesmo tamanho (o que não significa que o tempo de processamento das tarefas seja igual).

Ambos os pais são selecionados e um ponto *p* para cada máquina é aleatoriamente atribuído para o pai 1. Nesse caso $p_1 = 1$ (máquina 1) e $p_2 = 3$ (máquina 2). Então, o filho 1 é formado com as tarefas pertencentes ao pai 1 na posição 1 (tarefa 5) para a máquina 1 e com as tarefas da posição 1 até 3 (8, 2 e 3) na máquina 2. Então o filho 2 é formado com as tarefas pertencentes ao pai 1 das posições 2 e 3 (4 e 6) na máquina 1 e com as tarefas da posição 4 e 5 (1 e 7) para a máquina 2. Na Figura 9 b) verifica-se que as tarefas pertencentes ao pai 2 que ainda não foram atribuídas a um dos filhos são inseridas em suas respectivas máquinas. Para o filho 1, as tarefas 7, 4 e 6 são inseridas na máquina 1 e a tarefa 1 na máquina 2. Para o filho 2, as tarefas 2, 3 e 8 são inseridas na máquina 1 e a tarefa 5 é inserida na máquina 2. A Figura 9 c) representa o estado final para os dois filhos gerados pela operação de cruzamento. Observa-se que as tarefas provenientes do pai 2 são inseridas nos indivíduos filhos na posição que reduz o tempo de processamento para a máquina. Nesse caso, para a máquina 1 do filho 2 as tarefas foram sequenciadas em (4,2,3,6,8). A ordenação original caso esse método de inserção não fosse aplicado seria (4,6,2,3,8).

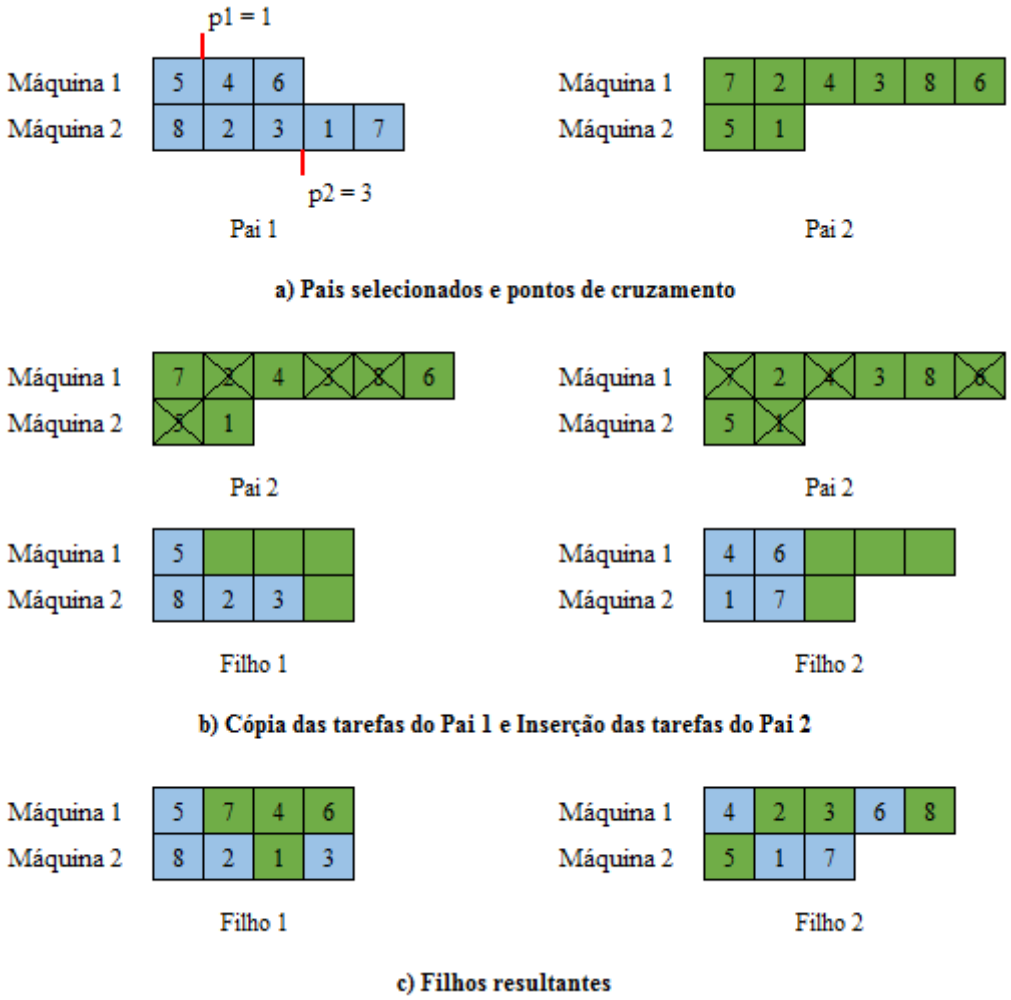


**Figura 9: Exemplo da aplicação do operador de cruzamento.**
**Fonte: Vallada & Ruiz, 2011.**

Após aplicação do operador de cruzamento, os indivíduos filhos gerados podem sofrer algum tipo de mutação. Para o AG proposto os indivíduos podem sofrer até três tipos de mutação: *MutaçãoInserção*, *MutaçãoTrocaMD* e *MutaçãoTrocaMM*. Esses operadores de mutação baseiam-se na utilização de três estruturas de vizinhança: *Inserção*, *TrocaMD* e *TrocaMM*, respectivamente. Para cada operador de mutação, sua respectiva estrutura de vizinhança é aplicada 25 vezes sobre o indivíduo selecionado.

Um indivíduo pode ser submetido à mutação ou não com probabilidade de ocorrência de 30%. Caso a mutação ocorra, apenas um dos três operadores será aplicado no indivíduo, com probabilidade de escolha semelhante para todos eles.

### 3.2.4 Buscas locais e atualização da população

Os procedimentos de busca local são amplamente utilizados nos algoritmos genéticos para melhoria da solução corrente e obtenção de indivíduos mais aptos. O AG proposto utiliza três buscas locais, que serão discutidas detalhadamente na próxima seção, são elas: *BuscaLocalInserção*, *BuscaLocalTrocaMD* e *BuscaLocalTrocaMM*. Após a aplicação dos operadores genéticos, um único indivíduo pode ser submetido sequencialmente aos três procedimentos de buscas locais. Cada procedimento de busca local pode ser aplicado com uma probabilidade de ocorrência equivalente a 50%.

Outro importante aspecto a ser considerado na execução do algoritmo é a forma como os filhos, após a aplicação dos operadores genéticos e procedimentos de busca local, são inseridos na população. Para o AG proposto, novamente utilizou-se a metodologia aplicada por Vallada & Ruiz (2011): os filhos gerados são aceitos apenas quando considerados mais aptos que os piores indivíduos da população. Dessa forma, as gerações evoluem para melhores médias de *makespan* e, ao mesmo tempo, mantêm diferentes soluções, o que ajuda a garantir a diversidade e a evitar a convergência prematura da população.

Finalmente, por meio de testes computacionais, determinou-se que o melhor critério de parada a ser adotado corresponde à execução de dez gerações sem que haja mudança na melhor solução pertencente a cada uma das populações criadas.

## 3.3 Algoritmo VNS-VND

De modo a tornar a comparação entre ambas meta-heurística a mais justa possível, durante o desenvolvimento da meta-heurística VNS-VND, vários aspectos de implementação foram baseados nos procedimentos anteriormente aplicados no AG. Dessa forma, muitos métodos apresentados nessa seção não serão explicados detalhadamente, pois já foram mencionados na seção anterior.

Para a criação da solução inicial para a meta-heurística VNS-VND, optou-se pela utilização da melhor solução gerada a partir da população inicial do AG. Essa solução inicial já possui características positivas, pois constitui o melhor resultado obtido dentre um conjunto de $p$ soluções geradas a partir de uma heurística construtiva, sendo $p$ o tamanho da população do AG.

Para a aplicação da busca local VND, foram utilizadas três estruturas de vizinhança como mecanismos de intensificação da busca: *BuscaLocalInserção*, *BuscaLocalTrocaMD* e *BuscaLocalTrocaMM*. Sempre que um vizinho não apresentar melhoria com relação à solução corrente, utiliza-se a próxima estrutura de vizinhança na busca de um ótimo local. O Quadro 4 apresenta o pseudocódigo para a busca local VND aplicada na meta-heurística VNS-VND.

```
Início;
1.  s' = s0;
2.  k = 0;
3.  Enquanto (k < 3) faça:
4.      Se (k = 0):
5.          Então s'' = BuscaLocalInserção(s')
6.          Senão se (k = 1):
7.              Então s'' = BuscaLocalTrocaMD(s')
8.              Senão s'' = BuscaLocalTrocaMM(s')
10.     Se (custo da solução s'' < custo da solução s'):
11.         Então s' = s''; k = 0;
12.         Senão k = k + 1;
13. Retorne s';
Fim.
```

**Quadro 4: Pseudocódigo para o procedimento VND aplicado.**

A meta-heurística VNS também utiliza três estruturas de vizinhança, tomadas agora como mecanismos de diversificação da busca: *MutaçãoInserção*, *MutaçãoTrocaMD* e *MutaçãoTrocaMM*. Esses procedimentos são os mesmos aplicados como operadores de mutação para o AG e, portanto, baseiam-se na utilização das estruturas de vizinhança *Inserção*, *TrocaMD* e *TrocaMM*, respectivamente. Para cada operador de mutação, sua respectiva estrutura de vizinhança é aplicada 25 vezes sobre a solução corrente.

De modo equivalente ao determinado para a aplicação do AG, toma-se como critério de parada para a meta-heurística VNS-VND a execução de dez repetições para o laço externo do algoritmo sem que haja melhoria na solução corrente. O Quadro 5 apresenta, de forma resumida, o pseudocódigo para a meta-heurística VNS-VND aplicada para resolução do problema abordado neste trabalho.

**Início;**
1. $s^* = s_0$;
2. **Enquanto houver melhoria da solução corrente em até 10 (dez) iterações faça:**
3. $k = 0$;
4. **Enquanto ($k < 3$) faça:**
5. **Se (k = 0):**
6. Então $s' = MutaçãoInserção(s^*)$
7. Senão **se (k = 1):**
8. Então $s' = MutaçãoTrocaMD(s^*)$
9. Senão $s' = MutaçãoTrocaMM(s^*)$
10. $s'' = VND(s')$;
11. **Se (custo da solução $s''$ < custo da solução $s^*$):**
12. Então $s^* = s''$; $k = 0$;
13. Senão $k = k + 1$;
14. Retorne $s^*$;

**Fim.**

**Quadro 5: Pseudocódigo para o procedimento VNS-VND aplicado.**

## 3.4 Estruturas de Vizinhança

Para a resolução do Problema de Sequenciamento de Máquinas Paralelas Não-Relacionadas com Tempo de Preparação Dependente da Sequência são utilizadas três estruturas de vizinhança, todas elas baseadas em movimentos de troca ou realocação de tarefas na mesma máquina ou entre duas máquinas diferentes.

1- *Inserção*: retirada de uma tarefa *a* de sua posição original e inserção em uma nova posição em uma máquina diferente.
2- *TrocaMD*: movimento de troca de posição entre duas tarefas *a* e *b* alocadas em máquinas diferentes.
3- *TrocaMM*: movimento de troca de posição entre duas tarefas *a* e *b* alocadas na mesma máquina.

A Figura 10 apresenta um exemplo de possíveis movimentos de trocas que um indivíduo *i* foi submetido após a aplicação de cada uma das estruturas de vizinhança em tarefas aleatórias pertencentes ao escalonamento. As células em azul correspondem às tarefas que sofreram modificação de posição devido aos movimentos impostos por cada uma das estruturas.

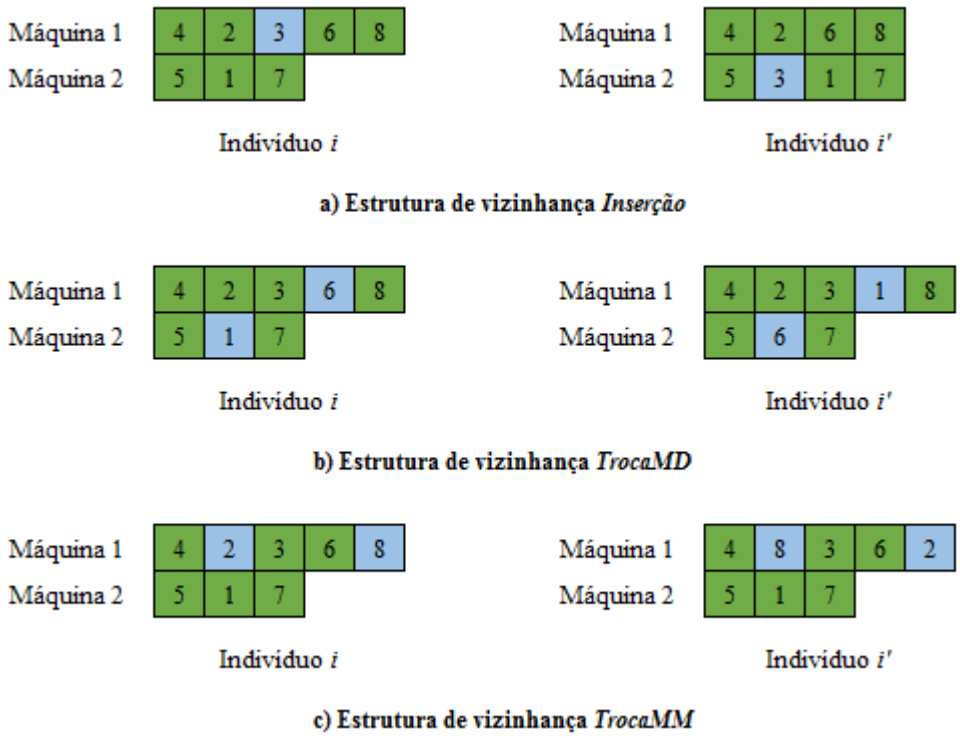


**Figura 10: Movimentos realizados pelas estruturas de vizinhança.**

## 3.5 Métodos de busca local

Os métodos de busca local são amplamente utilizados nas meta-heurísticas para a melhoria dos sequenciamentos obtidos para a solução do problema. Para os algoritmos propostos são aplicados três métodos de busca local, sendo todos eles baseados em uma das estruturas de vizinhança apresentadas na seção anterior.

### 3.5.1 BuscaLocalInserção

A busca local *BuscaLocalInserção* é baseada na estrutura de vizinhança *Inserção* e consiste na análise da possibilidade de remoção de uma tarefa de uma máquina e alocação em todas as posições das demais máquinas existentes. Essa busca utiliza a estratégia *First Improvement*, isto é, sempre realiza o movimento de inserção quando a simulação da mudança de uma tarefa for aceita. Para cada par de máquinas, o movimento de inserção é aceito nos seguintes casos:

1- Se o novo tempo de processamento de ambas as máquinas for reduzido;
2- Caso o tempo de processamento de umas das máquinas for reduzido e o tempo de processamento da outra máquina for aumentado, então o movimento é aceito se o valor do tempo reduzido for maior que o valor do tempo aumentado e se o valor do *makespan*, considerando apenas as duas máquinas, não sofrer nenhuma elevação.

Sempre que um movimento de inserção é aceito (e portanto executado) a busca inicia a simulação da inserção da tarefa subsequente do sequenciamento. A busca só termina quando um ótimo local é encontrado, ou seja, quando não for aceito mais nenhum movimento de inserção para todas as vizinhanças. O Quadro 6 apresenta o procedimento *BuscaLocalInserção*.

```
Início;
1.  s' = s_0;
2.  Inserção = True;
3.  Enquano (Inserção = True) faça:
4.      Inserção = False;
5.      Para (cada máquina M) faça:
6.          Para (cada tarefa i ∈ M) faça:
7.              Para (cada máquina N) faça:
8.                  Se (M ≠ N):
9.                      Para (cada posição p da máquina N) faça:
10.                         Seja s" resultado da inserção da tarefa i na posição p;
11.                         Se (critério de aceitação for atendido):
12.                             s' = s";
13.                             Inserção = True;
14.                             Retorne a busca em 6;
15. Retorne s';
Fim.
```

**Quadro 6: Pseudocódigo para o procedimento *BuscaLocalInserção*.**

### 3.5.2 BuscaLocalTrocaMD

O método de busca local *BuscaLocalTrocaMD* baseia-se na estrutura de vizinhança *TrocaMD*, isto é, na análise da possibilidade de troca de posição de duas tarefas pertencentes a máquinas diferentes. Ele também utiliza a estratégia *First Improvement*, sendo os critérios de aceitação os mesmos estabelecidos para a *BuscaLocalInserção*.

Novamente, sempre que um movimento de inserção é aceito, a análise da troca começa a ser analisado para a tarefa subsequente do sequenciamento. Analisadas todas as tarefas, a busca é reiniciada e só termina quando um ótimo local é encontrado. O Quadro 7 apresenta o pseudocódigo para o procedimento *BuscaLocalTrocaMD*.

```
Início;
1.  s' = s0;
2.  Troca = True;
3.  Enquano (Troca = True) faça:
4.      Troca = False;
5.      Para (cada máquina M) faça:
6.          Para (cada tarefa i ∈ M) faça:
7.              Para (cada máquina N) faça:
8.                  Se (M ≠ N):
9.                      Para (cada tarefa j ∈ N) faça:
10.                         Seja s" o resultado da troca entre as tarefas i e j;
11.                         Se (critério de aceitação for atendido):
12.                             s' = s";
13.                             Troca = True;
14.                             Retorne a busca em 6;
15. Retorne s';
Fim.
```

**Quadro 7: Pseudocódigo para o procedimento *BuscaLocalTrocaMD*.**

### 3.5.3 BuscaLocalTrocaMM

De modo semelhante aos outros procedimentos de busca local, a *BuscaLocalTrocaMM* é baseada em uma estrutura de vizinhança, nesse caso a *TrocaMM*. As trocas de tarefas são realizadas em uma mesma máquina seguindo a estratégia *First Improvement*.

Os movimentos de troca são aceitos sempre que o tempo de conclusão da máquina analisada for reduzido, sendo a busca continuada para as tarefas subsequentes. O algoritmo é reiniciado sempre que houver um movimento de troca e só termina quando um ótimo local for encontrado. O procedimento *BuscaLocalMM* é apresentado em detalhes no Quadro 8.

**Início;**
1. $s' = s_0$;
2. Troca = True;
3. **Enquano (Troca = True) faça:**
4. Troca = False;
5. **Para (cada máquina *M*) faça:**
6. **Para (cada tarefa $i \in M$) faça:**
7. **Para (cada tarefa $j \in M$, com $j > i$ ) faça:**
8. Seja $s$" o resultado da troca entre as tarefas $i$ e $j$;
9. **Se (critério de aceitação for atendido):**
10. $s' = s$";
11. Troca = True;
12. Retorne a busca em 6;
13. Retorne $s'$;

**Fim.**

**Quadro 8: Pseudocódigo para o procedimento *BuscaLocalTrocaMM*.**

# 4 EXPERIMENTAÇÃO

Ambos algoritmos propostos para resolução do Problema de Sequenciamento de Máquinas Paralelas Não-Relacionadas com Tempo de Preparação Dependente da Sequência (PSMPNTDS) foram implementados na linguagem Pascal e compilados no ambiente Lazarus IDE versão 1.2.4. Os experimentos foram realizados em uma máquina com quatro processadores Intel Core i5 de 2,27 GHz com 4 GB de memória RAM. O sistema operacional utilizado foi o Windows 7.

## 4.1 Instâncias para o Problema

Para avaliar a eficiência dos algoritmos, ambas meta-heurísticas foram testadas considerando conjuntos padronizados de dados de entrada para o problema. O tempo de processamento para cada tarefa ($p_{ij}$) foi definido como números inteiros gerados aleatoriamente a partir de uma distribuição uniforme *U*[200,600], enquanto o tempo de preparação ($s_{ijk}$) considerou números inteiros gerados aleatoriamente a partir da distribuição uniforme *U*[0,150].

Para o tamanho do problema, foram gerados conjuntos de instâncias com a combinação de número de máquinas $m = \{4, 6, 8, 10\}$ e número de tarefas $n = \{50, 75, 100 \text{ e } 150\}$. Para cada combinação de máquinas e tarefas forma geradas 100 replicações para o problema. Cada uma das instâncias geradas foi submetida a dois métodos de resolução (aplicação do Algoritmo Genético e da meta-heurística VNS-VND), totalizando 3.200 testes computacionais para a comparação dos algoritmos implementados.

## 4.2 Resultados Computacionais

Para apresentar uma comparação entre os resultados computacionais obtidos por cada uma das meta-heurísticas propostas, inicialmente são analisadas as médias dos conjuntos de dados para cada combinação de máquinas e tarefas sob três aspectos correlacionados: redução do valor do *makespan*, tempo de processamento e amplitude do tempo de processamento para cada solução.

A Tabela 1 apresenta os resultados computacionais obtidos para todas as instâncias testadas considerando a redução do valor do *makespan*. A primeira, segunda e terceira coluna da

tabela apresentam a quantidade de máquinas, de tarefas e de instâncias testadas no experimento, respectivamente. A quarta coluna da tabela apresenta o valor da média do *makespan* para a solução inicial. Conforme mencionado no capítulo anterior, tem-se sempre a mesma solução inicial para o AG e para a meta-heurística VNS-VND, e esta corresponde ao melhor indivíduo pertencente à população inicial gerada para o AG. Por fim, as quatro últimas colunas apresentam a média do valor do *makespan* obtido após a aplicação do AG e meta-heurística VNS-VND e o percentual de redução do valor obtido com base no *makespan* da solução inicial. Os dados em negrito destacam a meta-heurística que obteve melhor percentagem de redução do valor do *makespan*.

**Tabela 1: Valor do *makespan* para o AG e meta-heurística VNS-VND.**

| #Máq. | #Tarefas | Instânc. Testadas | Mak Sol. Inic. | AG | | VNS-VND | |
|---|---|---|---|---|---|---|---|
| | | | | Mak | % Red. | Mak | % Red |
| 4 | 50 | 100 | 4289,33 | 3886,82 | 9,38 | 3876,91 | **9,62** |
| 4 | 75 | 100 | 6389,80 | 5753,34 | 9,96 | 5712,30 | **10,60** |
| 4 | 100 | 100 | 8557,06 | 7691,27 | 10,12 | 7619,24 | **10,96** |
| 4 | 150 | 100 | 12702,28 | 11381,44 | 10,40 | 11247,94 | **11,45** |
| 6 | 50 | 100 | 2742,47 | 2449,12 | 10,70 | 2437,69 | **11,11** |
| 6 | 75 | 100 | 4103,19 | 3650,23 | 11,04 | 3622,08 | **11,73** |
| 6 | 100 | 100 | 5429,26 | 4803,02 | 11,53 | 4748,13 | **12,55** |
| 6 | 150 | 100 | 8101,60 | 7157,54 | 11,65 | 7044,92 | **13,04** |
| 8 | 50 | 100 | 2026,17 | 1809,34 | 10,70 | 1804,25 | **10,95** |
| 8 | 75 | 100 | 3010,37 | 2662,22 | 11,57 | 2646,26 | **12,10** |
| 8 | 100 | 100 | 3978,61 | 3501,76 | 11,99 | 3471,15 | **12,75** |
| 8 | 150 | 100 | 5898,82 | 5196,74 | 11,90 | 5110,55 | **13,36** |
| 10 | 50 | 100 | 1604,16 | 1435,21 | 10,53 | 1427,27 | **11,03** |
| 10 | 75 | 100 | 2366,77 | 2095,49 | 11,46 | 2084,15 | **11,94** |
| 10 | 100 | 100 | 3136,03 | 2769,15 | 11,70 | 2742,18 | **12,56** |
| 10 | 150 | 100 | 4637,03 | 4085,23 | 11,90 | 4011,96 | **13,48** |

O gráfico da Figura 11 apresenta, de modo visual, o percentual de redução do valor do *makespan* da solução inicial para cada uma das meta-heurísticas testadas em todos os casos de aplicação.

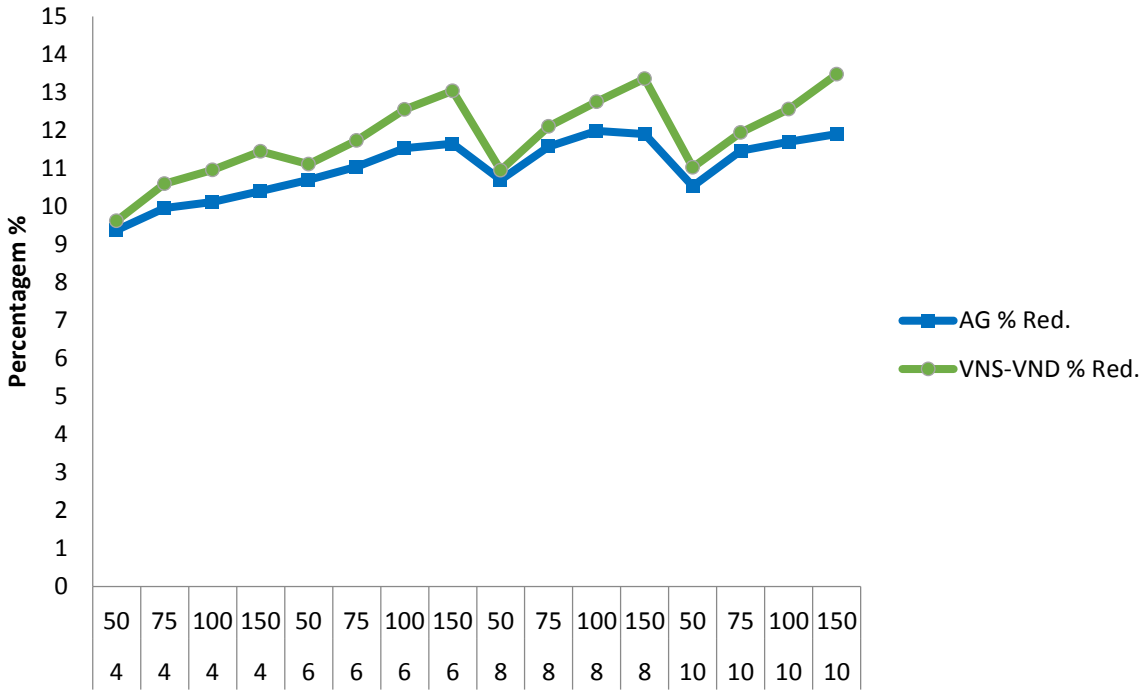


**Figura 11: Percentual médio de redução do valor do *makespan* para todos os casos testados.**

A partir dos dados apresentados na Tabela 1 e do gráfico da Figura 11, é possível verificar que a meta-heurística VNS-VND obteve a menor média do valor do *makespan* para todos os conjuntos de instâncias testadas. Isso significa que, de modo geral, essa meta-heurística alcançou melhor redução do valor do *makespan* a partir da solução inicial e provou melhor eficiência no alcance do objetivo do problema.

A Tabela 2 apresenta o tempo de processamento necessário para a execução de ambas meta-heurísticas. A quarta e quinta coluna representam o tempo médio despendido em segundos para o processamento do AG e da meta-heurística VNS-VND, respectivamente. Como a solução inicial utilizada para a aplicação da meta-heurística VNS-VND corresponde à melhor solução encontrada a partir da geração prévia da população inicial para o AG, o tempo despendido para a geração da população inicial também é considerado na contagem final para o tempo de execução do algoritmo VNS-VND. A sexta e última coluna da tabela corresponde ao percentual de redução da média do tempo de processamento da meta-heurística VNS-VND em relação à média do tempo de processamento do AG.

**Tabela 2: Tempo de processamento para o AG e meta-heurística VNS-VND.**

| #Máq. | #Tarefas | Instânc. Testadas | Tempo (s) AG | Tempo (s) VNS-VND | % Redução |
|---|---|---|---|---|---|
| 4 | 50 | 100 | 0,443 | 0,257 | 41,99 |
| 4 | 75 | 100 | 0,680 | 0,415 | 38,97 |
| 4 | 100 | 100 | 0,959 | 0,630 | 34,31 |
| 4 | 150 | 100 | 2,248 | 1,351 | 39,90 |
| 6 | 50 | 100 | 0,462 | 0,247 | 46,54 |
| 6 | 75 | 100 | 0,765 | 0,433 | 43,40 |
| 6 | 100 | 100 | 1,295 | 0,762 | 41,16 |
| 6 | 150 | 100 | 2,999 | 1,709 | 43,01 |
| 8 | 50 | 100 | 0,511 | 0,257 | 49,71 |
| 8 | 75 | 100 | 0,900 | 0,434 | 51,78 |
| 8 | 100 | 100 | 1,504 | 0,775 | 48,47 |
| 8 | 150 | 100 | 3,452 | 1,733 | 49,80 |
| 10 | 50 | 100 | 0,458 | 0,228 | 50,22 |
| 10 | 75 | 100 | 0,816 | 0,407 | 50,12 |
| 10 | 100 | 100 | 1,340 | 0,708 | 47,16 |
| 10 | 150 | 100 | 3,088 | 1,566 | 49,29 |

O gráfico da Figura 12 resume os dados da Tabela 2, apresentando o tempo médio despendido em segundos para a execução de ambas meta-heurísticas em todos os casos de aplicação.

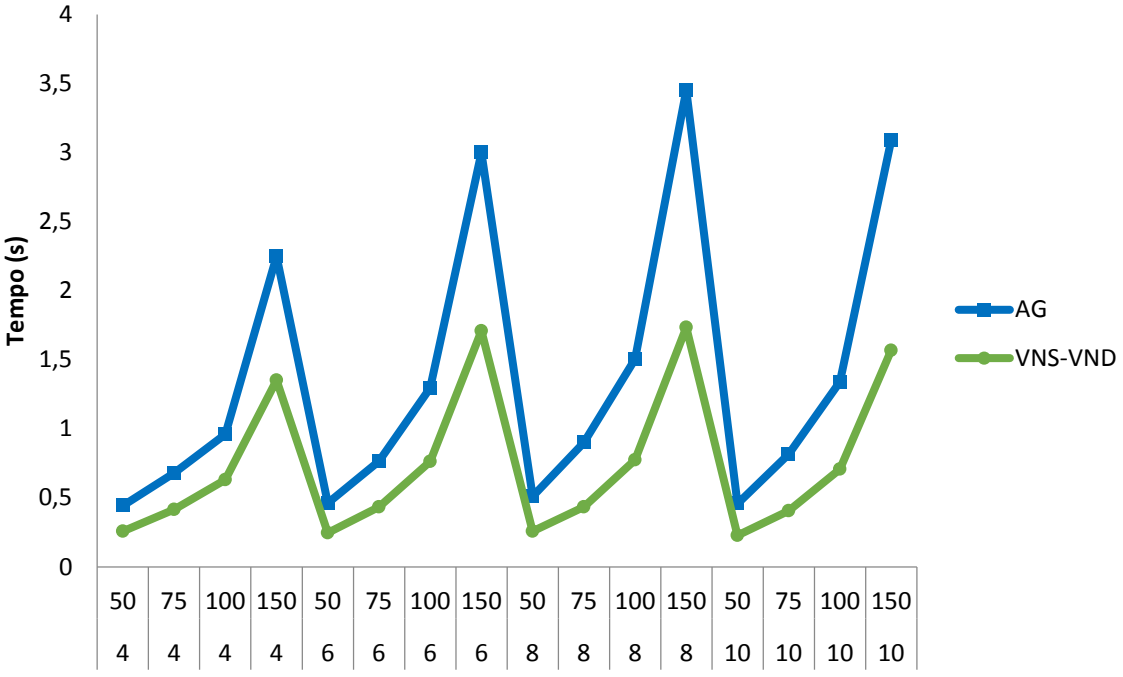


**Figura 12: Tempo médio de processamento para todos os casos testados.**

Com base na Tabela 2 e a partir da análise do tempo de processamento de ambas meta-heurísticas, verifica-se que o algoritmo VNS-VND obteve melhores resultados quando comparado ao AG. Os dados da tabela demostram um percentual de redução no tempo de processamento que varia de 34,31% a 51,78%. É razoável esperar que o tempo de processamento da meta-heurística VNS-VND seja inferior ao tempo de processamento do AG devido às suas próprias características, já que o AG é baseado em população e deve processar vários indivíduos durante cada uma de suas gerações. Além disso, de modo geral, considerando-se a complexidade de ambas meta-heurísticas e as configurações do computador utilizado para experimentação, verifica-se que o tempo obtido para o processamento computacional viabiliza a aplicação prática dos métodos propostos em ambiente fabril.

A Tabela 3 sumariza os dados referentes à amplitude do tempo de processamento para as máquinas pertencentes às soluções encontradas para cada meta-heurística. Os dados da quarta coluna referem-se à média dos tempos de processamento para a máquina que despende o menor tempo para finalizar todas as tarefas a ela associadas. Os dados da quinta coluna referem-se à média do tempo da máquina gargalo, isto é, a média do tempo de processamento para a máquina mais demorada. A sexta coluna representa a amplitude, isto é, a diferença entre os dados representados nas duas colunas anteriores. A quarta, quinta e sexta coluna da tabela representam os dados para as soluções obtidas pelo AG. Já, os dados representados na sétima, oitava e nona coluna correspondem aos mesmos tempos apresentados nas três colunas anteriores, analisando as soluções obtidas pela meta-heurística VNS-VND. Por fim, os valores em negrito destacam a meta-heurística que obteve a melhor amplitude.

A amplitude obtida para as soluções constitui uma importante característica a ser analisada, visto que o objetivo do escalonamento é minimizar o valor do *makespan*. Reduzir o valor do *makespan* para uma solução implica em uma melhor distribuição das tarefas entre as suas máquinas e, consequentemente, em uma maior aproximação do tempo de processamento de cada uma delas (amplitude tendendo a zero).

**Tabela 3: Amplitude do tempo de processamento para o AG e meta-heurística VNS-VND.**

| #Máq. | #Tarefas | Instânc. Testadas | Tempo Máq. AG | | | Tempo Máq. VNS-VND | | |
|---|---|---|---|---|---|---|---|---|
| | | | Menor | Maior | Amplit. | Menor | Maior | Amplit. |
| 4 | 50 | 100 | 3761,2 | 3886,4 | 125,22 | 3765,9 | 3876,9 | **110,96** |
| 4 | 75 | 100 | 5601,7 | 5753,3 | 151,60 | 5597,7 | 5712,3 | **114,63** |
| 4 | 100 | 100 | 7494,3 | 76,91,3 | 196,96 | 7504,8 | 7619,2 | **114,46** |
| 4 | 150 | 100 | 11172,8 | 11381,4 | 208,62 | 11119,9 | 11247,9 | **128,04** |
| 6 | 50 | 100 | 2276,0 | 2449,1 | 173,08 | 2282,28 | 2437,69 | **155,41** |
| 6 | 75 | 100 | 3467,1 | 3650,2 | 186,16 | 3463,9 | 3622,1 | **158,15** |
| 6 | 100 | 100 | 4604,4 | 4803,0 | 198,59 | 4578,3 | 4748,1 | **169,88** |
| 6 | 150 | 100 | 6900,5 | 7157,5 | 257,02 | 6890,0 | 7044,9 | **154,94** |
| 8 | 50 | 100 | 1609,6 | 1809,3 | 199,72 | 1609,6 | 1804,3 | **194,66** |
| 8 | 75 | 100 | 2445,4 | 2662,2 | 216,84 | 2453,2 | 2646,2 | **193,02** |
| 8 | 100 | 100 | 3291,9 | 3501,8 | 209,83 | 3291,8 | 3471,1 | **179,37** |
| 8 | 150 | 100 | 4961,7 | 5196,7 | 235,01 | 4918,2 | 5110,6 | **192,38** |
| 10 | 50 | 100 | 1219,0 | 1435,2 | 216,20 | 1241,9 | 1427,3 | **185,42** |
| 10 | 75 | 100 | 1873,7 | 2095,5 | 221,82 | 1880,7 | 2084,2 | **203,43** |
| 10 | 100 | 100 | 2540,9 | 2769,2 | 228,29 | 2537,8 | 2742,2 | **204,39** |
| 10 | 150 | 100 | 3839,3 | 4085,2 | 245,91 | 3814,0 | 4012,0 | **197,93** |

O gráfico da Figura 13 apresenta a amplitude média obtida para cada um dos casos testados para ambas as meta-heurísticas AG e VNS-VND.

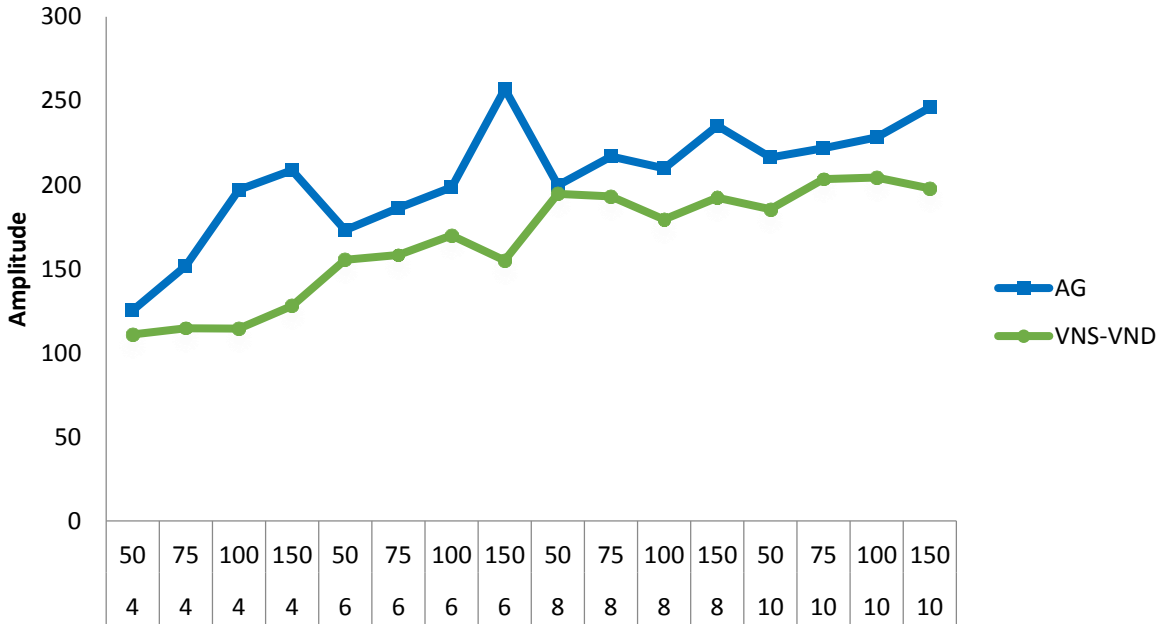


**Figura 13: Amplitude média obtida para todos os casos testados.**

Os dados apresentados na Tabela 3 e ilustrados na Figura 13 novamente reforçam uma melhor eficiência da meta-heurística VNS-VND em comparação aos resultados obtidos pelo AG. Para todos os casos de testes, independendo do tamanho do problema, a meta-heurística VNS-VND sempre obteve melhores amplitudes para o tempo de processamento das máquinas. Esses resultados conectam-se diretamente com os resultados extraídos da Tabela 1, isto é, demonstram que a redução do valor do *makespan* para a meta-heurística VNS-VND é consequência de uma melhor distribuição das tarefas a serem processadas nas máquinas existentes, permitindo a redução da amplitude para o escalonamento.

Diferentemente das tabelas anteriores, os dados da Tabela 4 sumarizam os resultados obtidos pelo AG e meta-heurística VNS-VND considerando a comparação direta e pontual de cada uma das soluções obtidas para as instâncias testadas. As colunas quatro, cinco e seis da tabela representam, para todo o conjunto de 100 instâncias, o percentual de casos em que o AG obteve melhor *makespan*, o percentual de casos em que a meta-heurística VNS-VND obteve melhor *makespan* e o percentual de casos para os quais o *makespan* encontrado para ambos os métodos possui o mesmo valor.

Por meio da análise dos dados da Tabela 4 verifica-se que a meta-heurística VNS-VND sempre obteve um percentual superior de soluções cujo *makespan* é inferior ao obtido pelo AG. Entretanto, verifica-se que para os casos em que o tamanho do problema possui pouca quantidade de tarefas, o AG alcançou percentuais mais expressivos de soluções melhores. A interpretação para esse resultado é a pouca adaptabilidade do AG ao ser considerado diferentes tamanhos de dados de entrada para o problema.

De modo a apresentar um exemplo do desempenho e convergência dos algoritmos implementados, o gráfico da Figura 14 representa o valor do *makespan* obtido considerando as iterações realizadas por ambas meta-heurísticas até que o critério de parada de dez iterações sem melhoria na solução corrente fosse satisfeito (considerando-se como solução corrente a melhor solução para cada geração do AG). Os dados do gráfico representam os resultados obtidos para uma instância aleatória do maior conjunto de dados de entrada para o problema, 10 máquinas paralelas e 150 tarefas em processamento.

**Tabela 4: Percentagem para a melhor solução obtida pelo AG e meta-heurística VNS-VND.**

| #Máq. | #Tarefas | Instânc. Testadas | Melhor Solução | | |
|---|---|---|---|---|---|
| | | | % AG | % VNS-VND | % Empate |
| 4 | 50 | 100 | 35,0 | **63,0** | 2,0 |
| 4 | 75 | 100 | 16,0 | **84,0** | 0,0 |
| 4 | 100 | 100 | 3,0 | **97,0** | 0,0 |
| 4 | 150 | 100 | 1,0 | **99,0** | 0,0 |
| 6 | 50 | 100 | 34,0 | **66,0** | 0,0 |
| 6 | 75 | 100 | 27,0 | **71,0** | 2,0 |
| 6 | 100 | 100 | 7,0 | **93,0** | 0,0 |
| 6 | 150 | 100 | 0,0 | **100,0** | 0,0 |
| 8 | 50 | 100 | 42,0 | **58,0** | 0,0 |
| 8 | 75 | 100 | 28,0 | **71,0** | 1,0 |
| 8 | 100 | 100 | 17,0 | **82,0** | 1,0 |
| 8 | 150 | 100 | 1,0 | **99,0** | 0,0 |
| 10 | 50 | 100 | 35,0 | **62,0** | 3,0 |
| 10 | 75 | 100 | 32,0 | **67,0** | 1,0 |
| 10 | 100 | 100 | 19,0 | **81,0** | 0,0 |
| 10 | 150 | 100 | 1,0 | **99,0** | 0,0 |

Por meio do gráfico da Figura 14, comprova-se que os algoritmos implementados partem de um mesmo valor de *makespan* associado à solução inicial. A partir da execução de cada uma das iterações observa-se que ambas meta-heurísticas apresentam rápida convergência, com expressiva melhoria da solução para as primeiras iterações e convergência menos acentuada para as demais. Para a instância testada, a meta-heurística VNS-VND demostrou melhor capacidade em reduzir o valor do *makespan*, obtendo uma melhor solução com base nos objetivos do problema. Verifica-se que já na segunda iteração, o VNS-VND obteve uma solução melhor do que a solução final para o AG, o que comprova a sua eficiência.

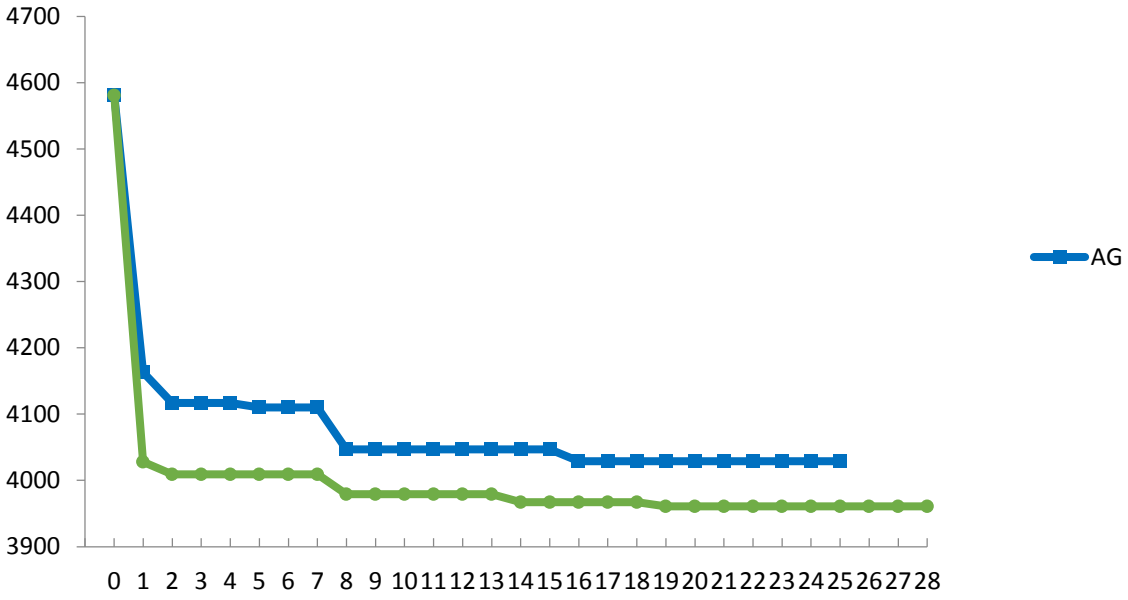


**Figura 14: Iterações para o AG e meta-heurística VNS-VND.**

# 5 CONSIDERAÇÕES FINAIS

Esse trabalho propôs a aplicação de duas meta-heurísticas, algoritmo genético e VNS-VND, para resolução do Problema de Sequenciamento de Máquinas Paralelas Não-Relacionadas com Tempo de Preparação Dependente da Sequência, cujo objetivo principal é a redução do *makespan*, ou tempo de conclusão associado à máquina gargalo para o sequenciamento.

Apesar da expressiva diferença existente entre as características de ambas meta-heurísticas aplicadas, semelhantes aspectos de implementação foram considerados para que os métodos de resolução e a forma de comparação dos resultados obtidos fosse realizada de forma justa, sem que nenhum dos métodos tivesse algum tipo de vantagem sobre o outro.

Os resultados computacionais apresentam diferentes aspectos para as soluções obtidas pelos dois algoritmos. A meta-heurística VNS-VND demostrou melhor capacidade para melhoria da solução inicial e flexibilidade para com o tamanho dos problemas, possibilitando uma melhor distribuição das tarefas existentes e redução da amplitude entre os tempos de processamento das máquinas paralelas. Além disso, o algoritmo VNS-VND apresentou melhor tempo de processamento quando comparado ao AG, com resultados expressivos e tempo de execução hábil que viabiliza a aplicação prática do mesmo em ambiente fabril.

## 5.1 Contribuições

Este trabalho permitiu a análise dos resultados obtidos por dois métodos heurísticos para a resolução de um problema considerado NP-difícil. Esses resultados demostram uma melhor adaptabilidade da meta-heurística VNS-VND na minimização do *makespan* para uma classe pouco explorada dentre os Problemas de Sequenciamento de Máquinas Paralelas (PSMP). Dessa forma, fomentou-se a utilização de novas estratégias e a comparação de diferentes métodos heurísticos na resolução de um problema específico de otimização combinatória na Engenharia de Produção.

## 5.2 Dificuldades e Limitações

A principal dificuldade enfrentada inicialmente diz respeito à própria temática deste trabalho, na identificação do problema clássico da pesquisa operacional para aplicação e resolução por

meio dos métodos heurísticos. Mais tarde, escolheu-se o problema de sequenciamento de máquinas paralelas não-relacionadas com tempo de preparação dependente da sequência devido a sua aplicabilidade e importância prática no contexto da Engenharia de Produção. Também foram encontradas dificuldades na forma de implementação do problema, considerando a não familiaridade com a complexidade e tamanho dos algoritmos resultantes.

## 5.3 Trabalhos Futuros

Como proposta para trabalhos futuros, tem-se a avaliação e teste das duas meta-heurísticas desenvolvidas considerando diferentes aspectos de implementação, como tamanho da população, método de seleção e atualização da população para o AG, estruturas de vizinhança para o VNS-VND e critério de parada para ambos algoritmos.

Além disso, pode-se considerar a aplicação dos métodos propostos para resolução do Problema de Sequenciamento de Máquinas Paralelas Não-Relacionadas com Tempo de Preparação Dependente da Sequência cujo obtivo seja a avaliação de outras medidas de desempenho, além do *makespan*. Dessa forma, características adicionais para o escalonamento podem ser acopladas aos dados de entrada, permitindo a avaliação de outros aspectos de escalonamento como a minimização do atraso máximo para as tarefas ou quantidade de tarefas em atraso. Do mais, adaptações no algoritmo poderiam considerar análises multicritérios, onde várias medidas de desempenho são tomadas simultaneamente para a análise dos resultados.

# REFERÊNCIAS

AARDAL, Karen. *et al*. **A Decade of Combinatorial Optimization**. Maastricht University, Maastricht Research School of Economics of Technology and Organization, 1997.

ARENALES, Marcos. *et al*. **Pesquisa Operacional**. 2ª Reimpressão. Rio de Janeiro: Elsevier, 2007.

BISSCHOP, Johannes. **AIMMS Optimization Modeling**. Paragon Decision Technology, 2000.

BLUM, Christian; ROLI, Andrea. **Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison**. ACM Comput. Surv., v. 35, n. 3, p. 268-308, 2003. Disponível em: <http://kursinfo.himolde.no/forskningsgrupper/optimering/phdkurs/Metaheuristics%20in%20Combinatorial%20Optimization.pdf >. Acesso em: 21 mar. 2013.

BONA, Anderson Andrei de. **Algoritmo de Otimização Combinatorial: Uma Proposta Híbrida Utilizando os Algoritmos Simulated Annealing e Genético em Ambiente Multiprocessado**. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis, 2005. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/102644/223154.pdf?sequence=1>. Acesso em: 23 abr. 2014.

CANTIERE, Patricia Castoldi; BOIKO, Thays Josyanee Perassoli. Programação da Produção em Sistema com Máquinas Paralelas com Tempos de *Setup* Separados dos Tempos de Processamento. In: ENCONTRO DE PRODUÇÃO CIENTÍFICA E TECNOLÓGICA, 6., 2011, Campo Mourão. **Anais...** Campo Mourão: Faculdade Estadual de Ciências e Letras de Campo Mourão, 2011. Disponível em: < http://www.fecilcam.br/nupem/anais_vi_epct/PDF/engenharias/04.pdf>. Acesso em: 28 jul. 2014.

CASTRO, Rodrigo Evangelista de. **Otimização de Estruturas, com Multiobjetivos Via Algoritmo Genético de Pareto**. Tese (Doutorado em Engenharia Civil) – Coordenação dos Programas de Pós Graduação de Engenharia – COPPE, Universidade Federal do Rio de Janeiro – UFRJ, Rio de Janeiro, 2001.

CHAVES, Antonio Augusto. **Uma Meta-Heurística Híbrida com Busca por Agrupamento Aplicada a Problemas de Otimização Combinatória**. Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2009. Disponível em: <http://www.lac.inpe.br/~lorena/antonio/tese-antonio.pdf>. Acesso em: 23 mar. 2013.

CHAVES, Antonio Augusto. *et al*. Metaheurísticas Híbridas para Resolução do Problema do Caixeiro Viajante com Coleta de Prêmios. **Produção**, v. 17, n. 2, p. 263-272, Maio/Ago. 2007.

CONTI, Cássio Rodrigo; ROISENBERG, Mauro. **A Importância da Informação Heurística Visibilidade para Algoritmos Baseados em Otimização por Colônias de Formigas Aplicados a Domínios Contínuos**. Universidade Federal de Santa Catarina. Florianópolis,

2011. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/enia/2011/0038.pdf>. Acesso em: 21 mar. 2014.

FREITAS, Lia Mara Borges de; MONTANÉ, Fermin Alfredo Tang. Metaheurísticas VNS-VND e GRASP-VND para Problema de Roteamento de Veículos com Coleta e Entrega Simultânea. In: SIMPÓSIO DE PESQUISA OPERACIONAL E LOGÍSTICA DA MARINHA, 11., 2008, Rio de Janeiro. **Anais...** Campos dos Goytacazes: UCAM-Campos, 2008.

GOLDGARG, Marco Cesar; LUNA, Henrique Pacca L. **Otimização Combinatória e Programação Linear: Modelos e Algoritmos**. Rio de Janeiro: Elsevier, 2000.

HADDAD, Matheus Nohra. **Algoritmos Heurísticos Híbridos para o Problema de Sequenciamento em Máquinas Paralelas Não-Relacionadas com Tempo de Preparação Dependente da Sequência**. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Ouro Preto (UFOP), Ouro Preto, 2012. Disponível em: < http://www.repositorio.
ufop.br/bitstream/123456789/3269/1/DISSERTA%C3%87%C3%83O_AlgoritmosHeur%C3%ADsticosH%C3%ADbridos.pdf>. Acesso em: 28 jul. 2014.

HANSEN, P.; MLADENOVIC, N. Variable Neighborhood Search Methods. In: FLOUDAS, C. A.; PARDALOS, P. M. **Encyclopedia of Optimization**. Springer Reference, 2009. p. 3975-3976.

HARREL, Charles R; GHOSH, Biman K.; BOWDEN, Royce. **Simulation Using Promodel**. McGraw-Hill, 2000.

HERTZ, Alain; WIDMER, Marino. Guidelines for the Use of Meta-Heuristics in Combinatorial Optimization. **European Journal of Operational Research**, n. 151, p. 247-252, 2003. Disponível em: <http://sci2s.ugr.es/docencia/algoritmica/guidelines-for-the-use-of-meta-heuristics.pdf>. Acesso em: 03 mai. 2014.

HILLIER, Frederick S.; LIEBERMAN, Gerald J. **Introdução à Pesquisa Operacional**. Tradução Ariovaldo Griesi; Revisão técnica João Chang Junior. São Paulo: McGraw-Hill, 2006.

IZQUIERDO, Vaneci Brusch. **Uma proposta de Especificação Formal e Fundamentação Teórica para o *Simulated Annealing***. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, 2000.

LACERDA, E. G. M.; CARVALHO, A. C. P. L. Introdução aos algoritmos genéticos. In: GALVÃO, C. O.; VALENÇA, M.J.S. **Sistemas inteligentes: aplicações a recursos hídricos e ciências ambientais**. Porto Alegre: Ed. Universidade/UFRGS: Associação Brasileira de Recursos Hídricos. p. 99-150, 1999.

LOPES, Heitor Silverio; RODRIGUES, Luiz Carlos de Abreu; STEINER, Maria Teresinha Arns. **Meta-Heurísticas em Pesquisa Operacional**. Curitiba: Omnipax, 2013.

LUGER, George F. **Estruturas e Estratégias para a Solução de Problemas Complexos**. 4. ed. Porto Alegre: Bookmann, 2004.

LUKE, Sean. **Essentials of Metaheuristics**. 2. ed. Fairfax, VA: Lulu, 2013. Department of Computer Science. George Mason University. Disponível em: <http://cs.gmu.edu/~sean/book/ metaheuristics/>. Acesso em: 29 abr. 2014.

MACEDO, Marcelo Alvaro da Silva; ALYRIO, Rovigati Danilo; ANDRADE, Rui Otávio Bernardes de. Análise do Comportamento Decisório: Um Estudo Junto a Acadêmicos de Administração. **Revista de Ciências da Administração**, Santa Catarina, v. 9, n. 18, p. 35-55, mai./ago. 2007. Disponível em: <http://dialnet.unirioja.es/descarga/articulo/4006119.pdf>. Acesso em: 21 mar. 2014.

MARINHO, Euler Horta. **Heurísticas Busca Tabu para o Problema de Programação de Tripulações de Ônibus Urbano**. Dissertação (Mestrado em Computação) – Universidade Federal Fluminense, Niterói, 2005. Disponível em: <http://www.decom.ufop.br/prof/marcone/ Orientacoes/dissEulerFinal.pdf>. Acesso em: 05 mai. 2014.

MELIÁN, José A. Belén; PÉREZ, J. Moreno; VEGA, Marcos Moreno. Metaheurísticas: Una Visión Global. **Revista Iberoamericana de Inteligencia Artificial**. Asociación Española de Inteligencia Artificial, v. 2, n. 19, 2003. Disponível em: <http://jamoreno.webs.ull.es/www/ papers/MMM03IA.pdf>. Acesso em: 21 mar. 2014.

MORAIS, Márcia de Fátima. **Métodos Heurísticos Construtivos para Redução do Estoque em Processo em Ambientes de Produção *Flow Shop* Híbridos com Tempo de *Setup* Dependente da Sequência**. Dissertação (Mestrado em Engenharia de Produção) – Universidade de São Paulo (USP), São Carlos, 2008. Disponível em: <http:// www.teses.usp.br/teses/disponiveis/18/18140/tde-20062008-081005/pt-br.php>. Acesso em: 28 jul. 2014.

MOURA, Arnaldo. *et al*. Técnicas Metaheurísticas Aplicadas à Construção de Grades Horárias Escolares. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 36., 2004, São João Del Rei. **Anais...** São João Del Rei: UFSJ, 2004.

NETO, Luiz Biondi. *et al*. **Fundamentos de Otimização e Inteligência Artificial**. Instituto Nacional de Pesquisas Espaciais, 2010. Disponível em: <http://mtc-m19.sid.inpe.br/col/ sid.inpe.br/mtc-m19@80/2010/01.20.19.22/doc/cap4.pdf>. Acesso em: 03 mai. 2014.

NIEVERGELT, J. *et al*. All the Needles in a Haystack: Can Exhaustive Search Overcome Combinatorial Chaos? In: LEEUWEN, V. J. **Computer Science Today**, Recent Trends and Development. Berlin: Springer-Verlag, 1995. p. 254-274.

ÓLAFSSON, Sigurdur. Metaheuristics. In: NELSON, B. L.; HENDERSON, S. G. **Handbooks in Operations Research and Management Science: Simulation**. Elsevier, 2006. v. 13. p. 633-654.

PAULA, Mateus Rocha de. **Heurísticas para a Minimização dos Atrasos em Sequenciamento de Máquinas Paralelas com Tempos de Preparação Dependentes da Sequência**. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de

Minas Gerais (UFMG), Belo Horizonte, 2008. Disponível em: <http://www.bibliotecadigital.ufmg.br/dspace/handle/1843/RVMR-7PVQT8>. Acesso em: 28 jul. 2014.

PINEDO, Michael L. **Scheduling: Theory, Algorithms and Systems**. New York: Springer, 2008.

RANGEL, Socorro. **Introdução à Construção de Modelos de Otimização Linear e Inteira**. São Carlos: SBMAC; São Paulo: Plêiade, 2005.

REZENDE, Solange Oliveira. **Sistemas Inteligentes: Fundamentos e Aplicações**. Barueri, SP: Manole, 2005.

SARAMAGO, Sezimária F. Pereira. Métodos de Otimização Randômica: Algoritmos Genéticos e Simulated Annealing. In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, 26., Uberlândia, 2003. **Anais...** Uberlândia: Universidade Federal de Uberlândia, 2003. p. 1-40.

SENTHILKUMAR, Panneerselvam; NARAYANAN, Sockalingam. Literature Review of Single Machine Scheduling Problem with Uniform Parallel Machines. **Intelligent Information Management**, Oakland University, USA, n. 2, p. 457-474, 2010.

SILVA, Aneirson Francisco da. **Pesquisa Operacional: Desenvolvimento e Otimização de Modelos Matemáticos por meio da Linguaguem GAMS**. Universidade Estadual Paulista Júlio de Mesquita Filho. 2013. Disponível em: <http://www.feg.unesp.br/~fmarins/GAMS/APOSTILA%20GAMS-2013.pdf >. Acesso em: 21 mar. 2014.

SILVA, Edna Lúcia da; MENEZES, Estera Muszkat. **Metodologia da Pesquisa e Elaboração de Dissertação**. Florianópolis, 2005. 139 p. Disponível em: <http://moodlep.uem.br/course/view.php?id=132>. Acesso em: 11 mar. 2014.

SILVA, Wesley Alves da. **Otimização de Parâmetros da Gestão Baseada em Atividades Aplicada em uma Célula de Manufatura**. Dissertação (Mestrado em Engenharia de Produção) – Universidade Federal de Itajubá, Itajubá, 2005. Disponível em: <http://juno.unifei.edu.br/ bim/0029646.pdf>. Acesso em: 29 abr. 2014.

SOUZA, Marcone Jamilson Freitas**. Problema do Caixeiro Viajante com Coleta de Prêmios**. Universidade Federal de Ouro Preto. Ouro Preto, 2012. Disponível em: <http://www.decom.ufop.br/prof/marcone/Disciplinas/OtimizacaoCombinatoria/PCVCP.pdf>. Acesso em: 21 mar. 2014.

TAHA, Hamdy A. **Pesquisa Operacional: Uma Visão Geral**. Tradução Arlete Simille Marques; Revisão Técnica Rodrigo Arnaldo Scarpel. 8. ed. São Paulo: Pearson Prentice Hall, 2008.

TORGA, Bruno Lopes Mendes. **Modelagem, Simulação e Otimização em Sistemas Puxados de Manufatura**. Dissertação (Mestrado em Engenharia de Produção) – Universidade Federal de Itajubá, Itajubá, 2007. Disponível em: <http://www.iepg.unifei.edu.br/arnaldo/download/ dissertacoes/Bruno.pdf>. Acesso em: 29 abr. 2014.

VALLADA, Eva; RUIZ, Rubén. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependente setup times. **European Journal of Operational Research**, n. 211, p. 612-622, 2011.

VIANA, Ana Maria Marques de Moura Gomes. **Metaheuristics for the Unit Commitment Problem – The Constraint Oriented Neighbourhoods Search Strategy**. Tese (Doutorado em Engenharia Eletrotécnica e de Computadores) – Faculdade de Engenharia da Universidade do Porto, Porto/Portugal, 2004.

WILLIAMS, H. Paul. **Model Building in Mathematical Programming**. 5. ed. John Wiley & Sons, 2013.

YING, Kuo-Ching; LEE, Zne-Jung; LIN, Shih-Wei. Makespan Minimization for Scheduling Unrelated Parallel Machines with Setup Times. **Journal of Intelligent Manufacturing**, v. 23, p. 1795-1803, 2012.

**Universidade Estadual de Maringá**
**Departamento de Engenharia de Produção**
**Av. Colombo 5790, Maringá-PR CEP 87020-900**
**Tel: (044) 3011-4196/3011-5833 Fax: (044) 3011-4196**