UNIVERSIDADE NOVE DE JULHO - UNINOVE
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA E GESTÃO DO CONHECIMENTO - PPGI

APARECIDA DE FÁTIMA CASTELLO ROSA

# ALGORITMO GENÉTICO HÍBRIDO BASEADO NA ANÁLISE DE COMPONENTES PRINCIPAIS DO *FITNESS LANDSCAPE* PARA O PROBLEMA DE *JOB SHOP SCHEDULING*

São Paulo
2019

APARECIDA DE FÁTIMA CASTELLO ROSA

# ALGORITMO GENÉTICO HÍBRIDO BASEADO NA ANÁLISE DE COMPONENTES PRINCIPAIS DO *FITNESS LANDSCAPE* PARA O PROBLEMA DE *JOB SHOP SCHEDULING*

Tese apresentada ao Programa de Pós-Graduação em Informática e Gestão do Conhecimento da Universidade Nove de Julho - UNINOVE, como requisito parcial para a obtenção do título de Doutor em Informática e Gestão do Conhecimento.

Prof. Orientador: Dr. Fábio Henrique Pereira

**São Paulo**
**2019**

Rosa, Aparecida de Fátima Castello.

Algoritmo genético híbrido baseado na análise de componentes principais do Fitness Landscape para o problema de Job Shop Scheduling. / Aparecida de Fátima Castello Rosa. 2019.

137 f.

Tese (Doutorado) - Universidade Nove de Julho - UNINOVE, São Paulo, 2019.

Orientador (a): Prof. Dr. Fábio Henrique Pereira.

1. Algoritmo genético híbrido. 2. Job Shop Scheduling Problem. 3. Fitness landscape. 4. Diversificação e intensificação. 5. Análise de componentes principais.

I. Pereira, Fábio Henrique. II. Titulo.

CDU 004

UNINOVE

Universidade Nove de Julho

## PARECER – EXAME DE DEFESA

Parecer da Comissão Examinadora designada para o exame de defesa do Programa de Pós-Graduação em Informática e Gestão do Conhecimento, a qual se submeteu a aluna regularmente matriculada Aparecida de Fátima Castello Rosa.

Tendo examinado o trabalho apresentado para obtenção do título de "Doutor em Informática e Gestão do Conhecimento", com Tese intitulada "ALGORITMO GENÉTICO HÍBRIDO BASEADO NA ANÁLISE DE COMPONENTES PRINCIPAIS DO FITNESS LANDSCAPE PARA O PROBLEMA JOB SHOP SCHEDULING", a Comissão Examinadora considerou o trabalho:

(X) **Aprovado** ( ) **Aprovado condicionalmente**

( ) **Reprovado com direito a novo exame** ( ) **Reprovado**

Parecer: A banca julgou o trabalho aprovado.

### EXAMINADORES

Prof(a). Dr(a).Fabio Henrique Pereira (Orientador – UNINOVE)

Prof(a). Dr(a). Deisemara Ferreira (Membro Externo – UFSCAR)

Prof(a). Dr(a). Fabio Favaretto (Membro Externo - UNIFEI)

Prof(a). Dr(a). Renato Jose Sassi (Membro Interno – UNINOVE)

Prof(a). Dr(a).Sidnei Alves de Araujo (Membro Interno – UNINOVE)

São Paulo, 01 de julho de 2019.

# Agradecimentos

A Deus por não me deixar fraquejar diante dos vários obstáculos encontrados no decorrer do desenvolvimento deste trabalho e porque sem Ele nada acontece.

Ao professor Dr. Fábio Henrique Pereira pela amizade, orientação, dedicação, compreensão e incentivo dispensado ao desenvolvimento deste trabalho.

Aos membros da banca examinadora pelo tempo dedicado na avaliação do trabalho e sem dúvida pela valiosa contribuição.

Ao meu esposo Luiz e minha filha Talita pelo apoio incondicional, compreensão e incentivo em todos os momentos. À minha irmã Marcia e meu irmão Gabriel e especialmente à minha Mãe Esmeralda *in memoriam*.

As minhas amigas Fernanda Morán Menezes Pereira, Andreia Miranda Domingues e Rosana Cordovil, pela amizade, companheirismo e apoio principalmente nas horas mais difíceis.

Aos professores do PPGI que contribuíram para a minha formação e desenvolvimento deste trabalho.

A Universidade Nove de Julho – UNINOVE, pela concessão da bolsa de estudos.

# Resumo

Este trabalho trata do problema de *scheduling* em ambiente de produção do tipo *job shop*, conhecido na literatura internacional por *Job Shop Scheduling Problem* (JSSP). Em função da sua complexidade computacional, metaheurísticas têm sido comumente empregadas em sua solução, mas um desempenho comparável ao estado da arte depende de uma exploração eficiente das características do espaço de soluções desse problema, o que pode ser realizado por meio da análise do *fitness landscape*. Resumidamente, o *fitness landscape* representa uma paisagem do espaço das soluções geradas - formada pelas soluções no espaço, seus valores de aptidão, e uma noção de vizinhança - e sua analise fornece informações relevantes para o aprimoramento do método de otimização. Assim o objetivo deste trabalho é o desenvolvimento de um Algoritmo Genético Híbrido (HGA) com estratégias de diversificação e intensificação baseadas na Análise de Componentes Principais do *fitness landscape* para o problema de *Job Shop Scheduling*. Para tal, é proposto um método baseado na Análise de Componentes Principais (PCA) para avaliar características de diversidade das soluções de uma população, classificando cada indivíduo com base em sua semelhança aos demais. Essa classificação determina um índice individual de contribuição que é utilizado para selecionar soluções semelhantes que poderão ser substituídas na etapa de diversificação. Quando se tratam de soluções subótimas, a substituição do indivíduo selecionado é feita considerando a melhor solução encontrada na etapa de intensificação. Nesse caso, um Algoritmo Genético Binário Bidimensional (GAB) é utilizado para ampliar a região de busca para além da vizinhança do indivíduo selecionado a procura de uma melhor solução. Por fim, as soluções inicial e final são reconectadas e a melhor solução no caminho entre elas é inserida na população em substituição àquela selecionada pela PCA. Comparando-se o HGA com um método de busca local os resultados mostram que houve melhora em várias instâncias tanto na redução do valor de *makespan* quanto em número de gerações do algoritmo. Em relação a outras abordagens referenciadas na literatura, o HGA obteve resultados competitivos para algumas instâncias, mas ainda se faz necessário um refinamento no método para problemas de maior dimensão, o qual é dependente da escolha adequada dos parâmetros do HGA e na seleção das soluções iniciais para aplicação da intensificação.

**Palavras-chave**: Algoritmo Genético Híbrido, *Job Shop Scheduling Problem*, *Fitness landscape*, Diversificação e intensificação, Análise de Componentes Principais, *Path relinking*.

# Abstract

This work deals with the scheduling problem in a job shop environment, known in the international literature as Job Shop Scheduling Problem (JSSP). Due to their computational complexity, metaheuristics have been commonly employed in their solution, but performance comparable to the state-of-the-art depends on upon an efficient exploration of the solution space characteristics of this problem, through the fitness landscape analysis. Briefly, the fitness landscape represents a space landscape of the generated solutions - formed by the solutions in space, their fitness values, and a notion of neighborhood - and its analysis provides relevant information for improving the optimization method. Thus the objective of this work is the development of a Hybrid Genetic Algorithm (HGA) with diversification and intensification strategies based on the fitness landscape Principal Component Analysis for the Job Shop Scheduling Problem. Therefore, a method based on Principal Component Analysis (PCA) is proposed to evaluate diversity characteristics of solutions of a population, classifying each individual based on their similarity to the others. This classification determines an individual contribution rate that is used to select similar solutions that can be substituted in the diversification stage. When dealing with suboptimal solutions, the replacement of the selected individual is done considering the best solution found in the intensification step. In this case, a Bidimensional Binary Genetic Algorithm (GAB) is used to extend the search region beyond the neighborhood of the selected individual in search of a better solution. Finally, the initial and final solutions are reconnected and the best solution in the path between them is inserted into the population instead of the one selected by the PCA. Comparing the HGA with a local search method the results show that there was improvement in several instances both in reducing the makespan value and in the number of generations of the algorithm. In relation to other approaches referenced in the literature, HGA obtained competitive results for some instances, but it is still necessary to refine the method for larger problems, which is dependent on the adequate choice of HGA parameters and the selection of initial solutions for intensification application.

**Keywords**: Hybrid Genetic Algorithm, Job Shop Scheduling Problem, Fitness landscape, Diversification and intensification, Principal Component Analysis, Path relinking.

# Sumário

# Lista de Figuras

# Lista de Abreviaturas

| | |
|---|---|
| AG | Algoritmo Genético |
| API | (*Application Programming Interface* |
| AS | Vizinhança de troca adjacente, do inglês, *Adjacente swapping neighborhood* |
| BKS | *Best Known Solution* - Solução ótima conhecida |
| BL | Busca Local |
| BRKGA | *Biased Random-Key Genetic Algorithm* |
| COP | *Combinatorial Optimization Problem* |
| CP | Componentes principais |
| FIFO | *First In First Out* |
| FL | *Fitness Landscape* |
| FT | Conjunto de instâncias do problema *Job Shop* proposto por Fisher e Thompson (1963) |
| GA | *Genetic Algorithm* |
| GAB | Algoritmo Genético Binário |
| GABL | Algoritmo Genético com busca local |
| HGA | Algoritmo Genético Híbrido |
| *d*HGA | Procedimento/método computacional para diversificação da população |
| *i*HGA | Procedimento/método computacional para intensificação das buscas |
| JSSP | *Job Shop Scheduling Problem* |
| LA | Conjunto de instâncias do problema *Job Shop* proposto por Lawrence (1984) |
| MKP | *Makespan* |
| PCA | Análise de Componentes Principais, do inglês, *Principal Component Analysis* |
| POC | Problema de Otimização Combinatória, do inglês, *Combinatorial Optimization Problem* - COP |
| PR | *Path-Relinking* |
| RKGA | *Random-Key Genetic Algorithm* |
| RS | Conjunto de referência, do inglês, *Reference set* |
| SP | *Scheduling Problem* |
| TS | *Tabu Search* - Busca Tabu |

# Lista de Símbolos

## Job Shop Scheduling Problem

| | |
|---|---|
| $i$ | Índice subscrito que representa a máquina |
| $j$ | Índice subscrito que representa o *job* |
| $B$ | Bacia de atração do espaço de busca |
| $c_{ji}$ | Tempo de término da operação $O_{ji}$ |
| $y_{ji}$ | Tempo de início da operação $O_{ji}$ |
| $C$ | Conjunto de arcos conjuntivos |
| $C_{max}$ | Variável que representa o *makespan* |
| $D$ | Conjunto de arcos disjuntivos |
| $f$ | Função objetivo ou função *fitness* ou fução de aptidão |
| $J$ | Conjunto de *jobs* |
| $m$ | Variável que representa o número de máquinas |
| $M$ | Conjunto de máquinas |
| $N$ | Conjunto de soluções vizinhas |
| $n$ | Variável que representa o número de *jobs* |
| $O_{ji}$ | Operação do *job* $j$ na máquina $i$ |
| $P$ | Conjunto de ótimos locais |
| $p$ | Solução ótima local |
| $P_{jk}$ | Matriz de tempos de processamento |
| $p_{ji}$ | Tempo de processamento do *job* $j$ na máquina $i$ |
| $S$ | Espaço de soluções/ espaço de busca |
| $s$ | Variável que representa uma solução para o JSSP |
| $T_{jk}$ | Matriz de sequência ou matriz de operações dos *jobs* |
| $V$ | Conjunto de vértices do grafo disjuntivo |

## BRKGA

| | |
|---|---|
| $nk$ | Dimensão do vetor de chaves aleatórias ou número de alelos de um cromossomo do BRKGA |
| $pe$ | Conjunto população elite do BRKGA |
| $pm$ | Conjunto população mutante BRKGA |
| $po$ | Indivíduos gerados pelo operador de cruzamento do BRKGA |
| $pop$ | População do BRKGA |
| $X$ | Vetor de chaves aleatórias do BRKGA |
| $Y$ | Representa decodificação do vetor de chaves aleatórias $X$ |

**PCA**

| | |
|---|---|
| $\lambda$ | Autovalor da matriz $A$ |
| $\eta$ | Número de componentes principais |
| $\zeta$ | Número de indivíduos na população do GA |
| $A$ | Matriz de coeficientes de regressão normalizados |
| $A'$ | Representação da matriz $A$ no espaço dos componentes principais |
| $A_j$ | $j$-ésima coluna da matriz $A$ |
| $CI$ | Índice de contribuição de um indivíduo para diversidade populacional |
| $CP$ | Componentes Principais |
| $e_i$ | *Eigenvectors* |
| $p$ | Autovetores da matriz $A$ |
| $R$ | Matriz de projeção no espaço dos componentes principais |
| $U$ | Matriz de covariância |
| $u_i$ | Autovetor ou componente principal da matriz $A$ |

Capítulo 1

# Introdução

## 1.1 Contextualização

A produção eficaz é uma das questões mais críticas no ambiente competitivo atualmente visto que as empresas estão sob pressão para atender às necessidades e requisitos do cliente e manter sua satisfação. Nesse contexto, o planejamento e programação da produção têm um grande impacto no aumento da sua eficiência. A programação da produção é um processo de tomada de decisão que diz respeito à alocação de recursos limitados entre tarefas concorrentes ao longo do tempo, com o objetivo de otimizar um ou mais objetivos. Portanto, a programação adequada leva a maior eficiência e utilização da capacidade, menor tempo necessário para concluir tarefas e, consequentemente, maior lucratividade para a organização (YAZDANI *et al.*, 2017).

Nas indústrias de manufatura a programação da produção é responsável por programar e controlar o uso dos recursos no processo produtivo. Um dos objetivos nesse contexto consiste em determinar quais máquinas devem ser utilizadas para processar as operações de produção, bem como programar uma sequência para processamento dessas operações. Trata-se de uma atividade crítica, pois pode envolver uma série de fatores como, por exemplo, a disponibilidade de recursos, o consumo de materiais, etc. (NOOR *et al.*, 2015; PINEDO, 2012).

Especificamente, a programação de uma sequência para execução das operações no ambiente de produção é um problema de otimização chamado de *Scheduling Problem* (SP), que vem sendo estudado desde a década de 1950 devido a sua importância prática e complexidade computacional. O problema envolve determinar em que ordem os *jobs* devem ser processados pelas máquinas disponíveis, sendo *job* definido como um conjunto de operações (LENSTRA; RINNOOY KAN; BRUCKER, 1977; MORALES; RONCONI, 2016). As operações de um *job* respeitam uma relação de precedência definida em cada problema, chamada sequência tecnológica, mas a ordem de processamento de operações de diferentes *jobs* em uma mesma máquina pode ser modificada para atender a algum objetivo.

Assim, o objetivo do SP é determinar uma sequência de processamento das operações em cada máquina, de forma a otimizar alguma medida de desempenho da produção como, por exemplo, a minimização do tempo total de processamento de todos os *jobs*, denominado de *makespan* (ARENALES *et al.*, 2011).

Trata-se de um problema de difícil solução classificado como NP-*hard* (*nondeterministic polynomial-time hard*), o que significa que o tempo de execução para que um algoritmo garanta a melhor solução cresce exponencialmente com o tamanho do problema (ÇALIŞ; BULKAN, 2015; GAO *et al.*, 2011; LENSTRA; RINNOOY KAN; BRUCKER, 1977).

A complexidade dos problemas de *scheduling* é especialmente elevada em ambientes do tipo *job shop*, nos quais as operações de cada um dos $n$ *jobs* devem ser processadas nas $m$ máquinas, de acordo com uma sequência tecnológica única para cada *job*, e com tempos de processamento determinísticos. Nesse tipo de ambiente o problema é chamado *Job Shop Scheduling Problem* (JSSP) clássico, o qual é considerado como um dos problemas de otimização mais difíceis de serem resolvidos (KUHPFAHL; BIERWIRTH, 2016; PINEDO, 2012).

## 1.2 Justificativa

O JSSP é classificado como um problema de otimização combinatória, uma vez que o seu espaço de soluções é formado pelas várias possibilidades de combinar as operações dos *jobs* nas máquinas. O número total de todas as combinações possíveis é dado por $(n!)^m$ (MORALES; RONCONI, 2016). Por exemplo, para a instância do problema proposto por Fisher e Thompson (1963), denominada FT10 com 10 *jobs* e 10 máquinas (100 operações), o tamanho do espaço de soluções é $(10!)^{10}$, aproximadamente $3,96 \times 10^{65}$. Para se ter ideia da complexidade dessa instância, a solução ótima foi encontrada por Carlier e Pinson (1989) somente um quarto de século após a sua publicação, utilizando método exato *branch and bound* (BAYKASOĞLU; HAMZADAYI; KÖSE, 2014; GAO *et al.*, 2011; GRABOWSKI; WODECKI, 2005). Ressalta-se que espaço de soluções e espaço de busca são sinônimos e são utilizados indistintamente.

Conforme aumenta o tamanho do espaço de soluções do JSSP, torna-se difícil o uso de métodos exatos como, por exemplo, *Shifting Bottleneck Procedure* (ADAMS; BALAS; ZAWACK, 1988) e *Branch and Bound* (AKKAN; KARABATI, 2004), os quais avaliam todo o espaço de soluções em busca da solução ótima. Assim, os métodos exatos são aplicados a problemas considerados pequenos, necessitando mais de duas horas em problemas com mais de 140 operações (FUCHIGAMI; MOURA; BRANCO, 2017; GAO *et al.*, 2015; ZHAO *et al.*, 2017).

Alternativamente, as técnicas metaheurísticas têm recebido atenção dos pesquisadores para resolver o JSSP. Essas técnicas são constituídas por algoritmos de busca que percorrem o espaço de soluções do problema e possuem mecanismos para escapar de ótimos locais, encontrando soluções de boa qualidade, com custo computacional em geral inferior àqueles obtidos com a utilização de métodos exatos (BASSEUR; GOËFFON, 2015; GAO *et al.*, 2011; GAO *et al.*, 2015).

Com respeito às técnicas metaheurísticas, existem dois diferentes grupos de abordagens. Algumas realizam buscas a partir de uma única solução, tais como, *Simulated Annealing* (KIRKPATRICK; GELATT; VECCHI, 1983), *Iterated Local Search* (LOURENÇO; MARTIN; STÜTZLE, 2003), Busca Tabu (*Tabu Search* (TS)) (GLOVER; LAGUNA, 1997), *Variable Neighborhood Search* (MLADENOVIĆ; HANSEN, 1997). Outras técni-

cas trabalham simultaneamente com um conjunto de soluções, chamado de população. Como exemplos dessa categoria cita-se o Algoritmo Genético (AG) (*Genetic Algorithm* (GA)) (HOLLAND, 1975), *Biased Random-Key Genetic Algorithm* (BRKGA) (Algoritmo Genético de Chaves Aleatórias) (TOSO; RESENDE, 2015) que pertence a classe dos Algoritmos Genéticos, Algoritmo Colônia de Formigas (*Ant Colony Optimization*) (DORIGO; MANIEZZO; COLORNI, 1996), Algoritmo por Enxame de Partículas (*Particle Swarm Optimization*) (KENNEDY; EBERHART, 1995), entre outras. Em geral, para o JSSP, as técnicas do primeiro grupo são utilizadas em estratégias de busca local, como extensões para melhorar a exploração do espaço de busca, em conjunto com as do segundo grupo.

Apesar do sucesso na resolução do JSSP, as metaheurísticas, trabalhando sozinhas, não são capazes de encontrar soluções ótimas para todas as instâncias do problema. Sabe-se que essa dificuldade está relacionada não apenas ao tamanho dos problemas, mas também a outras características do espaço de busca (MALAN; ENGELBRECHT, 2014; WATSON, 2010). Consequentemente, diversos autores investigaram essas características por meio da análise da superfície que é formada pela variação dos valores de *makespan* das diferentes soluções no espaço de busca, chamada de *fitness landscape*, com intuito de ampliar a compreensão das limitações das metaheurísticas nesse problema (BIERWIRTH; MATTFELD; WATSON, 2004; LU *et al.*, 2018; MATTFELD; BIERWIRTH; KOPFER, 1999; MIRSHEKARIAN; ŠORMAZ, 2016).

Portanto, do ponto de vista teórico, pode-se observar os esforços realizados para ampliar os conhecimentos sobre o espaço de busca e, assim explicar as dificuldades dos problemas e/ou das técnicas em resolver o JSSP. Por outro lado, do ponto de vista prático, existem pesquisas que exploram esses conhecimentos para o desenvolvimento de métodos mais efetivos, como por exemplo, Bożejko *et al.* (2018), Nowicki e Smutnicki (2005a), Pardalos, Shylo e Vazacopoulos (2010), Wong *et al.* (2008), Wong *et al.* (2010). Nesse sentido, Watson, Howe e Whitley (2006) propuseram uma desconstrução do Algoritmo Busca Tabu, o qual sabidamente apresenta um bom desempenho nesse problema, e concluíram que a aplicação de mecanismos de intensificação e diversificação é fundamental para que a metaheurística alcance bons resultados.

Resumidamente, diversificação é uma busca de diferentes áreas do espaço de busca para encontrar regiões mais promissoras. Intensificação é uma busca na vizinhança dos indivíduos para produzir melhores soluções (GAO *et al.*, 2011). Ambos os conceitos são definidos com base nas características do *fitness landscape*. A fundamentação sobre *fitness landscape* e, diversificação e intensificação é abordada no Capítulo 2, Seções 2.3 e 2.5.3, respectivamente.

Entretanto, apesar das tentativas de levar o conhecimento teórico sobre o *fitness landscape* para a prática (BÜRGY, 2017), ainda são escassas as abordagens que realizam uma análise *online* do *fitness landscape* com vistas a gerar métodos e mecanismos mais adaptados às características específicas de cada problema. Ademais, conforme pode ser visto

em Bożejko *et al.* (2017), Bożejko *et al.* (2018) e Nagata e Ono (2017), diversas instâncias do JSSP permanecem em aberto, ou seja, a solução ótima ainda é desconhecida.

## 1.3 Objetivos

O principal objetivo deste trabalho é o desenvolvimento de um Algoritmo Genético Híbrido (HGA) com estratégias de diversificação e intensificação baseadas na análise de componentes principais do *fitness landscape* para o problema de *Job Shop Scheduling*.

### 1.3.1 Objetivos Específicos

1. Investigar as características do *fitness landscape* obtido a partir da aplicação do HGA com Busca Local.
2. Avaliar a diversidade populacional do HGA com busca local ao longo do processo de otimização, em especial em momentos de estagnação da convergência.
3. Desenvolver e avaliar uma abordagem de diversificação da população com base na Análise de Componentes Principais (PCA, do inglês) do *fitness landscape* para identificar grupos de indivíduos com características semelhantes.
4. Desenvolver e avaliar uma abordagem de intensificação da busca pela solução ótima.

## 1.4 Organização da Tese

Este trabalho está organizado em seis Capítulos, descritos a seguir:

- Capítulo 1 - Na introdução é apresentada a contextualização deste trabalho bem como a identificação das lacunas, justificativa e objetivos.
- Capítulo 2 - Este Capítulo apresenta a fundamentação teórica sobre problemas de otimização, problemas de sequenciamento em ambientes de produção em geral e, especialmente sobre *Job Shop Scheduling Problem* (JSSP). Em seguida, apresenta-se a fundamentação sobre *fitness landscape*, Algoritmo Genético e *Biased Random-Key Genetic Algorithm* (BRKGA), Análise de Componentes Principais (PCA) e finalmente apresenta-se o método *Path-Relinking* (PR) .
- Capítulo 3 - Neste Capítulo apresenta-se a revisão da literatura sobre as técnicas metaheurísticas utilizadas na resolução dos problemas de sequenciamento da produção, principalmente para o JSSP. Apresenta-se também trabalhos que tratam da análise do *fitness landscape*.

- Capítulo 4 - São apresentados neste Capítulo os materiais e métodos utilizados para o desenvolvimento deste trabalho.
- Capítulo 5 - Apresenta-se neste Capítulo a discussão dos resultados obtidos com as abordagens propostas.
- Capítulo 6 - Neste Capítulo apresenta-se as conclusões, limitações, sugestões para continuidade e contribuições da pesquisa.

Além dos Capítulos relacionados são apresentadas as referências bibliográficas.

Capítulo 2

# Fundamentação Teórica

Este Capítulo apresenta alguns dos principais conceitos sobre problemas de otimização, problemas de sequenciamento em ambientes de produção em geral e, em especial, no ambiente do tipo *job shop*, chamados *Job Shop Scheduling Problem* (JSSP). Discute-se também as características do *fitness landscape* e o espaço de soluções para esses problemas. Finalmente, são apresentados os fundamentos do Algoritmo Genético e do *Biased Random-Key Genetic Algorithm*, Análise de Componentes Principais e o método *Path Relinking*.

## 2.1 Problemas de otimização

Os problemas de otimização são de grande importância tanto para o mundo industrial quanto para o científico. Muitos problemas de otimização consistem na escolha da melhor configuração ou conjunto de variáveis para alcançar um ou mais objetivos. Eles geralmente envolvem a minimização ou maximização de uma função objetivo com uma ou mais variáveis, sobre um determinado domínio, sujeito a um conjunto de restrições (BLUM; ROLI, 2003; BLUM, 2005).

Os problemas de otimização geralmente são classificados em dois grupos: aqueles que são codificados com variáveis contínuas denominados de Problema de Otimização Contínuo e, os que são representados por variáveis discretas os quais são denominados de Problemas de Otimização Combinatória (POC) (do inglês, *Combinatorial Optimization Problem* - COP) (RESENDE; RIBEIRO, 2016a). Os problemas de *scheduling* de uma forma geral (apresentado na Seção 2.2) são exemplos de POC.

Os POC's reduzem-se à busca por uma solução em um conjunto finito (ou, alternativamente, infinitamente contável), que geralmente pode ser formado por variáveis binárias ou inteiras, permutações, caminhos, árvores ou grafos.

## 2.2 Problemas de Scheduling e suas representações

Os Problemas de *Scheduling* (SP, do inglês) são problemas de otimização combinatória de grande importância tanto para a academia quanto para a indústria manufatureira. Dentre os ambientes de produção pode-se citar:

- *Job shop* ($Jm$): em um ambiente de produção do tipo *job shop* com $m$ máquinas, cada *job* tem a sua própria sequência, também chamada de rota ou roteiro, pré-determinada para seguir. Nesse ambiente existe uma distinção no qual os *jobs* devem passar uma única vez na máquina, ou no caso com recirculação, que pode ser processado pela mesma máquina mais de uma vez.

- *Flow shop* ($Fm$): o ambiente de produção do tipo *flow shop* é um caso particular do *job shop*. Existem $m$ máquinas em série. Cada *job* têm que ser processado em cada uma das $m$ máquinas. Todos os *jobs* tem a mesma sequência nas $m$ máquinas, por exemplo, eles tem que ser processado primeiro na máquina 1, depois na máquina 2, e assim por diante. Após completar o processo em uma máquina, o *job* entra em uma fila para ser processado na próxima máquina. Geralmente, as filas seguem a disciplina *First In First Out (FIFO)*, ou seja, um *job* não pode assumir o lugar de outro enquanto espera na fila.

- *Open shop* ($Om$): existem $m$ máquinas. Cada *job* tem que ser processado novamente em cada uma das $m$ máquinas. Entretanto, alguns desses tempos de processamento poderão ser iguais a zero. Não há restrições no que diz respeito a sequência de cada *job* através do ambiente de máquina. O planejador da produção pode determinar a sequência para cada *job* e, *jobs* diferentes podem ter sequências diferentes.

Merece destaque neste trabalho o problema de *scheduling* em ambiente tipo *job shop*, apresentado a seguir. A notação, definições e terminologia apresentadas baseiam-se nos trabalhos do Pinedo (2012) e Yamada (2003).

### 2.2.1 Job Shop Scheduling Problem

O problema geral do mínimo-*makespan* no sequenciamento no ambiente tipo *job shop* $n \times m$ é designado pelos símbolos $n/m/G/C_{max}$. Geralmente conhecido como *Job Shop Scheduling Problem* (JSSP), pode ser descrito como um conjunto de $n$ *jobs* $\{J_j\}_{1 \leq j \leq n}$ que são processados em um conjunto de $m$ máquinas $\{M_i\}_{1 \leq i \leq m}$ e pode ser caracterizado, segundo Asadzadeh (2015), Pinedo (2012), Yamada e Nakano (1997) e Yamada (2003) da seguinte forma:

a) todas as máquinas estão disponíveis no instante de tempo $t_0 = 0$ e são liberadas no instante $t_0 = 0$;

b) cada *job* deve ser processado em cada máquina de acordo com a ordem predefinida na sequência tecnológica das máquinas;

c) cada máquina pode processar somente um *job* por vez;

d) o processamento do *job* $J_j$ na máquina $M_i$ é denominado de operação $O_{ji}$;

e) a operação $O_{ji}$ requer o uso exclusivo da máquina $M_i$, por uma duração de tempo de processamento ininterrupto, $p_{ji}$; sendo que não é permitido interrupção da operação, isto é, não admite preemptividade, significando que não é permitida a interrupção, ainda que temporária, do processamento de uma operação em execução;

f) o tempo de início e o tempo de término de uma operação $O_{ji}$ são representados por $y_{ji}$ e $c_{ji}$ respectivamente. O sequenciamento é um conjunto de tempos de término para cada operação $c_{ji}, 1 \leq j \leq n, 1 \leq i \leq m$ de tal forma que satisfaçam as restrições anteriores. Alternativamente, uma solução pode ser apresentada apenas como uma sequência de processamento das operações em cada máquina, com tempos $y_{ji}$ e $c_{ji}$ obtidos a partir dessa sequência;

g) o tempo requerido para o término de todas as operações é chamado de *makespan*, o qual é designado por $C_{max}$. Por definição, $C_{max} = \max_{1 \leq j \leq n, 1 \leq i \leq m} c_{ji}$.

O *makespan*, definido como $\max(C_1, \cdots, C_n)$, é equivalente ao tempo de término do último *job* quando sai do sistema.

O problema é geral, utilizando-se o símbolo $G$, no que diz respeito a sequência tecnológica das máquinas que pode ser diferente para cada *job*, conforme item (b); a ordem dos *jobs* a serem processados em uma máquina também pode ser diferente para cada máquina.

O JSSP pode ser representado de forma matricial, por gráfico de Gantt e grafo disjuntivo. Essas representações são apresentadas nas Seções à seguir.

### 2.2.2 Representações do Job Shop Scheduling Problem

Para ilustrar as diferentes formas de representações do JSSP adotou-se o exemplo de um problema com 3 *jobs* e 3 máquinas apresentado em Yamada (2003), definido conforme Tabela 2.1.

**Tabela 2.1:** *Exemplo de um problema JSSP com 3 jobs e 3 máquinas. Cada coluna representa a sequência tecnológica das máquinas para cada job com seu tempo de processamento entre parênteses.*

| ***Job*** | **Máquina (tempo de processamento)** | | |
|---|---|---|---|
| J1 | 1(3) | 2(3) | 3(3) |
| J2 | 1(2) | 3(3) | 2(4) |
| J3 | 2(3) | 1(2) | 3(1) |

Fonte : Traduzido de Yamada (2003).

Na Tabela 2.1 cada linha determina a sequência tecnológica de máquinas para o *job* correspondente e, o tempo de processamento do *job* naquela máquina é apresentado entre parênteses. Por exemplo, nessa Tabela, a linha 1 corresponde as operações do *job* 1 que deverão ser realizadas na seguinte ordem: $O_{11} \longrightarrow O_{12} \longrightarrow O_{13}$, ou seja, o *job* 1 é processado primeiro na máquina 1 e seu tempo de processamento é igual a 3, depois na máquina 2 com tempo de processamento igual a 3 e por último, na máquina 3 com tempo de processamento igual a 3. Para os demais *jobs* apresentados no exemplo, segue-se a mesma lógica: *job* 2: $O_{21} \longrightarrow O_{23} \longrightarrow O_{22}$; e *job* 3: $O_{32} \longrightarrow O_{31} \longrightarrow O_{33}$.

#### 2.2.2.1 Representação Matricial

A sequência tecnológica predefinida para cada *job* pode ser dada como uma matriz $\{T_{jk}\}$, na qual cada elemento $t_{jk} = i$ corresponde à $k$-ésima operação do *job* $j$ na máquina $i$, denotada por $O_{ji}$ (YAMADA, 2003).

À partir dos dados apresentados na Tabela 2.1, o problema pode ser representado pela matriz de sequência, ou matriz de operações dos *jobs* $\{T_{jk}\}$ e, pela matriz de tempos de processamento $\{P_{jk}\}$ conforme ilustra a Figura 2.1

$$\{T_{jk}\} = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \end{bmatrix} \qquad \{P_{jk}\} = \begin{bmatrix} 3 & 3 & 3 \\ 2 & 3 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

**Figura 2.1:** *Matriz de sequência $T_{jk}$ dos jobs, e matriz dos tempos de processamento $P_{jk}$ para o problema $3 \times 3$ apresentado na Tabela 2.1. O elemento $t_{jk} = i$ representa que a k-ésima operação do job $J_j$ deve ser processado na máquina $M_i$ por $p_{ji}$ unidades de tempo.*

Fonte : Adaptado de Yamada (2003).

#### 2.2.2.2 Representação por Gráfico de Gantt

O gráfico de Gantt é uma importante ferramenta visual que facilita a análise e visualização da utilização e das operações nas máquinas ao longo do tempo, bem como o tempo total, ou seja, o *makespan*, para a execução de todas as operações nas respectivas máquinas.

O eixo da abscissa representa a unidade de tempo e o eixo da ordenada representa as máquinas ou recursos. Cada retângulo no gráfico representa uma operação que é atribuída a uma máquina em um determinado instante de tempo. A largura do retângulo corresponde ao tempo de processamento da operação na máquina (GAO *et al.*, 2015).

A Figura 2.2 ilustra um exemplo do gráfico de Gantt para uma solução ótima do problema apresentado na Tabela 2.1.

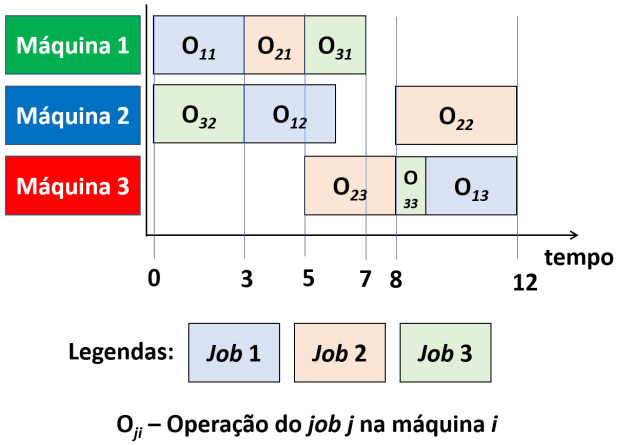



**Figura 2.2:** *Representação gráfico de Gantt para uma solução ótima do exemplo apresentado na Tabela 2.1.*

Fonte : Adaptado de Yamada (2003).

Como pode ser observado no gráfico de Gantt (Figura 2.2), a máquina 1 realiza as operações dos *jobs* 1, 2 e 3, nessa ordem, e termina todas as operações com 7 unidades de tempo; a máquina 2 tem as operações dos *jobs* 3, 1 e 2, nessa ordem, e termina as operações com 12 unidade de tempo, observa-se também que houve um tempo de espera da máquina entre as operações dos *jobs* 1 e 2. Já a máquina 3, aguardou durante 5 unidades de tempo até que uma operação fosse liberada para ser processada nessa máquina e, a sequência das operações realizadas nessa máquina é *job* 2, 3 e 1, que também termina as operações com 12 unidades de tempo. O tempo total para esse sequenciamento, ou seja, o *makespan* é de 12 unidades de tempo.

A representação matricial e o gráfico de Gantt descritos são formas simples e eficientes de representar as soluções dos problemas. No entanto, elas não trazem outras informações, como por exemplo, se o sequenciamento é factível ou não, ou seja, se a sequência das operações nas máquinas está de acordo com a sequência tecnológica predefinida.

#### 2.2.2.3 Representação por Grafo Disjuntivo

O grafo disjuntivo é uma representação gráfica que apresenta as característica do problema JSSP, sendo possível uma melhor visualização do sequenciamento das operações e da sequência tecnológica nas máquinas.

O problema de sequenciamento da produção JSSP pode ser representado por um grafo disjuntivo $G = (V, C \cup D)$, introduzido por (BALAS, 1969), no qual:

- $V$ é um conjunto de vértices $(j, i)$, também chamados de nós, que representam todas as operações $O_{ji}$ dos *jobs*, e mais dois vértices especiais: um vértice inicial, denotado por 0, e um vértice final, denotado por $*$, que não representam operações, mas apenas indicam o início e o término do sequenciamento, respectivamente, e estão associados

a tempos nulos. Aos demais vértices é associado o tempo de processamento $p_{ji}$ da operação $O_{ji}$.

- $C$ é um conjunto de arcos conjuntivos (arcos orientados) representando a sequência tecnológica de máquinas para cada *job*. Nesse conjunto existe um arco para cada par ordenado de operações $O_{ji} \rightarrow O_{jk}$. Se, por exemplo, existe um arco entre os vértices $(j,i)$ e $(j,k)$, então existe uma relação de precedência para o *job* $j$ que deverá ser processado primeiramente na máquina $i$ e depois na máquina $k$, definindo uma ordem de processamento das operações de um mesmo *job*.

- $D = \bigcup_{i=1}^{m} D_i$, no qual $D_i$ é um conjunto de arcos disjuntivos (arcos não orientados) representando os pares de operações que devem ser processadas na mesma máquina $M_i$.

O grafo disjuntivo para o problema $3 \times 3$ apresentado na Tabela 2.1 é ilustrado na Figura 2.3.

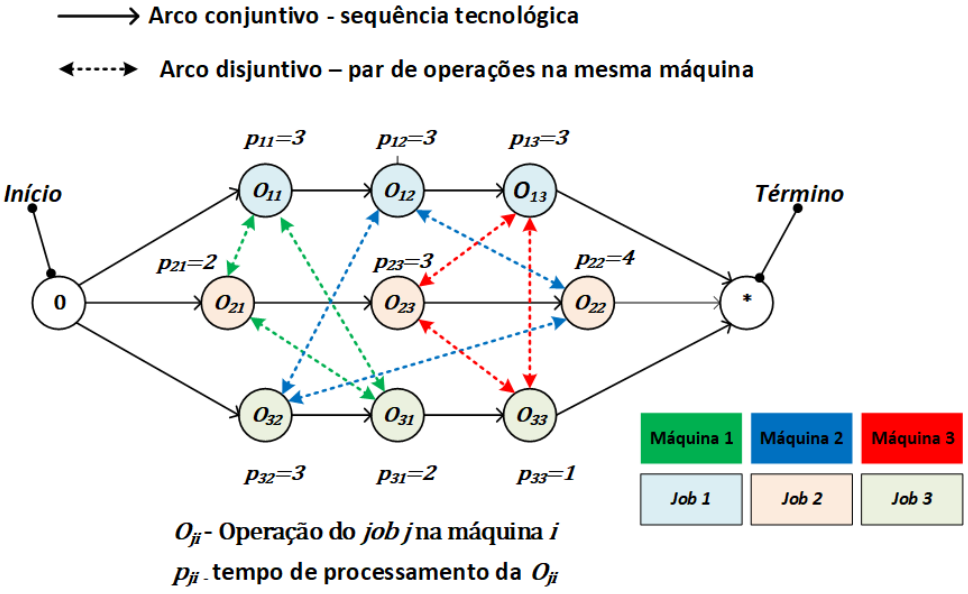



**Figura 2.3:** *Representação do JSSP no grafo disjuntivo para o exemplo apresentado na Tabela 2.1.*

Fonte : Adaptado de Yamada (2003).

No grafo disjuntivo (Figura 2.3), os arcos conjuntivos são representados pelas setas sólidas e os arcos disjuntivos pelas setas tracejadas, indicando os pares de operações que serão processadas em uma mesma máquina.

Assim, a representação de um sequenciamento no grafo disjuntivo é obtida atribuindo-se uma direção aos arcos disjuntivos, ou seja, definindo uma ordem de processamento para as operações em uma mesma máquina. Se o novo grafo criado por essa atribuição, chamado

grafo conjuntivo for acíclico, significa que não existem conflitos de precedência entre as operações e que o sequenciamento é factível. No grafo conjuntivo, o comprimento do caminho mais longo entre o vértice inicial e o vértice final, também chamado de caminho crítico ou *critical path* é igual ao valor de *makespan* ($C_{max}$) do sequenciamento factível (MENDES, 2003). O problema de minimizar o *makespan* pode ser definido como encontrar uma seleção de arcos disjuntivos que minimizam o comprimento do caminho mais longo, ou seja, do caminho crítico (PINEDO, 2012)

De acordo com Pinedo (2012) a formulação de programação disjuntiva para o JSSP está relacionada com a representação do grafo disjuntivo. Sendo a variável $y_{ji}$ que representa o tempo de início da operação $O_{ji}$; $V$ o conjunto de todas as operações $O_{ji}$; e $C$ o conjunto de todas as restrições da sequência tecnológica $(j,i) \rightarrow (j,k)$ implica que o *job* $j$ requer o processamento na máquina $i$ antes de ser processado na máquina $k$. A formulação matemática que minimiza o *makespan* pode ser descrita como segue:

$$\begin{aligned}
&\text{minimizar: } C_{max} && && \\
&\text{subjeto à} && && \\
&y_{jk} - y_{ji} \geq p_{ji} && \forall (j,i) \rightarrow (j,k) \in C && \text{(2.1a)} \\
&C_{max} - y_{ji} \geq p_{ji} && \forall (j,i) \in V && \text{(2.1b)} \\
&y_{ji} - y_{li} \geq p_{li} \quad \text{ou} \quad y_{li} - y_{ji} \geq p_{ji} && \forall (l,i) \quad \text{e} \quad (j,i), i = 1, \cdots, m && \text{(2.1c)} \\
&y_{ji} \geq 0 && \forall (j,i) \in V && \text{(2.1d)}
\end{aligned}$$

na qual $y_{ji}$ representa o tempo de início da operação $O_{ji}$; $p_{ji}$ o tempo de processamento do *job* $j$ na máquina $i$.

Nessa formulação, o primeiro conjunto de restrições (Equação 2.1a) garante que a operação $(j,k)$ não pode ser iniciada antes que a operação $(j,i)$ seja concluída. O segundo conjunto de restrições (Equação 2.1b) estabelece o limite inferior para $C_{max}$ garantindo que ele sempre será maior ou igual ao tempo de início da operação $O_{ji}$ e o tempo necessário para o seu processamento, ou seja, o $C_{max}$ será um tempo suficiente para processar todas as operações. O terceiro conjunto de restrições (Equação 2.1c) é chamado de restrição disjuntiva; elas garantem que deve existir uma ordem entre as operações de diferentes *jobs* que precisam ser processados na mesma máquina. Devido a essas restrições, essa formulação é chamada de formulação de programação disjuntiva (PINEDO, 2012).

Considera-se o exemplo de 3 *jobs* e 3 máquinas apresentado na Tabela 2.1 (Seção 2.2.2) para ilustrar a programação disjuntiva. A sequência tecnológica de máquinas bem como o tempo de processamento foram transcritos para a Tabela 2.2 apresentada a seguir.

**Tabela 2.2:** *Exemplo do problema JSSP com 3 jobs e 3 máquinas para a Programação Disjuntiva*

| *Jobs* | Sequência de máquinas | Tempo de processamento |
|---|---|---|
| J1 | 1, 2, 3 | $p_{11} = 3,\ p_{12} = 2,\ p_{13} = 3$ |
| J2 | 1, 3, 2 | $p_{21} = 2,\ p_{23} = 3,\ p_{22} = 4$ |
| J3 | 2, 1, 3 | $p_{32} = 3,\ p_{31} = 2,\ p_{33} = 1$ |

Fonte : A autora.

O primeiro conjunto de restrições (Equação 2.1a) consiste de seis restrições: duas para cada *job* neste exemplo,

$$y_{12} - y_{11} \geq 3 \quad (= p_{11}).$$

O segundo conjunto, Equação 2.1b, consiste de nove restrições, uma para cada operação, sendo, por exemplo

$$C_{max} - y_{11} \geq 3 \quad (= p_{11}).$$

O terceiro conjunto de restrições disjuntivas (Equação 2.1c) contém nove restrições sendo três para cada máquina (existem três operações para serem realizadas em cada máquina). Um exemplo de uma restrição disjuntiva é

$$y_{11} - y_{21} \geq 2 \quad (= p_{21}) \text{ ou } y_{21} - y_{11} \geq 3 \quad (= p_{11}).$$

O último conjunto (Equação 2.1d) inclui nove restrições de não-negatividade, uma para cada tempo de início.

Uma solução do JSSP para essa formulação é o conjunto de tempos de início para todas as operação, ou seja, os valores de $s$. Alternativamente, esse solução também pode ser dada como uma sequência das operações dos *jobs* nas máquinas. À partir dessa sequência é possível obter os tempos de início.

O grafo conjuntivo de uma solução ótima para o exemplo definido na Tabela 2.1 é ilustrado na Figura 2.4. Observa-se que dos $n$ arcos disjuntivos (3 no exemplo) que conectavam operações em uma mesma máquina, um arco foi removido e $n-1$ arcos foram direcionados de modo a evitar ciclos, definindo uma sequencia factível.

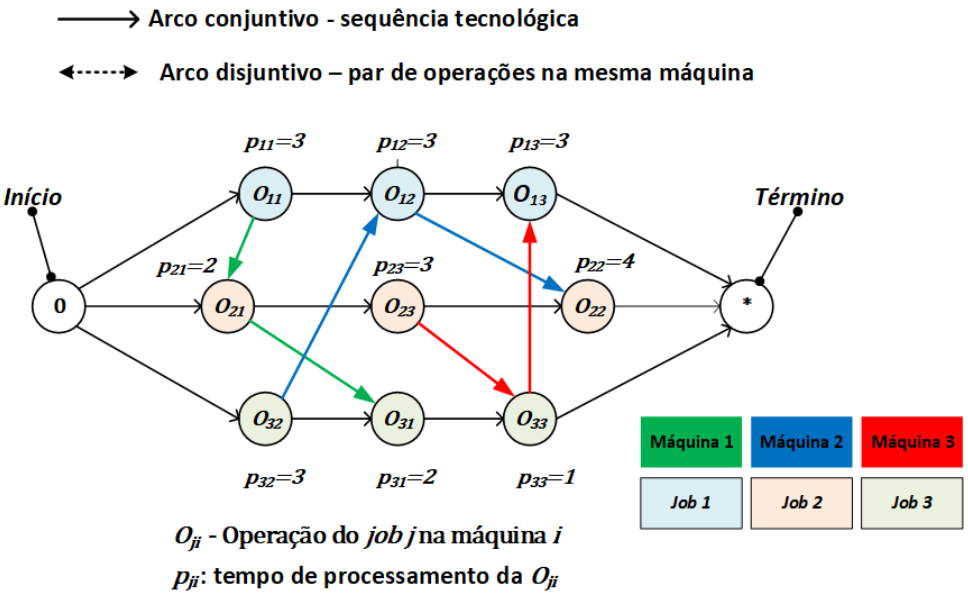



**Figura 2.4:** *Grafo conjuntivo para uma solução ótima do grafo disjuntivo (Figura 2.3) para o exemplo do JSSP apresentado na Tabela 2.1.*

Fonte : Adaptado de Yamada (2003).

Outros sequenciamentos factíveis são obtidos permutando a ordem de processamento das operações nas máquinas, ou seja, a sequência de operações na máquina, mas sem violar as restrições tecnológicas.

Em virtude de que cada sequência de operações em uma máquina pode ser permutada independentemente das sequências de operações das outras máquinas, tem-se um máximo de $(n!)^m$ soluções diferentes para uma instância do problema, sendo $n$ *jobs* e $m$ máquinas envolvidas. Para o exemplo apresentado na Tabela 2.1 (3 *jobs* x 3 máquinas, 9 operações), existem 216 soluções diferentes; já para um problema maior, por exemplo, para a instância FT10 com 10 *jobs* e 10 máquinas (100 operações) proposto por Fisher e Thompson (1963), tem-se $(10!)^{10}$, ou seja, aproximadamente $3,96 \times 10^{65}$ soluções, caracterizando a dificuldade do problema (BAYKASOĞLU; HAMZADAYI; KÖSE, 2014). De acordo com Mattfeld (1996), "mesmo problemas de tamanho moderado manterão qualquer computador ocupado por séculos" para testar todas as possíveis combinações de sequências.

Portanto, o conhecimento prévio das características do problema e do espaço de busca poderá ajudar no desenvolvimento de algoritmos ou refinar as técnicas já existentes. Mattfeld, Bierwirth e Kopfer (1999) e Bierwirth, Mattfeld e Watson (2004) ressaltam a importância do conhecimento do espaço de busca, da estrutura de vizinhança e do *fitness landscape* para o JSSP.

A eficiência dos algoritmos depende da estrutura do espaço de busca, assim, uma abordagem bem-sucedida é considerar o espaço de busca e suas propriedades por meio da análise do *fitness landscape* que é a principal ferramenta para o estudo do espaço de busca (BOŻEJKO *et al.*, 2018; LU *et al.*, 2017; SMITH-MILES; LOPES, 2012).

## 2.3 Fitness Landscape

O conceito do *fitness landscape* foi proposto pela primeira vez pelo biólogo Wright (1932) na década de 1930, na biologia teórica, como uma forma de visualizar a distribuição dos valores de aptidão (*fitness*) das combinações de genes, na adaptação evolutiva.

A principal ideia é visualizar um espaço genótipo de uma espécie como uma superfície, formada pelo *fitness* de todos os membros desse espaço, na qual os genótipos relacionados ocupam locais próximos. Essa superfície é denominada de *fitness landscape* e, metaforicamente, assemelha-se a uma região montanhosa com picos, vales, cumes e planaltos (BIERWIRTH; MATTFELD; WATSON, 2004; MATTFELD; BIERWIRTH; KOPFER, 1999). A Figura 2.5 ilustra parcialmente um *landscape* para uma instância do JSSP.

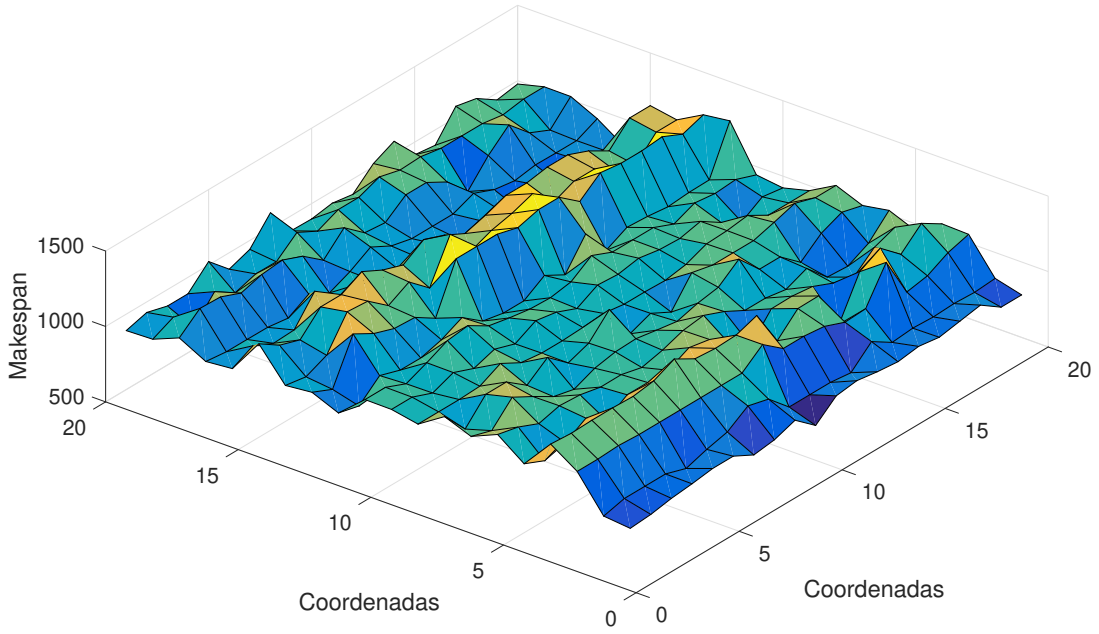



**Figura 2.5:** *Representação parcial de um fitness landscape para uma instância do JSSP.*
Fonte : A autora.

Atualmente, o conceito do *fitness landscapes* é amplamente utilizado em diversas áreas de pesquisa e, principalmente, na computação evolucionária, na qual os comportamentos de busca e as representações dos problemas podem ser analisados tanto do ponto de vista teórico quanto empírico (BASSEUR; GOËFFON, 2015). No domínio da otimização, o objetivo da caracterização dos *landscapes* é relacionar a descrição topológica do espaço de soluções com a dinâmica dos algoritmos de busca, e assim, extrair informações sobre a dificuldade de um determinado problema de otimização (MOSER; GHEORGHITA; ALETI, 2017).

Formalmente, o *fitness landscape* é representado por três componentes $\{S, N, F\}$ (MATTFELD; BIERWIRTH; KOPFER, 1999; MOSER; GHEORGHITA; ALETI, 2017; REIDYS; STADLER, 2002) no qual:

- $S$ é o conjunto de soluções para um determinado problema, denominado *espaço de busca* ou *espaço de soluções*.
- $N : S \longrightarrow S$ é a relação de vizinhança, que atribui a cada solução $s \in S$ a um conjunto de vizinhos $N(s) \subset S$ por meio de um operador de busca. Duas soluções são vizinhas se, pela aplicação de um operador de busca a uma solução é possível alcançar uma outra.
- $f : S \longrightarrow \Re$ é uma função *fitness*, também denominada de função objetivo ou função de aptidão, que associa um valor para cada solução, representando a qualidade da solução a ser minimizada ou maximizada.

A fundamentação a seguir tem por base o trabalho de Moser, Gheorghita e Aleti (2017).

A vizinhança de uma solução depende do operador aplicado na otimização então, um determinado problema pode ter qualquer número de *fitness landscapes*. Para o objetivo de descrever o *fitness landscape*, é comum a utilização de operadores locais que realizam pequenas mudanças na solução. O objetivo é criar *landscapes* no qual a distância entre uma solução e seu vizinho seja a menor possível quando comparada com outras soluções.

A topologia de um *fitness landscape* é composta por certas estruturas, tais como, *ótimo local*, *ótimo global* e *bacias de atração*. Em um problema de minimização, dado o espaço de busca $S$ e uma relação de vizinhança $N$, um *ótimo local* é um ponto $s^l \in S$ tal que para qualquer solução $s^n \in N, f(s^l) <= f(s^n)$. Um *ótimo global* é um ponto $s^g \in S$ que é um *ótimo local* e também $\forall s \in S, f(s^g) < f(s)$.

Uma *bacia de atração* $B$ de um ótimo local $s^l$ é definida como um conjunto de pontos $s_1 \cdots s_k$ do espaço de busca, tal que um algoritmo da máxima descida (*steepest descent*) iniciando em $s_i$, $(1 \leq i \leq k)$ termina no ótimo local $s^l$ em um número finito de passos.

O grau de dificuldade de um *fitness landscape* depende, em grande parte, da distribuição e do número de ótimos locais, conhecido como *modalidade*, e das bacias de atração em volta.

Assim, as características do *fitness landscape* se relacionam intimamente com os conceitos de distância e vizinhança. O objetivo fundamental da análise do *landscape* é detectar a regularidade do espaço de busca, especialmente o "*big valley*", que pode desempenhar o papel de um *atrator* no espaço de busca. A presença de "*big valley*" significa que as distâncias entre boas soluções ótimas locais e as soluções ótimas globais estão significativamente correlacionadas positivamente com os valores da função objetivo dessas soluções. Também significa que a maioria dos ótimos locais muito bons se concentram em uma área

relativamente pequena do espaço de soluções. A "regularidade" é também uma qualificação muito poderosa, uma vez que os problemas combinatórios frequentemente têm um grande número de ótimos locais, enquanto as soluções próximas no espaço de soluções podem ter valores de função objetivo significativamente diferente (NOWICKI; SMUTNICKI, 1996).

Teoricamente, o "*big valley*" justifica a filosofia do método *path relinking* (Seção 2.7), projetado com base na crença de que ao longo da trajetória ligando dois ótimos locais pode-se encontrar um novo ótimo local ou mesmo um ótimo global (GLOVER; LAGUNA, 1997). Na prática, o espaço de soluções pode ter natureza muito mais complicadas (NOWICKI; SMUTNICKI, 2005b).

A formação do "*big valley*" nas características do *landscape* tem forte implicação de como a heurística de busca deve ser executada. A estrutura do "*big valley*" sugere que a determinação de novos pontos iniciais para a busca deve basear-se no ótimo local anterior, em vez de basear-se em um ponto aleatório no espaço de busca. Isso ocorre porque boas soluções candidatas são frequentemente encontradas próximas de outras. Por meio da intensificação e diversificação das áreas próximas a esses locais de forma eficaz, a busca poderá ser direcionada para o ótimo global, eventualmente (WONG *et al.*, 2008).

Outra característica importante do espaço de soluções refere-se ao "*backbone*". Informalmente, o "*backbone*" de uma instância de um problema é o conjunto de atributos da solução que possuem valores idênticos em todas as soluções ótimas para essa instância (WATSON *et al.*, 2003).

### 2.3.1 Espaço de busca do Job Shop Scheduling Problem

O *landscape* para o *Job Shop Scheduling Problem* requer uma representação adequada e, com base nessa representação, uma definição de vizinhança (MATTFELD; BIERWIRTH; KOPFER, 1999). Uma estrutura de vizinhança pode ser definida com base na representação do grafo disjuntivo (Seção 2.2.2.3) para uma instância do JSSP.

A partir de uma instância de um problema, são especificados tanto o espaço de soluções ou espaço de busca, quanto a função objetivo. Como mencionado anteriormente, para o JSSP, por exemplo, o espaço de busca consiste na combinação de $n$ *jobs* e $m$ máquinas sendo $(n!)^m$ sequências possíveis (BAYKASOĞLU; HAMZADAYI; KÖSE, 2014; GAO *et al.*, 2011). A função objetivo para o JSSP retorna o valor do *makespan* de uma sequência.

Seja $S$ um conjunto de todos os sequenciamentos ou soluções factíveis para uma instância do JSSP. Um sequenciamento $s_1 \in S$ é movido para $s_2$ permutando a relação de precedência de duas operações sucessivas na mesma máquina. Dessa forma, um movimento nessa vizinhança realiza a menor modificação possível em uma solução (MATTFELD; BIERWIRTH; KOPFER, 1999).

Qualquer tipo de definição de vizinhança define implicitamente o conjunto de ótimos locais $P \subset S$ (Seção 2.3). Um ótimo local $p \in P$ é dado se $f(p) \leq f(s)\ \forall\ s \in N(p)$. Soluções

ótimas locais de $P$ são geradas à partir de uma solução qualquer de $S$. Soluções que levam ao mesmo ótimo local $p$ formam a bacia de atração de $p$ no *landscape* (MATTFELD; BIERWIRTH; KOPFER, 1999).

### 2.3.2 Estrutura de vizinhança do Job Shop Scheduling Problem

Uma estrutura de vizinhança no JSSP definida com base no grafo disjuntivo, é um mecanismo que pode obter um novo conjunto de soluções vizinhas aplicando-se uma pequena perturbação a uma determinada solução, sendo que, cada solução vizinha é alcançada a partir de uma solução por um único movimento (ZHANG *et al.*, 2007).

Para o JSSP, uma vizinhança $N(s)$ de uma solução $s$ pode ser definida como um conjunto de sequenciamentos que podem ser alcançados a partir de $s$ por uma única perturbação em $s$. O operador de transição que define uma vizinhança pela permutação de pares de operações consecutivas no caminho crítico (operações críticas) foi introduzido por Balas (1969) em sua abordagem *branch and bound*, denominada de vizinhança de troca adjacente (do inglês, *adjacente swapping neighborhood* (AS)) conforme apresentado na Figura 2.6.

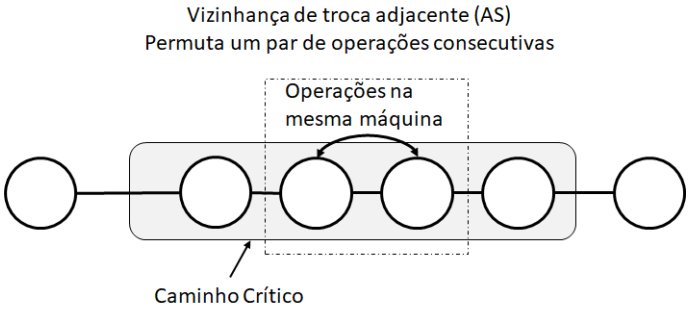



**Figura 2.6:** *Ilustração do operador de transição definido em Balas (1969), o qual permuta pares de operações críticas adjacentes*

Fonte : Adaptado de Yamada e Nakano (1997).

Exitem seis principais estruturas de vizinhança para o JSSP definidas na literatura como N1, N2, $\cdots$ , N6 (ver Amirghasemi e Zamani (2014)). Essas estruturas são baseadas no caminho crítico de um grafo conjuntivo. Por exemplo, a vizinhança N1 proposta por Laarhoven, Aarts e Lenstra (1992), define um movimento para permutar duas operações sucessivas na mesma máquina no caminho crítico. A estrutura de vizinhança N1 foi elaborada com base em dois princípios: ($a$) alterar a ordem de duas operações não-críticas não pode melhorar a solução e pode apenas criar um ciclo no grafo disjuntivo, e ($b$) alterar a ordem de duas operações adjacentes não pode criar um ciclo. As demais estruturas de vizinhança diferem da N1 apenas na escolha das operações críticas que serão permutadas.

### 2.3.3 Busca Local

O principio básico dos algoritmos de Busca Local (BL) é realizar uma perturbação nas soluções existentes com vistas a obter melhorias. Eles partem de uma solução inicial factível, a qual é gerada aleatoriamente ou por meio de algum procedimento heurístico, e tentam iterativamente substituir parte ou a solução inteira por uma solução factível melhor em um conjunto apropriadamente definido de soluções vizinhas. Para substituir partes de uma solução inicial, os métodos de busca local executam uma série de movimentos que levam à formação de novas soluções na mesma vizinhança. O espaço de busca, a função objetivo, o operador de movimento e a estratégia para guiar a busca no espaço de busca por meio da exploração na vizinhança são os principais componentes de uma busca local (MATTFELD; BIERWIRTH; KOPFER, 1999; RESENDE; RIBEIRO, 2016b).

O operador de movimento especifica o conjunto de modificações permitidas, ou seja, a vizinhança, para a solução atual $s$ em qualquer iteração específica, uma das quais é selecionada por meio da estratégia de navegação que servirá de base para a próxima iteração. A melhor solução encontrada em qualquer iteração é armazenada e retornada quando o algoritmo termina, normalmente após um limite de tempo ou quando um número máximo de iterações é alcançado (WATSON, 2010).

Em um espaço de busca $S$, a noção de localidade em um algoritmo de busca local é fornecida pelo operador de movimento ou operador de vizinhança $N$, o qual define o conjunto de modificações permitidas para a solução atual $s$ em qualquer iteração. Uma solução vizinha é obtida por meio de uma única aplicação de um operador de movimento em uma solução. Dada uma solução $s$, o conjunto $N(s)$ é conhecido como vizinhança de $s$. Da mesma forma, se $s' \in N(s)$, então $s'$ é vizinho de $s$ (WATSON, 2010).

Um problema específico define um espaço de busca único. Diferentes *landscapes* são criados pelos diferentes operadores de busca heurísticos utilizados para pesquisá-lo (REEVES, 1999). A estrutura do *landscape* muda com o operador, dependendo da forma como os operadores são aplicados.

Estudos sobre a análise do *fitness landscape* para o JSSP comprovam que trata-se de um problema muito difícil de ser resolvido, assim sendo, diversos algoritmos de aproximação foram proposto ao longo do tempo para a resolução do problema, ainda que esses métodos não garantam que a solução ótima seja encontrada.

Destacam-se nesse contexto, por exemplo, o *Tabu Search* (SMUTNICKI; BOŻEJKO, 2018; NOWICKI; SMUTNICKI, 2005a); *Particle swarm optimization* (SHA; LIN, 2010); *Bee colony optimisation* (WONG *et al.*, 2010); Algoritmos Genéticos (MATTFELD; BIERWIRTH, 2004; ASADZADEH, 2015), *Biased random-key genetic algorithm* (GONÇALVES; MENDES; RESENDE, 2005; GONÇALVES; RESENDE, 2014). A busca tabu predominou por muito tempo como abordagem mais eficiente para o JSSP, por possuir aspectos adequados às características do espaço de busca do problema. Destacam-se as

aplicações com busca tabu para o JSSP: TSAB de (NOWICKI; SMUTNICKI, 1996), *i*-TSAB de (NOWICKI; SMUTNICKI, 2005a) e TSSA de (ZHANG *et al.*, 2008).

Conforme levantamento realizado por Çaliş e Bulkan (2015), o Algoritmo Genético é provavelmente uma das técnicas metaheurísticas mais utilizadas na resolução JSSP.

## 2.4 Algoritmo Genético

O Algoritmo Genético (AG) foi criado por Holland (1975) e em 1975, em seu livro "*Adaptation in Natural and Artificial Systems*" apresentou o AG como uma abstração da evolução biológica e deu uma estrutura teórica para a adaptação no domínio do AG. Ele se inspirou no mecanismo da evolução das espécies da teoria de Charles Darwin e nos trabalhos de Mendel sobre genética natural (HAUPT; HAUPT, 2004; MITCHELL, 1998).

O AG pode ser visto como um método de busca e otimização não determinístico que, em um grande espaço de busca, procura uma solução candidata que obtenha o melhor resultado, o qual é definido como aquele que otimiza uma medida numérica predefinida para o problema, denominada de função de aptidão ou função *fitness*. Apesar de não ser garantido encontrar uma solução candidata "ótima", frequentemente o AG encontra soluções candidatas de alta aptidão com tempo computacional relativamente rápido. Ele tem sido aplicado a problemas de otimização tais como problemas de layout de circuito, *job shop scheduling problem*, otimização de processos e parâmetros, dentre outros (GOLDBERG, 1989; MITCHELL, 1997).

O AG opera de forma iterativa atualizando um conjunto de indivíduos, chamado população. A cada iteração os indivíduos da população (ou cromossomos) são gerados e avaliados de acordo com uma função de aptidão, também chamada de função *fitness* para evoluírem em busca da solução de um determinado problema de otimização. Uma nova população é gerada probabilisticamente, selecionando os indivíduos mais aptos da população atual. Alguns destes indivíduos selecionados são transportados para a próxima geração da população sem modificação, outros são utilizados como base para a criação de novos indivíduos descendentes por meio da aplicação de operações genéticas como cruzamento e mutação. Dessa forma, a cada nova geração deve haver um indivíduo mais próximo de uma solução ótima para o problema (HAUPT; HAUPT, 2004; MITCHELL, 1997).

No contexto do AG os termos cromossomo e indivíduo podem ter o mesmo significado; eles são uma estrutura de dados que representam uma das possíveis soluções do problema no espaço de busca. Os cromossomos são formados por genes organizados em uma sequência linear, que podem ter um determinado valor entre vários possíveis, chamados de alelos. Cada gene controla a herança genética das características dos indivíduos e possui um local fixo no cromossomo denominado *locus*. Outros termos importantes a serem considerados são genoma, genótipo e fenótipo (MICHALEWICZ, 1996).

O genótipo representa a estrutura do cromossomo codificado, e pode ser identificado

no AG com o termo estrutura; cada genótipo representa uma potencial solução para um determinado problema. Fenótipo corresponde à interação do conteúdo genético com o ambiente, ou seja, o cromossomo decodificado. A Tabela 2.3 apresenta a analogia da terminologia utilizada na biologia e no AG (para mais detalhes sobre os algoritmos genéticos ver Linden (2012)).

**Tabela 2.3:** *Terminologia referente a analogia entre a Biologia e o AG.*

| **Biologia** | **Algoritmos Genéticos** |
|---|---|
| Cromossomo | Indivíduo ou cromossomo |
| Gene | Característica |
| Alelo | Valor |
| *Locus* | Posição de cada gene |
| Genótipo | Estrutura do cromossomo |
| Fenótipo | Conjunto de parâmetros |
| Geração | Ciclo ou iteração |

Fonte : Linden (2012).

No que diz respeito ao JSSP, um dos pontos mais difíceis de entender quando se trabalha com AG é o mapeamento genótipo-fenótipo. Devido às complexas restrições envolvidas no problema, conforme apresentado na Seção 2.2.2, os métodos de representação são complexos e não permitem uma visão direta do que está acontecendo no nível do fenótipo. Por exemplo, na representação baseada em *job* uma medida de alta diversidade para o genótipo (código) não necessariamente implica em uma medida de alta diversidade do fenótipo (sequenciamento) (BRIZUELA; SANNOMIYA, 1999).

### 2.4.1 Algoritmo Genético Clássico

A estrutura de um AG clássico pode ser descrita da seguinte forma: durante as gerações, um AG mantém uma população de indivíduos (soluções potenciais). Cada indivíduo é avaliado de acordo com uma função de aptidão (função *fitness*) e uma nova população é então formada selecionando-se os indivíduos mais aptos. Alguns indivíduos dessa população sofrem alterações por meio de cruzamento e mutação para formar novos indivíduos e esse processo é repetido até que o número máximo de gerações seja atingido ou algum outro critério de parada preestabelecido. A Figura 2.7 apresenta o fluxograma do processo descrito.

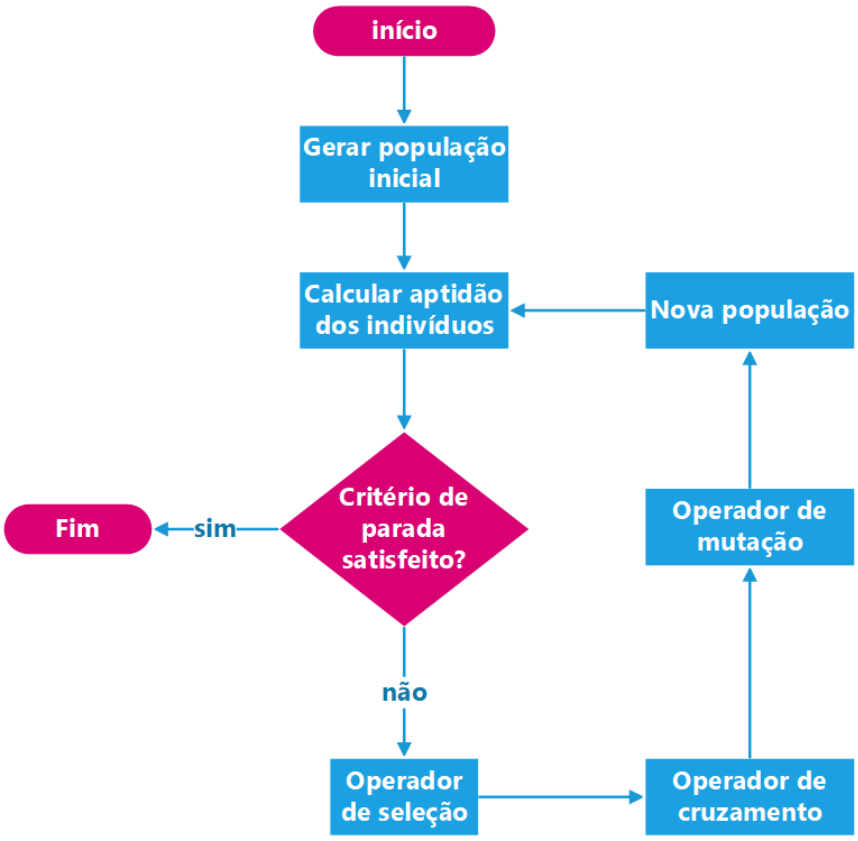



**Figura 2.7:** *Fluxograma de um Algoritmo Genético clássico.*

Fonte : A autora.

De uma forma geral, os algoritmos genéticos têm cinco componentes (MICHALEWICZ, 1996):

a) uma representação das possíveis soluções do problema;

b) uma forma de gerar uma população inicial das soluções;

c) uma função de avaliação para classificar as soluções em termos de sua aptidão;

d) operadores genéticos que alterem a composição genética dos filhos durante a reprodução;

e) valores dos parâmetros que o AG utiliza tais como: tamanho da população, número de gerações, taxa de mutação, método de seleção, entre outros.

A representação das soluções candidatas no espaço de busca de um problema de otimização define a estrutura do cromossomo a ser manipulado pelo AG, que por sua vez, depende do tipo de problema e do que, essencialmente, se deseja manipular geneticamente. Nos AGs as operações não são realizadas diretamente sobre as soluções candidatas, mas sobre uma codificação das mesmas. Um cromossomo, normalmente um vetor, pode representar números binários, números reais, números inteiros, permutação de símbolos,

símbolos repetidos, dentre outras. A Figura 2.8 apresenta a estrutura de um cromossomo no AG.

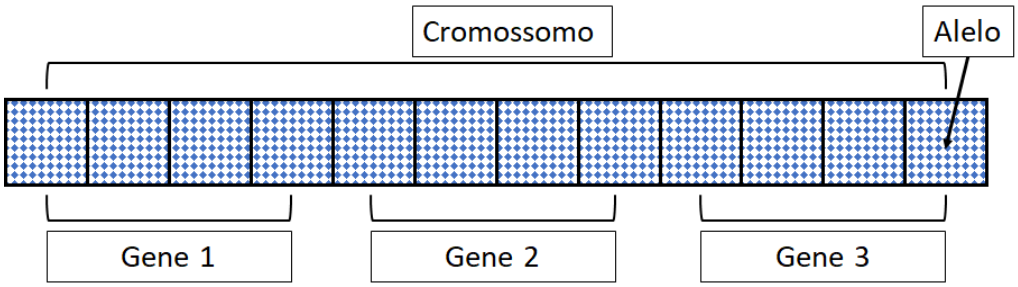



**Figura 2.8:** *Representação da estrutura de um cromossomo no Algoritmo Genético.*
Fonte : A autora.

### 2.4.2 Operadores do Algoritmo Genético

A cada geração da população no AG, os indivíduos se reproduzem, sobrevivem ou desaparecem de uma população por meio dos operadores genéticos. A geração de indivíduos sucessores é determinada por um conjunto de operadores que recombinam e transformam os indivíduos da população corrente para a próxima geração e, a forma mais simples do AG para evoluir uma população de indivíduos envolve três tipos de operadores: seleção, cruzamento ou *crossover* e mutação (LIMA; ARAÚJO, 2018; MITCHELL, 1997).

O operador de seleção, como o próprio nome diz, seleciona os indivíduos da população para reprodução e determina quantas vezes um indivíduo será reproduzido na geração.

Análogo ao processo de seleção natural, a cada geração o AG seleciona os melhores indivíduos (cromossomos pais) da população para gerar novos filhos (cromossomos filhos) por meio dos operadores de cruzamento e mutação. Geralmente, a capacidade de um indivíduo ser selecionado para reprodução depende de sua aptidão e, os indivíduos com melhor aptidão são reproduzidos com mais frequência do que os outros, porém não se pode descartar os indivíduos menos aptos, pois eles podem conter características genéticas que favoreçam a geração de um indivíduo que pode ser a melhor solução do problema. Caso seja utilizado somente os melhores indivíduos para se reproduzirem, a população tenderá a ter indivíduos cada vez mais semelhantes causando o efeito denominado de convergência genética que se dá pela falta de diversidade na população (LINDEN, 2012).

A forma mais comum de cruzamento (*crossover*) é a seleção de dois "cromossomos pai" pelo operador de seleção, para gerar dois "cromossomos filho". O cruzamento de um ponto é o método mais simples no qual, escolhe-se um ponto de corte nos cromossomos, isto é, o ponto de separação entre cada um dos genes que compõem o material genético de cada pai. O primeiro filho é composto pela junção da parte do primeiro pai à esquerda do ponto de corte com a parte da direito do ponto de corte do segundo pai. O segundo filho

é composto pela junção da parte do segundo pai à direita do ponto de corte com a parte do primeiro pai à esquerda do ponto de corte. A Figura 2.9 ilustra o processo descrito para o cruzamento de dois pais e a geração de dois filhos considerando o ponto de corte na segunda posição do cromossomo.

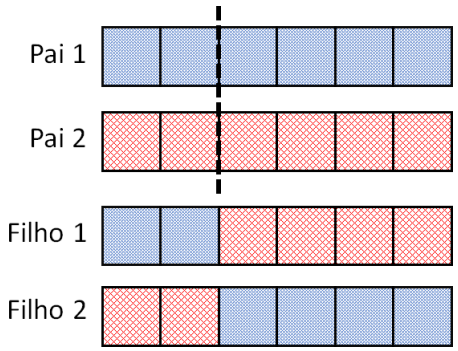



**Figura 2.9:** *Par de cromossomos pais com ponto de corte na segunda posição para gerar dois novos cromossomos filhos por meio do operador de cruzamento.*

Fonte : A autora.

O cruzamento de dois pontos de corte é similar ao de um ponto, neste caso são selecionados dois pontos. O primeiro filho é formado pelas partes do primeiro pai que estão fora dos pontos de corte e pela parte do segundo pai que está entre os dois pontos de corte. O segundo filho tem sua formação semelhante ao primeiro filho, porém invertendo-se os pais (LINDEN, 2012).

A operação de mutação é um importante mecanismo para preservação da diversidade genética da população onde pequenas trocas aleatórias são efetuadas nos cromossomos selecionados. A operação de mutação modifica aleatoriamente os genes de um indivíduo de acordo com uma baixa probabilidade conhecida como taxa de mutação que é um dos parâmetros que compõe o AG.

Depois da geração dos filhos na etapa de cruzamento, aplica-se o operador de mutação que, geralmente está associado uma probabilidade baixa à taxa de mutação, como por exemplo, 0,5%, e então é gerado um número aleatório entre 0 e 1. Se o número for menor do que a taxa de mutação definida, então, o operador de mutação atua no gene do cromossomo alterando o seu valor aleatoriamente. A Figura 2.10 ilustra a mutação do último alelo no cromossomo do primeiro filho gerado no processo anterior.

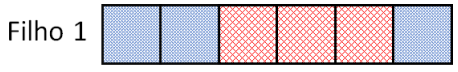



**Figura 2.10:** *Operação de mutação no último alelo do cromossomo.*

Fonte : A autora.

## 2.5 Biased Random-Key Genetic Algorithm

O *Biased Random-Key Genetic Algorithm* (BRKGA) é uma metaheurística evolucionária da classe dos Algoritmos Genéticos proposta por Toso e Resende (2015) para encontrar soluções para problemas de otimização combinatória. O método é baseado no *Random-Key Genetic Algorithm* (RKGA) introduzido por Bean (1994) no qual as soluções podem ser representadas como vetores de permutação como é o caso dos problemas combinatórios envolvendo sequenciamento, incluindo os problemas de sequenciamento de múltiplas máquinas, problemas de alocação de recursos e *quadratic assignment problem*, por exemplo.

O BRKGA tem três características principais que especializam dos Algoritmos Genéticos (TOSO; RESENDE, 2015):

- uma codificação de cromossomo utilizando um vetor de $nk$ chaves aleatórias ou alelos no intervalo de $[0, 1)$, no qual o valor de $nk$ depende da instância do problema;
- um processo evolutivo definido adotando-se um cruzamento (*crossover*) uniforme parametrizado para gerar os descendentes e assim evoluir a população;
- a introdução de novos cromossomos denominados de mutantes no lugar do operador de mutação normalmente encontrado nos algoritmos evolucionários.

O BRKGA de forma geral, utiliza um vetor $X$ de chaves aleatórias de dimensão $nk$ dentro do intervalo $[0, 1)$ para a representação do cromossomo, o qual é decodificado e mapeado por um decodificador em uma solução do problema de otimização. O *fitness* da solução é calculado por meio da função objetivo, que é associado a cada indivíduo e, o valor de $nk$ é dependente da instância do problema. O algoritmo evolui uma população $pop$ de vetores de chaves aleatórias a qual é dividida em um pequeno conjunto denominado *elite* $pe$ que contém os melhores indivíduos e o restante da população compõe o conjunto *não-elite* $(pop - pe)$

O BRKGA inicia com uma população $pop$ contendo exatamente $np$ cromossomos de $nk$ chaves aleatórias e então evolui essa população por $k$ gerações da seguinte forma:

1. cada cromossomo é decodificado para calcular o seu *fitness* por meio da função objetivo e quando o cromossomo é associado ao seu *fitness* ele é chamado de indivíduo;
2. a população $pop$ é particionada em dois grupos: um conjunto denominado de elite que contém $pe$ indivíduos com os melhores valores do *fitness* e, um conjunto denominado de não-elite que contém os indivíduos remanescentes $(pop - pe)$;
3. a população a cada geração consistirá então de:

(a) $pe$ indivíduos do conjunto elite da geração corrente que serão copiados para a próxima geração $k+1$;

(b) $pm$ cromossomos gerados aleatoriamente denominados de mutantes;

(c) $po = pop - pm - pe$ cromossomos evoluídos (descendentes) que são produzidos pelo cruzamento de dois pais da geração corrente, selecionados aleatoriamente, com substituição, sendo um pai do conjunto elite e outro do conjunto de não-elite.

O fluxograma do algoritmo BRKGA é apresentado na Figura 2.11.

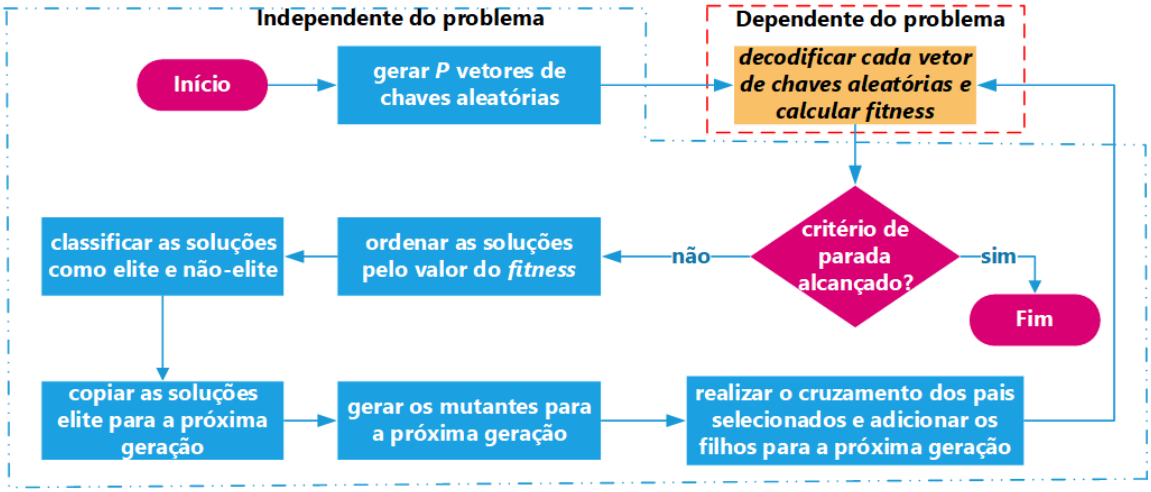



**Figura 2.11:** *Fluxograma do Biased Random-Key Genetic Algorithm - BRKGA*

Fonte : Adaptado de Gonçalves e Resende (2011) e Gonçalves e Resende (2014).

O processo de evolução da população é apresentado na Figura 2.12. No lado esquerdo dessa Figura está a população corrente. Após a ordenação de todos os indivíduos com base nos valores do *fitness*, os melhores indivíduos encontram-se na partição ELITE, e os outros estão na partição denominada NÃO-ELITE. Os indivíduos da população elite são copiados sem alterações para a partição denominada TOP para a próxima geração (lado direito da figura). Os indivíduos mutantes são gerados aleatoriamente e inseridos na partição BOT da nova geração. O restante da população para a próxima geração é formada pela operação de cruzamento (SILVA; RESENDE, 2013). A característica importante das chaves aleatórias é que todos os filhos gerados pelo cruzamento são soluções factíveis (CHAVES *et al.*, 2016).

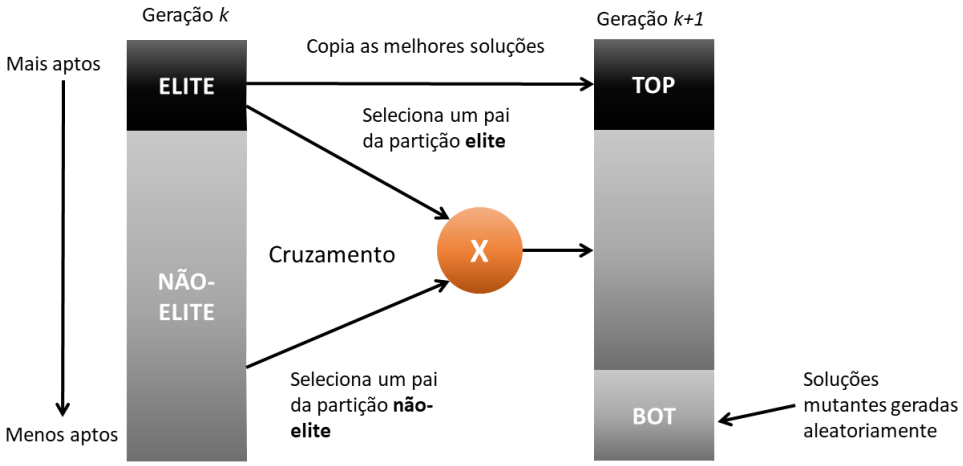



**Figura 2.12:** *Processo da evolução da população do BRKGA*

Fonte : Adaptado de Gonçalves e Resende (2011), Silva *et al.* (2013).

A Figura 2.13 apresenta o processo de cruzamento de dois vetores de chaves aleatórias com nove elementos cada. O Cromossomo 1 representa o indivíduo (Pai 1) do conjunto elite e, o Cromossomo 2 (Pai 2) do conjunto não-elite. No exemplo apresentado (Figura 2.13), a probabilidade de um filho herdar um componente do pai elite é de 70% e, do pai não-elite é de 30%. Um número aleatório é gerado no intervalo entre $(0, 1]$. Se o resultado é menor ou igual a 70%, então o filho herdará o componente do pai elite senão, o filho herdará o componente do pai não-elite. Esse processo é realizado para compor a próxima população e, os indivíduos são avaliados de acordo com a função objetivo (SILVA; RESENDE, 2013).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Cromossomo Pai 1 | 0,50934 | 0,674434 | 0,145577 | 0,797062 | 0,817017 | 0,713488 | 0,42521 | 0,192037 | 0,648522 |
| Cromossomo Pai 2 | 0,648522 | 0,756746 | 0,192037 | 0,713488 | 0,764749 | 0,797062 | 0,90824 | 0,253094 | 0,0841146 |
| Número Aleatório | 0,58 | 0,89 | 0,68 | 0,92 | 0,38 | 0,75 | 0,83 | 0,32 | 0,48 |
| Probabilidade < 0,7 | < | > | < | > | < | > | > | < | < |
| Cromossomo Filho | 0,50934 | 0,756746 | 0,145577 | 0,713488 | 0,817017 | 0,797062 | 0,90824 | 0,192037 | 0,648522 |



**Figura 2.13:** *Processo de cruzamento uniforme parametrizado entre dois pais para geração de um filho*

Fonte : Adaptado de Gonçalves e Resende (2011), Silva *et al.* (2013).

### 2.5.1 Codificação e Decodificação do BRKGA

O BRKGA trabalha com vetores de chaves aleatórias para a representação do cromossomo o qual deve ser codificado e decodificado. Por exemplo, sendo o vetor $X$ codificado pelo BRKGA apresentado a seguir e, os valores do seu índice que são utilizados para realizar a decodificação.

| *índice* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| $X =$ | [0,50934 | 0,756746 | 0,145577 | 0,713488 | 0,817017 | 0,797062 | 0,90824 | 0,192037 | 0,648522] |

Então, o vetor $X$ é ordenado em ordem crescente, e os índices são desordenados conforme apresentado a seguir.

| *índice* | 2 | 7 | 0 | 8 | 3 | 1 | 5 | 4 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| $X =$ | [0,145577 | 0,192037 | 0,50934 | 0,648522 | 0,713488 | 0,756746 | 0,797062 | 0,817017 | 0,90824] |

Assim, para a codificação de uma representação, utiliza-se os valores do índice do vetor depois da etapa anterior, e então, esses valores serão armazenados em um outro vetor $Y$ que deverá ser decodificado para a representação do problema.

$$Y = [2 \quad 7 \quad 0 \quad 8 \quad 3 \quad 1 \quad 5 \quad 4 \quad 6]$$

Um exemplo da decodificação do vetor $Y$ para a representação do JSSP é apresentado no Capítulo 4 na Seção 4.6.1.

### 2.5.2 Diversidade Populacional

O processo de reprodução e mutação são aplicados em indivíduos selecionados da população dos AG's. Os indivíduos com maior aptidão são selecionados mais frequentemente do que os menos aptos, de forma que as boas características dos mais aptos começam a predominar dentro da nova população de soluções. No entanto, os indivíduos menos aptos não devem ser descartados do processo de cruzamento para que não ocorra uma rápida convergência genética de todas as soluções para um mesmo conjunto de características, e evitaria uma busca mais ampla pelo espaço de soluções. A convergência genética implica em uma população com baixa diversidade genética que, por possuir genes similares, não consegue evoluir, a não ser pela ocorrência de mutações. Isso pode ser considerado como a perda da diversidade e, quanto maior for essa perda, mais rápida será a convergência genética o que faz com que o AG, depois de um certo número de gerações fique estagnado (LINDEN, 2012).

A diversidade populacional desempenha um papel importante no mecanismo de adaptação dos AG's para que se tenha sucesso no processo de busca. A convergência prematura do algoritmo acontece porque, durante o processo evolutivo a população é facilmente preenchida com indivíduos mais semelhantes reduzindo assim a diversidade populacional

(BRIZUELA; SANNOMIYA, 1999; KUVAT; ADAR, 2016; JI; WANG, 2012; ZHANG; CHIONG, 2016).

Os AG's são formados por indivíduos que representam diferentes regiões do espaço de busca e, eles podem encontrar a solução em diferentes pontos no espaço ao mesmo tempo. No entanto, à medida que a população evolui à cada geração, se as soluções encontradas estiverem fora da região onde se encontra a melhor solução, torna-se difícil obter bons resultados, portanto, se faz necessário manter a diversidade populacional (KUVAT; ADAR, 2016).

### 2.5.3 Intensificação e Diversificação

As abordagens metaheurísticas devem ser projetadas com o objetivo de explorar um espaço de busca de forma efetiva e eficientemente. A busca realizada pela metaheurística deve explorar intensamente áreas do espaço de busca com soluções de alta qualidade e, mover-ser para áreas inexploradas do espaço de busca quando necessário (BLUM; ROLI, 2003).

Os conceitos para se atingir esses objetivos são chamados de intensificação e diversificação os quais são utilizados na Busca Tabu (GLOVER; LAGUNA, 1997). Em outras áreas, como por exemplo, na Computação Evolucionária, os conceitos relacionados são descritos como *exploitation* relacionado à intensificação e, *exploration* que está relacionado à diversificação, porém com um significado mais restrito (BLUM; ROLI, 2003).

O uso dos termos diversificação e intensificação em seu significado inicial tornou-se cada vez mais aceito por todo o campo das metaheurísticas. O equilíbrio entre diversificação e intensificação é importante para identificar rapidamente regiões no espaço de busca de alta qualidade (BLUM; ROLI, 2003; NERI; COTTA, 2012). Um algoritmo de busca deve encontrar um equilíbrio entre intensificação e diversificação que às vezes são conflitantes (LOZANO; GARCÍA-MARTÍNEZ, 2010).

A diversificação geralmente refere-se à capacidade de visitar muitas e diferentes regiões do espaço de busca, enquanto que a intensificação refere-se à capacidade de obter soluções de alta qualidade nessas regiões (LOZANO; GARCÍA-MARTÍNEZ, 2010).

Portanto, resumidamente, diversificação é uma busca realizada em diferentes áreas do espaço de busca para encontrar as regiões mais promissoras e, intensificação é a busca na vizinhança de um indivíduo para produzir melhores soluções (GAO *et al.*, 2011; MORITA *et al.*, 2016).

As metaheurísticas mais clássicas têm vários componentes para intensificação e diversificação. Blum e Roli (2003) definem um componente de intensificação e diversificação como qualquer componente algorítmico ou funcional que tenha efeito de intensificação e/ou diversificação no processo de busca, como por exemplo, os operadores genético, perturbações de distribuição de probabilidade, a utilização de listas tabu, mudanças na fun-

ção objetivo, etc. Assim os componentes de intensificação e diversificação são operadores, ações ou estratégias das metaheurística (LOZANO; GARCÍA-MARTÍNEZ, 2010).

## 2.6 Análise de Componentes Principais

A Análise de Componentes Principais (do inglês, *Principal Component Analysis* - PCA) é uma técnica da análise multivariada utilizada na redução da dimensão de um conjunto de dados que contém um grande número de variáveis inter-relacionadas, enquanto mantém o máximo possível a variação presente nesse conjunto de dados (JOLLIFFE, 2002). Essa redução é alcançada transformando-se os dados para um novo conjunto de variáveis, os Componentes Principais (CP), que não estão correlacionados, e são ordenados (classificados) de tal forma que os primeiros CP retêm a maior parte da variação presente em todas as variáveis originais.

A PCA permite identificar padrões nos dados destacando-se suas semelhanças e diferenças e, a partir desses padrões encontrados é possível comprimi-los e expressá-los em um subespaço de dimensão reduzida sem acarretar muita perda de informação (SADOWSKI; NIKOO; NIKOO, 2015; SANTO, 2012; SANTO; PEREIRA; JÚNIOR, 2012).

Essa técnica tem sido utiliza com sucesso na redução da dimensão de dados em várias áreas do conhecimento, inclusive em abordagens hibridas, como, por exemplo, em reconhecimento de faces Hussein Al-Arashi, Ibrahim e Azmin Suandi (2014) utilizaram PCA com AG; na química Ghaedi *et al.* (2014) utilizaram PCA, redes neurais artificiais e AG; Vanhatalo, Kulahci e Bergquist (2017) utilizaram em monitoramento de processos estatísticos; PCA com rede neural artificial para previsão de preços de ações utilizado por Zahedi e Rounaghi (2015); em compressão de imagens por Santo (2012), entre outros.

Matematicamente, a PCA projeta um conjunto de dados $X$ em uma base ortogonal no $\mathbb{R}^N$, a qual é definida como um conjunto de $p$ autovetores (*eigenvectors*) $e_i \in \mathbb{R}^N$, $i = 1, ..., p$, da matriz de covariância de $X$. Essa base ortogonal é orientada nas direções que fornecem a variância máxima de $X \in \mathbb{R}^N$, com a finalidade de representar as informações mais relevantes.

O princípio da redução da dimensionalidade é a representação do conjunto de dados $X$ em termos dos autovetores da matriz de covariância, os quais são chamados de componentes principais. A fim de realizar a redução da dimensionalidade, o conjunto de dados é representado como uma matriz real $U_{n \times N}$, na qual $n$ e $N$ são, o número de linhas e colunas da matriz, respectivamente.

A matriz de covariância de $U$ é calculada, assim como os seus autovalores e autovetores correspondentes. Esses autovetores formam um conjunto de vetores linearmente independentes, ou seja, uma base $\phi_i, i = 1, ..., n$, que consiste em um novo sistema de eixos. Por fim, para realizar a redução de dimensionalidade, as linhas de $U$ são projetadas na base formada pelos $p$ autovetores relacionados aos maiores autovalores $\lambda_p$. As coordenadas de

$U$ projetadas neste subespaço $p$-dimensional são especificadas como $U_{\phi_1}, U_{\phi_2}, ..., U_{\phi_n}$.

### 2.6.1 Seleção de características com Análise de Componentes Principais

Como resultado do processo apresentado na Seção 2.6, a PCA retorna uma projeção em um novo espaço que é diferente dos dados originais.

Em algumas situações no entanto é necessário selecionar um subconjunto das variáveis originais, ao invés de representar os dados em um subespaço. Isso é feito usualmente para propósitos de regressão ou classificação, escolhendo um subconjunto das características originais que contém a maior parte das informações essenciais.

De fato, devido à sua eficiência, a PCA tem sido utilizada repetidamente como base para abordagens de seleção de variáveis em grandes bancos de dados (JOLLIFFE, 1972; JOLLIFFE, 1973; BAIR *et al.*, 2006). Em Jolliffe (1973), por exemplo, o autor apresentou quatro abordagens baseadas em PCA para seleção de variáveis e concluiu que o método é bem sucedido em produzir bons subconjuntos.

Importante destacar, no entanto, que geralmente a PCA é aplicada diretamente ao conjunto de dados que se deseja reduzir. Recentemente, de Oliveira Jr *et al.* (2015) propuseram uma versão da PCA, a qual chamaram supervisionada, que considera a aplicação de um filtro para pré-selecionar os atributos relevantes e, posteriormente, usar a PCA para remover atributos correlacionados. Nesse processo de filtragem os autores calculam um coeficiente de regressão padronizado para cada variável e usa essa informação para uma pré-redução, com posterior aplicação da PCA nos dados já reduzidos. O termo supervisionado vem do fato de que os coeficientes de regressão são calculados considerando a relação entre as variáveis de entrada e saída.

## 2.7 Método Path-relinking

O método *path-relinking* (PR – religação de caminhos ou reconexão de caminhos) é um procedimento metaheurístico evolucionário proposto por Glover (1997) como uma estratégia de intensificação para explorar trajetórias ou caminhos que conectam soluções elite obtidas pelo algoritmo *tabu search* ou *scatter search* (AIEX; BINATO; RESENDE, 2003).

O procedimento de *relinking* é utilizado para gerar novas soluções explorando-se trajetórias/caminhos que conectam soluções de alta qualidade (PENG; LÜ; CHENG, 2015). O *path relinking* geralmente é realizado entre duas soluções, sendo uma solução inicial e uma solução guia. Um ou mais caminhos no espaço de busca conectando essas soluções são explorados na busca por melhores soluções (RIBEIRO; RESENDE, 2012) e, a construção das trajetórias é tipicamente feita por meio de um processo de transformação que, iterativamente, converte duas soluções em uma outra (BIERWIRTH; KUHPFAHL, 2017).

O PR opera em um conjunto de soluções denominado *conjunto de referência* (do inglês, RS), utilizando estratégias para selecionar, combinar, melhorar e gerar novas e melhores soluções desse conjunto (PÉREZ; RODRÍGUEZ; VEGA, 2004).

O RS é uma coleção de soluções com maior representatividade do problema, selecionadas de um conjunto inicial de soluções que evoluem para exploração da região factível, por meio de intensificação e diversificação, o qual focam respectivamente a busca em regiões mais promissoras à partir de soluções anteriores e guiam a busca para explorar novas regiões ainda não exploradas.

Os mecanismos de diversificação e intensificação são aplicados pelo PR para atualizar o RS no intuito de guiar as buscas em direção a solução ótima do problema. A diversificação do PR é geralmente aplicada pela atualização de um subconjunto do RS com soluções que não estão no RS e que maximizam a distância até a solução mais próximo do RS. As soluções restantes do RS são atualizadas por meio de intensificação (PÉREZ; RODRÍGUEZ; VEGA, 2004).

Geralmente, a intensificação é executada combinando soluções diferentes para criar novas soluções iniciais para os procedimentos de busca local. O PR utiliza um mecanismo para combinar soluções que criam um ou mais vizinhos para passar de uma solução para outra, na forma de realizar as buscas na vizinhança.

Os componentes do *path relinking* em geral são as regras para construir um conjunto de referência, as regras para a escolha da solução inicial e da solução guia e, uma estrutura de vizinhança para movimentação ao longo dos caminhos (ZHANG *et al.*, 2016).

O PR pode ser visto como uma estratégia de busca local a qual é aplicada em uma solução inicial e somente um conjunto limitado de movimentos podem ser realizados. Várias abordagens têm sido utilizada para implementação do *path relinking* (RESENDE; RIBEIRO, 2005):

- *periodical relinking* : o *path relinking* não é sistematicamente aplicado, mas apenas periodicamente;
- *forward relinking*: o *path relinking* é aplicado usando a pior entre a solução inicial e a solução guia como a solução inicial e a outro como a solução guia;
- *backward relinking*: os papéis da solução inicial e da solução guia são trocados, o *path relinking* é aplicado usando o melhor entre as duas soluções como a solução inicial e o outro como a solução de guia;
- *back e forward relinking*: duas trajetórias diferentes são exploradas, a primeira usando a solução inicial e a segunda utilizando a solução guia nessa função;
- *mixed relinking*: dois caminhos são explorados simultaneamente, o primeiro partindo da solução inicial e o segundo da solução guia, até que eles se encontram em uma solução intermediária equidistante das soluções inicial e guia;

- *randomized relinking*: em vez de selecionar o melhor movimento ainda não selecionado, seleciona-se aleatoriamente um dentre uma lista de candidatos com os movimentos mais promissores no caminho que está sendo investigado;
- *truncated relinking*: a trajetória completa entre as soluções inicial e guia não é investigada, mas apenas parte dela.

Capítulo 3

# Revisão da literatura

Os trabalhos elencados neste Capítulo abordam a utilização das técnicas metaheurísticas aos problemas de sequenciamento da produção, principalmente o JSSP e suas variações. São apresentados e discutidos também trabalhos que tratam da análise do *fitness landscape*, seja para ampliar o conhecimento sobre o espaço de soluções desses problemas ou para desenvolver algoritmos mais eficientes.

## 3.1 Aplicação de técnicas metaheurísticas ao JSSP

Em virtude do sucesso alcançado pelas técnicas metaheurísticas, elas têm sido utilizadas largamente, combinadas com outras técnicas ou não, nos mais diversos tipos de problemas de sequenciamento da produção. No contexto do JSSP destacam-se algumas abordagens como, por exemplo, *tabu search*, *simulated annealing*, colônia de formigas e, especialmente, o algoritmo genético. Em geral, as metaheurísticas populacionais são utilizadas em conjunto com outras heurísticas/metaheurísticas de busca local, visando atender as demandas de diversificação e intensificação impostas pelas características do *fitness landscape* do JSSP.

O método *tabu search* (TS) permaneceu por vários anos no estado da arte dos problemas de *scheduling* (WATSON; HOWE; WHITLEY, 2006) e, ainda atualmente, permanece como uma das abordagens mais eficientes para esses problemas (TAMSSAOUET; DAUZÈRE-PÉRÈS; YUGMA, 2018). Consequentemente, o método tem sido frequentemente utilizado, como método de solução ou compondo abordagens híbridas como busca local. O TS foi utilizado por Zhang *et al.* (2007) com uma nova estrutura de vizinhança proposta pelos autores e aplicado para resolver o JSSP. Um algoritmo híbrido *Island Model Genetic Algorithm* com *Tabu Search* foi proposto por Kurdi (2015) com objetivo de minimizar o *makespan*. Com a finalidade de melhorar a eficácia do *Island Model Genectic Algorithm*, o autor propôs uma nova estratégia naturalmente inspirada na fase de auto adaptação que é capaz de atingir um melhor equilíbrio entre diversificação e intensificação do processo de busca. Na estratégia da fase de auto-adaptação proposta, os melhores indivíduos são selecionados para realizar uma busca local utilizando o TS, e os piores indivíduos são selecionados para realizar uma busca global utilizando uma combinação aleatória de três operadores clássicos de mutação. O algoritmo proposto foi testado em 76 instâncias de problemas com a estratégia de auto-adaptação e, sem auto-adaptação. Também foi comparado com outros quinze algoritmos recentemente relatados na literatura. Os resultados computacionais apresentaram as melhorias alcançadas com estratégia de auto-adaptação proposta, e mostraram a superioridade do algoritmo proposto sobre treze dos trabalhos comparados em termos de qualidade da solução.

Um algoritmo que incorpora o procedimento do *Tabu Search* em um *framework* do *path relinking* para gerar soluções para o JSSP foi apresentado por Peng, Lü e Cheng (2015). O algoritmo denominado *Tabu Search/Path Relinking* (TS/PR) compreende vários recursos distintos, tais como um procedimento de religamento específico na construção de um caminho que vincula a solução inicial e uma solução guia, e um mecanismo de determinação de solução referência com base em dois tipos de métodos de melhoria.

O *Tabu Search* também foi aplicado para melhorar uma solução inicial gerada pela heurística construtiva denominada *Shifting Bottleneck Procedure* na proposta de Mellado, Cubillos e Cabrera (2016) para minimizar o *makespan* do JSSP.

Vale frisar, no entanto, que o TS parece ter atingido o nível de saturação, uma vez que seu funcionamento já é quase que completamente compreendido (WATSON; HOWE; WHITLEY, 2006). Assim, muitas pesquisas consideraram outras abordagens, frequentemente com uso conjunto de técnicas de busca local.

Gao *et al.* (2011) propuseram uma melhoria no *Memetic Algorithm* para minimizar o *makespan* do JSSP. Os autores implementaram uma nova busca local que troca sistematicamente a vizinhança das soluções para escapar de ótimos locais e, as estruturas de vizinhança são com base no caminho critico. Esse mesmo grupo de pesquisa desenvolveu um algoritmo híbrido *Particle Swarm Optimization* e *Variable Neighborhood Search* para resolver o JSSP (GAO *et al.*, 2015). Os autores propuseram um método de avaliação das estruturas de vizinhança com base em um modelo logístico para orientar o processo de seleção dessas estruturas no JSSP.

O algoritmo *Particle Swarm Optimization* também foi utilizado por Sha e Lin (2010) para resolver o *job shop scheduling problem* com multi-objetivo para minimizar o *makespan*, o atraso total (*total tardiness*) e o tempo total de ociosidade da máquina.

Com o objetivo de minimizar o makespan Zhao *et al.* (2015) apresentaram um novo algoritmo inteligente denominado *Shuffled Complex Evolution*. O mecanismo de mapeamento de sequência é utilizado para alterar as variáveis no domínio contínuo para variáveis discretas no problema de otimização combinatória. A sequência, a qual tem como base a permutação dos *jobs* é adaptada para o mecanismo de codificação, e o de inserção de sequência para a decodificação. De acordo com os autores, o algoritmo básico *Shuffled Complex Evolution* tem desvantagens por apresentar soluções de baixa qualidade e menor taxa de convergência, então uma nova estratégia foi utilizada para alterar a evolução do indivíduo no algoritmo básico *Shuffled Complex Evolution*. A estratégia torna o novo indivíduo mais próximo do melhor na população corrente. Os resultados obtidos com o algoritmo *Shuffled Complex Evolution* melhorado denominado *Improved SCE Algorithm* mostram que o algoritmo melhorado é eficaz para o problema *Job Shop Scheduling*.

Abordagens recentes envolvendo *Artificial Bee Colony Algorithm* e *Simulated Annealing* podem ser vistas, respectivamente, em (ASADZADEH, 2016; AKRAM; KAMAL; ZEB, 2016). Asadzadeh (2016) propôs um algoritmo paralelo *Parallel Artificial Bee Co-*

*lony Algorithm* para resolver o JSSP. No algoritmo proposto, o *Artificial Bee Colony Algorithm* consiste em várias colônias que se localizam em diferentes *hosts* da rede e são realizados em várias colônias de forma paralela. A comunicação entre colônias é realizada por meio da troca de migrantes. Uma estratégia de migração dinâmica é utilizada para determinar quando uma colônia deve se comunicar com seus vizinhos. Akram, Kamal e Zeb (2016) propuseram uma nova abordagem híbrida que combina o algoritmo *Fast Simulated Annealing* com *Quenching*. Nessa abordagem o *Fast Simulated Annealing* é utilizado tanto para realizar busca global para escapar de ótimos locais. O *Quenching* é utilizado para realizar a busca local na vizinhança próxima da solução corrente.

Em resumo, o que se observa é que, dada a complexidade do problema, as abordagens mais eficientes são baseadas na combinação de diferentes técnicas, sendo o uso da busca local e de mecanismos de diversificação e intensificação que são ingredientes primordiais. Isso também é verdade para as abordagens com base no algoritmo genético.

### 3.1.1 Abordagens baseadas no Algoritmo Genético

Ao contrário do *Tabu Search*, o desempenho do Algoritmo Genético (AG) para o JSSP parece ser ainda pouco compreendido, o que deixa espaço para novas investigações. Como algumas das abordagens mais significativas nesse contexto cita-se os trabalhos de Gonçalves, Mendes e Resende (2005), Lei (2011), Asadzadeh (2015), El-Desoky *et al.* (2016) e Kurdi (2016).

Gonçalves, Mendes e Resende (2005) e Gonçalves e Resende (2014) utilizaram o algoritmo genético de chaves aleatórias *Biased Random-Key Genetic Algorithm* (BRKGA) para minimizar o *makespan* do *job shop scheduling problem*. O algoritmo foi utilizado para gerar regras de prioridades das sequências. Também foi implementado um algoritmo de busca local para efetuar troca de operações no caminho crítico. O BRKGA também foi utilizado pelos autores Silva *et al.* (2013) para encontrar soluções aproximadas em problemas de otimização global contínuo com restrições não lineares.

Essas abordagens comprovam que o BRKGA é capaz de produzir bons resultados para o JSSP e outros problemas semelhantes, ainda que precise ser combinado com outros métodos. Damm, Resende e Ronconi (2016), por exemplo, abordaram um problema frequente em empresas de prestação de serviços que é o agendamento de técnicos de campo. Esse problema considera a atribuição de um conjunto de tarefas a um grupo de técnicos. As tarefas estão em locais diferentes dentro de uma cidade, com diferentes janelas de tempo, prioridades e tempos de processamento. Os técnicos têm diferentes habilidades e horas de trabalho. O objetivo principal é maximizar a soma dos valores de prioridade associados às tarefas executadas a cada dia. Devido à complexidade desse problema, são desenvolvidas heurísticas construtivas que exploram características específicas do problema. Os autores propuseram um BRKGA customizado e apresentaram testes computacionais com 1040

instâncias. As heurísticas construtivas superaram as heurísticas da literatura em 90% das instâncias. Em um estudo comparativo com soluções ótimas obtidas para problemas de pequeno porte, o BRKGA atingiu 99% dos valores ótimos; para problemas de médio e grande porte, o BRKGA forneceu soluções que estão em média 3,6% abaixo dos limites superiores.

Asadzadeh (2015) propôs um AG com busca local baseada em agente (*Agent-based Local Search Genetic Algorithm*) para resolver o JSSP. Um sistema contendo vários agentes foi desenvolvido para a busca local do AG. Nessa abordagem, dois procedimentos de busca local são aplicados para aumentar a eficiência do algoritmo. De acordo com o autor, a abordagem é eficaz na descoberta de soluções ótimas ou próximas do ótimo quando aplicada a várias instâncias do JSSP. O algoritmo proposto é mais robusto do que um AG clássico para os valores de *makespan*. O autor concluiu que o algoritmo proposto é eficaz tanto do ponto de vista da qualidade das soluções quanto na robustez do algoritmo.

El-Desoky *et al.* (2016) propuseram um algoritmo híbrido para o JSSP com base no Algoritmo Genético combinado com busca local. Os autores propuseram um novo método de inicialização da população do AG e os operadores de cruzamento e mutação foram modificados para melhorar a qualidade da solução. Também utilizaram uma busca local com base na estrutura de vizinhança que foi aplicada no resultado do Algoritmo Genético.

Apesar dos bons resultados obtidos por esses trabalhos, o AG enfrenta uma grande dificuldade no JSSP devido a rápida perda de diversidade populacional. Nesse sentido, um novo *Island Model Genetic Algorithm* foi apresentado por Kurdi (2016) para solucionar o JSSP com o objetivo de minimizar o *makespan*. Esse trabalho é uma evolução do seu trabalho anterior (KURDI, 2015). Para melhorar a eficácia do algoritmo clássico *Island Model Genetic Algorithm* foi proposto um novo modelo de evolução e um novo mecanismo de seleção de migração que são capazes de melhorar a diversificação da busca e retardar a convergência prematura.

Entretanto, além de combinar o AG com outras técnicas, os trabalhos em geral não exploram explicitamente as características do espaço de busca no desenvolvimento dos algoritmos. Uma exceção são os trabalhos sobre o BRKGA que exploram e estrutura do espaço de soluções para propor o conceito dos ativos parametrizados como visto em Gonçalves, Mendes e Resende (2005).

As características do espaço de soluções são estudadas em trabalhos que tratam da análise do *fitness landscape*, mas que muitas vezes ficam restritos aos aspectos puramente teóricos.

## 3.2 Abordagens com base na Análise do Fitness Landscape

A investigação da estrutura do *fitness landscape* teve início principalmente considerando problemas de otimização combinatória, de uma forma geral, e o problema do caixeiro

viajante em particular.

Stadler e Schnabl (1992) investigaram o *landscape* do *Traveling Salesman Problems* utilizando a técnica de caminhada aleatória (*random walk*). O *Traveling Salesman Problems* também foi estudado por Tayarani-Najaran e Prügel-Bennett (2016), que investigaram o *fitness landscape* em onze tipos diferentes do problema, o qual difere na construção da matriz de distância. Os tipos diferem em como as distâncias entre as cidades são geradas. Foram estudadas muitas propriedades diferentes do *landscape*. As propriedades escolhidas são potencialmente relevantes para escolher um algoritmo de busca apropriado. A análise inclui um estudo na escala do tempo para alcançar o ótimo local, o número de ótimos locais, a probabilidade esperada para alcançar um ótimo local como uma função do seu *fitness*, o *fitness* esperado encontrado pela busca local e o melhor *fitness*, a probabilidade de alcançar um ótimo global, a distância entre o ótimo local e o ótimo global, o *fitness* esperado em função da distância de um ótimo, suas bacias de atração e uma análise de componentes principais (PCA) dos ótimos locais que mostra a correlação do ótimo local no espaço do componente. Os autores também apresentam como as propriedades dos componentes principais dos ótimos locais mudam de um tipo de problema para outro.

Reidys e Stadler (2002) publicaram uma revisão com foco nas conexões da teoria do *landscape* com a análise combinatória algébrica e a teoria de grafos aleatórios. Reeves e Eremeev (2004) discutem algumas ferramentas estatísticas para entender a natureza do *landscape* combinatório em métodos de busca local.

Tayarani-Najaran e Prügel-Bennett (2015) investigaram o *fitness landscape* do problema de otimização de coloração em grafos para grafos aleatórios densos. Os autores realizaram uma análise sistemática de muitas propriedades do *fitness landscape* para instâncias aleatórias com uma função do problema e do número de cores utilizadas. As propriedades estudadas incluem tanto propriedades estatísticas dos estados, tais como as de distribuição do *fitness* e a autocorrelação, como também as propriedades relacionadas ao ótimo local do problema. Essas propriedades incluem o tempo médio para alcançar o ótimo local, o número de ótimos locais e a probabilidade de alcançar ótimos locais de um determinado custo, a distância média entre ótimos globais e ótimos locais de um determinado custo e o ótimo local mais próximo, o custo esperado em função da distância de uma configuração e da correlação *fitness*-distância e por último, eles apresentam um análise de como um algoritmo de sucesso explora a correlação *fitness*-distância é realizado.

Um dos componentes centrais das técnicas metaheurísticas é a geração de soluções aproximadas em problemas de otimização combinatória, sendo que a técnica *Hill-climbing* (Subida de Encosta) é uma das formas mais simples. Basseur e Goëffon (2015) realizaram um estudo na avaliação dessa técnica no contexto em que movimentos piores não são permitidos, com a finalidade de isolar o aspecto da intensificação das metaheurísticas.Nesse sentido, o principal objetivo do trabalho foi fornecer diretrizes para a escolha do método mais adequados para "escalar" o *fitness landscape* com eficiência.

No que tange especificamente ao JSSP, Bierwirth, Mattfeld e Watson (2004) realizaram uma análise do *fitness landscape* e concluíram que ele é altamente irregular comparando-se com outros problemas de otimização combinatória conhecidos. Essa irregularidade conduz a um desvio, no qual uma caminhada aleatória é direcionada para regiões do espaço de busca com graus mais altos de conectividade. No JSSP, soluções de alta qualidade geralmente possuem alto grau de conectividade. Os autores supõem que a irregularidade deveria auxiliar tanto os caminhos aleatórios quanto os algoritmos de busca local estocásticos para o JSSP, orientando a busca por regiões do *fitness landscape* contendo soluções de alta qualidade.

Streeter e Smith (2005) e, Streeter e Smith (2006) também discutem em seus trabalhos a caracterização do espaço *landscape* de instâncias aleatórias do *job shop scheduling problem* em função da razão entre *job* e máquina $\left(\frac{N}{M}\right)$ e como essa relação pode ajudar à identificar as estruturas do espaço de busca.

Problemas de *scheduling* também foram investigados em Czogalla e Fink (2011). Os autores analisaram o *fitness landscape* do *no-wait (continous) flow shop problem* e contribuíram para a explicação do sucesso das metaheurísticas de busca local baseadas em população. Nesse trabalho os autores examinam a robustez do *landscape* e correlação entre a qualidade de uma solução e sua distância para uma solução ótima. Os resultados apontados pelos autores confirmam a presença de uma grande estrutura de vales igualmente existente em outros problemas de otimização combinatória. A adequação do *landscape* para busca com métodos de busca local e computação evolucionária são discutidas e os experimentos foram validados com dois algoritmos evolucionários, o Algoritmo Genético clássico e o *Discrete Particle Swarm Optmization Algorithm*.

A maioria dos trabalhos, entretanto, mantém o foco nas questões teóricas do *fitness landscape*. Assim, de acordo com Lu *et al.* (2018), os estudos atuais sobre os problemas de sequenciamento baseado em *jobs* (*job-based scheduling problem*) são isolados e faltam referências correspondentes e análises de forma geral. Para resolver esse problema, os autores analisaram as diferenças e similaridades em diferentes problemas desse tipo. Eles propuseram várias medidas de avaliação para explorar as características dos problemas do ponto de vista do espaço de soluções. Essas medidas podem refletir as características do espaço de soluções em termos do grau de similaridade, do grau de precisão, das características da forma, da duração do período e do *fitness* médio. Com base no seu impacto, essas medidas foram divididas em dois grupos: medidas primárias e medidas auxiliares. Além disso, os autores aplicaram essas medidas em problemas de pequeno e grande porte para explorar as características de cada problema e a relevância entre diferentes problemas. Eles concluíram que os espaços de soluções dos problemas de sequenciamento baseados em *jobs* são semelhantes em termos de periodicidade, da característica da forma e a regra de alteração da média do *fitness*.

Apesar de contribuir para ampliar o conhecimento do espaço de soluções, a tradução

de todo o conhecimento em um método de solução plenamente eficiente ainda é um problema. Apresenta-se, em geral, resultados das investigações teóricas e experimentais sobre o *landscape* do espaço de soluções para o JSSP (SMUTNICKI; BOŻEJKO, 2018).

Alguns trabalhos, no entanto, exploraram explicitamente as características do *fitness landscape* no JSSP. Nowicki e Smutnicki (2005a) apresentam algoritmo de aproximação com base na estrutura do *big valley*, algoritmo de busca tabu $i$-TSAB, elementos da técnica de *path relinking*, bem como novas propriedades teóricas de vizinhança para resolver o JSSP. Wong *et al.* (2008), e Wong *et al.* (2010) apresentam *Bee Colony Optimisation Algorithm* melhorado explorando o *big valley* no *fitness landscape* para resolver o JSSP, utilizando para testes as instâncias dos problemas proposto por Taillard (1993). Pardalos, Shylo e Vazacopoulos (2010) propuseram uma metaheurística para o *job shop scheduling* com abordagem que utiliza as propriedades do espaço de busca, tais como, *backbone* e *big valley* do JSSP para acelerar o processo de busca. Os resultados dos experimentos computacionais demonstraram a alta eficiência da abordagem proposta e novos limites superiores foram obtidos para muitos problemas.

Bożejko *et al.* (2017) apresentam um método para a construção de algoritmos para resolução de problemas de otimização discreto com base na análise do *landscape*. Os autores propõe uma metodologia para determinação da área formada por *big valley* para problemas de *scheduling* utilizando busca tabu. Mais recentemente, Bożejko *et al.* (2018) propuseram um método de controle da trajetória das metaheurísticas de busca local reduzindo a área de busca das metaheurísticas retirando da vizinhança as soluções não promissoras, o que é calculado com base em análise estatística da amostra de ótimos locais.

Portanto, em resumo, o principal objetivo da análise do *fitness landscape* é a extração de características do espaço de soluções, tanto de forma *online* quanto *offline*. As aplicações *offline* são destinadas a ampliar o entendimento do espaço de busca ou a prever o desempenho de algoritmos, para ajustar parâmetros ou escolher o melhor método de solução. Por outro lado, as abordagens *online* visam criar métodos auto-adaptativos ou controlar parâmetros em tempo de execução.

Nesse contexto, destacam-se abordagens baseadas em estatística básica, pois conforme Barnett (2003) quanto mais se conhece as propriedades estatísticas de uma classe de *fitness landscape* melhor equipado estará para criar algoritmos de busca efetivos para tais *landscapes*.

Por outro lado, a análise de componentes principais (PCA) tem sido usada com sucesso em diversas abordagens de seleção de atributos (BAIR *et al.*, 2006; JOLLIFFE, 1972; JOLLIFFE, 1973; SCHIMIT; PEREIRA, 2018). Nessas abordagens o objetivo é selecionar um subconjunto de variáveis a partir de um conjunto de observações, usualmente para propósitos de regressão ou classificação.

Portanto, dada a similaridade entre os problemas, justifica-se a investigação do uso da

PCA para extração de características do *fitness landscape* para o JSSP, principalmente quando se considera essas características para propor algoritmos mais eficientes. Vale frisar que, durante o período das pesquisas realizadas neste trabalho, as escassas aplicações da PCA no contexto do *fitness landscape* se concentram em aspectos teóricos e abordam o problema do caixeiro viajante (TAYARANI-NAJARAN; PRÜGEL-BENNETT, 2016).

Capítulo 4

# Materiais e Métodos

Para organização deste Capítulo, primeiramente apresenta-se o método de pesquisa realizado, em seguida os materiais, as configurações das instâncias do JSSP utilizadas nos experimentos, descrição da abordagem proposta, o planejamento das etapas experimentais e configurações dos parâmetros do BRKGA e do Algoritmo Genético Binário.

## 4.1 Métodos de Pesquisa

Este trabalho está contextualizado de acordo com Bertrand e Fransoo (2002) na Pesquisa Operacional (*Operations Research* – OR) no que diz respeito à aplicação de técnicas de otimização.

Segundo Chwif e Medina (2007), contextualiza-se também em Simulação e Otimização, visto que um dos objetivos é a busca por soluções ótimas ou próximas do ótimo para o JSSP fazendo uso de técnicas computacionais.

Este trabalho classifica-se, quanto ao tipo, como pesquisa descritiva explicativa com abordagem experimental, pois busca descrever e explicar resultados obtidos por meio de experimentos.

A pesquisa está delimitada ao problema de *Job Shop Scheduling* clássico com o objetivo de encontrar um sequenciamento das operações que minimize o *makespan* que é o tempo total de processamento de todas as operações nas máquinas no sistema de produção.

A Figura 4.1 apresenta as etapas desenvolvidas para esta pesquisa.

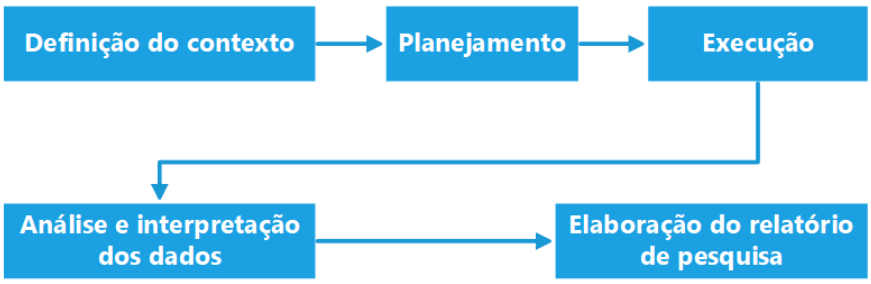



**Figura 4.1:** *Fluxo das etapas desenvolvidas para a pesquisa.*

Fonte : A autora.

Foram realizados estudos bibliográficos sobre o problema eas variáveis relacionadas. A pesquisa bibliográfica foi realizada, principalmente, por meio de sistemas de busca disponíveis no portal periódicos da CAPES: *Science Direct* On-Line e base de dados do IEEE foram as bases mais consultadas. Também foram obtidos artigos em anais de congressos, nacionais e internacionais. A estratégia de busca consistiu no uso dos seguintes termos: *job shop scheduling*, *fitness landscape*, *path-relinking*, análise de componentes principais,

técnicas metaheurísticas, algoritmo genético, espaço de busca, algoritmos de busca. Todos os termos foram utilizados em língua portuguesa e inglesa, de acordo com o sistema utilizado, sendo procurado no título, nas palavras-chave e no resumo. O levantamento bibliográfico foi iniciado no ano de 2015, sempre pesquisando os últimos 5 anos e, repetido até a conclusão do trabalho.

As etapas dos experimentos estão descritas na Seção 4.5 e os resultados obtidos foram comparados em relação à variável dependente *makespan* e com os resultados encontrados na literatura. Foram realizados estudos estatísticos dos resultados por meio das medidas de centralidade média e mediana.

## 4.2 Materiais

O Algoritmo Genético Híbrido (HGA) proposto tem por base o *Biased Random-Key Genetic Algorithm* (BRKGA), o qual é uma variação do Algoritmo Genético (Capítulo 2, Seção 2.5). Para a implementação HGA foi utilizada a API (*Application Programming Interface*, ou biblioteca em C++) do BRKGA proposta por Toso e Resende (2015) disponível em <http://mauricio.resende.info/src/brkgaAPI>.

Para a intensificação das buscas (Seção 4.4.2) foi implementado um Algoritmo Genético Binário Bidimensional (GAB) utilizando-se a biblioteca GAlib, escrita por Wall (2007), do *Massachusetts Institute of Technology*-MIT. A GAlib é uma biblioteca de componentes do algoritmo genético desenvolvida em C++, disponível em <http://lancet.mit.edu/ga/GAlib.html>.

A biblioteca de análise numérica e processamento de dados ALGLIB *Free Edition* foi utilizada para a Análise de Componentes Principais (PCA) e das medidas estatísticas de centralidade média e mediana. A ALGLIB *Free Edition* está disponível em <http://www.alglib.net>.

A linguagem de programação ANSI C/C++ foi utilizada para o desenvolvimento do Algoritmo Genético Híbrido. As bibliotecas são de código fonte aberto e estão disponíveis para download.

Foi implementado um algoritmo de Busca Local que utiliza o método de descida da encosta como estratégia de navegação e o operador de movimento N1 de Laarhoven, Aarts e Lenstra (1992) (Capítulo 2, Seção 2.3.3)

Os experimentos computacionais foram realizados em um computador com processador Intel© CORE(TM) i7 2,40GHz, sistema operacional Windows 10 de 64 bits e 8 GB de memória RAM.

## 4.3 Configurações das instâncias

Para a realização dos experimentos computacionais, foram utilizadas instâncias de duas classes dos problemas testes para o clássico *Job Shop Scheduling Problem*: as instâncias FT06 e FT10 propostas por Fisher e Thompson (1963), e as instâncias LA01 até LA33 propostas por Lawrence (1984), disponíveis na OR-Library (BEASLEY, 1990), em <https://www.eii.uva.es/elena/Elena%20Perez%20Vazquez_archivos/files_optimaJSSP/jobshop1.txt> mantida por Dirk C. Mattfeld e Rob J. M. Vaessens. Esses problemas são clássicos e o valor do *makespan* ótimo é conhecido conforme encontra-se na literatura (ASADZADEH, 2015; GAO *et al.*, 2015).

As configurações das instâncias são apresentadas na Tabela 4.1, na qual são listados os nomes de cada instância, o tamanho ($n \times m$), a quantidade de operações e, por fim, o valor do *makespan* da melhor solução conhecida (BKS, do inglês *Best Known Solution*).

**Tabela 4.1:** *Configurações das instâncias utilizadas nos experimentos computacionais.*

| **Instância** | **Tamanho** | **Oper.** | **BKS** | **Instância** | **Tamanho** | **Oper.** | **BKS** |
|---|---|---|---|---|---|---|---|
| FT06 | 6 x 6 | 36 | 55 | LA17 | 10 x 10 | 100 | 784 |
| FT10 | 10 x 10 | 100 | 930 | LA18 | 10 x 10 | 100 | 848 |
| LA01 | 10 x 5 | 50 | 666 | LA19 | 10 x 10 | 100 | 842 |
| LA02 | 10 x 5 | 50 | 655 | LA20 | 10 x 10 | 100 | 902 |
| LA03 | 10 x 5 | 50 | 597 | LA21 | 15 x 10 | 150 | 1046 |
| LA04 | 10 x 5 | 50 | 590 | LA22 | 15 x 10 | 150 | 927 |
| LA05 | 10 x 5 | 50 | 593 | LA23 | 15 x 10 | 150 | 1032 |
| LA06 | 15 x 5 | 75 | 926 | LA24 | 15 x 10 | 150 | 935 |
| LA07 | 15 x 5 | 75 | 890 | LA25 | 15 x 10 | 150 | 977 |
| LA08 | 15 x 5 | 75 | 863 | LA26 | 20 x 10 | 200 | 1218 |
| LA09 | 15 x 5 | 75 | 951 | LA27 | 20 x 10 | 200 | 1235 |
| LA10 | 15 x 5 | 75 | 958 | LA28 | 20 x 10 | 200 | 1216 |
| LA11 | 20 x 5 | 100 | 1222 | LA29 | 20 x 10 | 200 | 1157 |
| LA12 | 20 x 5 | 100 | 1039 | LA30 | 20 x 10 | 200 | 1355 |
| LA13 | 20 x 5 | 100 | 1150 | LA31 | 30 x 10 | 300 | 1784 |
| LA14 | 20 x 5 | 100 | 1292 | LA32 | 30 x 10 | 300 | 1850 |
| LA15 | 20 x 5 | 100 | 1207 | LA33 | 30 x 10 | 300 | 1719 |
| LA16 | 10 x 10 | 100 | 945 | | | | |

Fonte : Adaptado de Asadzadeh (2015) e Gao *et al.* (2015).

Dentre as instâncias listadas (Tabela 4.1) foram adotados critérios para selecionar àquelas que serão discutidas mais detalhadamente no Capítulo 5, sendo: duas instâncias

consideradas "fáceis" que são aquelas em que os métodos tradicionais encontram uma solução ótima facilmente, e duas instâncias consideradas "difíceis", para as quais os métodos não encontram uma ou mais soluções facilmente (LAMOS-DÍAZ *et al.*, 2017; PÉREZ; POSADA; HERRERA, 2012).

Ainda, as instâncias foram classificadas de acordo com o tamanho, ou seja, número de operações, sendo, instâncias quadradas no qual o número de *jobs* e máquinas são iguais, e instâncias retangulares. Após essa classificação então foram selecionadas uma instância de cada classe, listadas a seguir:

1. instâncias consideradas "fáceis": FT06 (quadrada) e LA07 (retangular);
2. instâncias consideradas "difíceis": FT10 (quadrada) e LA03 (retangular).

Cada instância do JSSP tem sua configuração caracterizada pela sequência tecnológica e pelos tempos de processamento das operações nas máquinas. As características para as instâncias FT06, FT10, LA03, LA07 são apresentadas a seguir, respectivamente nas Tabelas 4.2, 4.3, 4.4 e 4.5. Nessas tabelas, cada linha $i$ define a sequência tecnológica para o $i$-ésimo *job* e o tempo de processamento da operação, apresentado entre parênteses, na respectiva máquina.

A configuração da instância FT06 é apresentada na Tabela 4.2.

**Tabela 4.2:** *Sequência tecnológica e tempos de processamento da instância FT06.*

| ***Job*** | **Máquina (tempo de processamento)** | | | | | |
|---|---|---|---|---|---|---|
| J1 | 3 (1) | 1 (3) | 2 ( 6) | 4 (7) | 6 (3) | 5 (6) |
| J2 | 2 (8) | 3 (5) | 5 (10) | 6 (10) | 1 (10) | 4 (4) |
| J3 | 3 (5) | 4 (4) | 6 (8) | 1 (9) | 2 (1) | 5 (7) |
| J4 | 2 (5) | 1 (5) | 3 (5) | 4 (3) | 5 (8) | 6 (9) |
| J5 | 3 (9) | 2 (3) | 5 (5) | 6 (4) | 1 (3) | 4 (1) |
| J6 | 2 (3) | 4 (3) | 6 (9) | 1 (10) | 5 (4) | 3 (1) |

Fonte : Adaptado de OR-Library (BEASLEY, 1990).

A Tabela 4.3 apresenta a configuração da instância FT10.

**Tabela 4.3:** *Sequência tecnológica e tempos de processamento da instância FT10.*

| ***Job*** | **Máquina (tempo de processamento)** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| J1 | 1 (29) | 2 (78) | 3 (9) | 4 (36) | 5 (49) | 6 (11) | 7 (62) | 8 (56) | 9 (44) | 10 (21) |
| J2 | 1 (43) | 3 (90) | 5 (75) | 10 (11) | 4 (69) | 2 (28) | 7 (46) | 6 (46) | 8 (72) | 9 (30) |
| J3 | 2 (91) | 1 (85) | 4 (39) | 3 (74) | 9 (90) | 6 (10) | 8 (12) | 7 (89) | 10 (45) | 5 (33) |
| J4 | 2 (81) | 3 (95) | 1 (71) | 5 (99) | 7 (9) | 9 (52) | 8 (85) | 4 (98) | 10 (22) | 6 (43) |
| J5 | 3 (14) | 1 (6) | 2 (22) | 6 (61) | 4 (26) | 5 (69) | 9 (21) | 8 (49) | 10 (72) | 7 (53) |
| J6 | 3 (84) | 2 (2) | 6 (52) | 4 (95) | 9 (48) | 10 (72) | 1 (47) | 7 (65) | 5 (6) | 8 (25) |
| J7 | 2 (46) | 1 (37) | 4 (61) | 3 (13) | 7 (32) | 6 (21) | 10 (32) | 9 (89) | 8 (30) | 5 (55) |
| J8 | 3 (31) | 1 (86) | 2 (46) | 6 (74) | 5 (32) | 7 (88) | 9 (19) | 10 (48) | 8 (36) | 4 (79) |
| J9 | 1 (76) | 2 (69) | 4 (76) | 6 (51) | 3 (85) | 10 (11) | 7 (40) | 8 (89) | 5 (26) | 9 (74) |
| J10 | 2 (85) | 1 (13) | 3 (61) | 7 (7) | 9 (64) | 10 (76) | 6 (47) | 4 (52) | 5 (90) | 8 (45) |

Fonte : Adaptado de OR-Library (BEASLEY, 1990).

A configuração da instância LA03 é apresentada na Tabela 4.4.

**Tabela 4.4:** *Sequência tecnológica e tempos de processamento da instância LA03.*

| ***Job*** | **Máquina (tempo de processamento)** | | | | |
|---|---|---|---|---|---|
| J1 | 2 (23) | 3 (45) | 1 (82) | 5 (84) | 4 (38) |
| J2 | 3 (21) | 2 (29) | 1 (18) | 5 (41) | 4 (50) |
| J3 | 3 (38) | 4 (54) | 5 (16) | 1 (52) | 2 (52) |
| J4 | 5 (37) | 1 (54) | 3 (74) | 2 (62) | 4 (57) |
| J5 | 5 (57) | 1 (81) | 2 (61) | 4 (68) | 3 (30) |
| J6 | 5 (81) | 1 (79) | 2 (89) | 3 (89) | 4 (11) |
| J7 | 4 (33) | 3 (20) | 1 (91) | 5 (20) | 2 (66) |
| J8 | 5 (24) | 2 (84) | 1 (32) | 3 (55) | 4 (8) |
| J9 | 5 (56) | 1 (7) | 4 (54) | 3 (64) | 2 (39) |
| J10 | 5 (40) | 2 (83) | 1 (19) | 3 (8) | 4 (7) |

Fonte : Adaptado de OR-Library (BEASLEY, 1990).

Para a instância LA07, a sua configuração é apresentada na Tabela 4.5.

**Tabela 4.5:** *Sequência tecnológica e tempos de processamento da instância LA07.*

| ***Job*** | **Máquina (tempo de processamento)** | | | | |
|---|---|---|---|---|---|
| J1 | 1 (47) | 5 (57) | 2 (71) | 4 (96) | 3 (14) |
| J2 | 4 (75) | 2 (60) | 5 (22) | 4 (79) | 3 (65) |
| J3 | 2 (32) | 1 (33) | 3 (69) | 2 (31) | 5 (58) |
| J4 | 3 (44) | 2 (34) | 5 (51) | 4 (58) | 3 (47) |
| J5 | 3 (29) | 2 (44) | 1 (62) | 3 (17) | 5 (8) |
| J6 | 5 (15) | 3 (40) | 1 (97) | 5 (38) | 4 (66) |
| J7 | 5 (58) | 2 (39) | 1 (57) | 5 (20) | 4 (50) |
| J8 | 5 (57) | 4 (32) | 5 (87) | 1 (63) | 2 (21) |
| J9 | 4 (56) | 1 (84) | 3 (90) | 2 (85) | 4 (61) |
| J10 | 5 (15) | 1 (20) | 2 (67) | 4 (30) | 3 (70) |
| J11 | 5 (84) | 1 (82) | 2 (23) | 3 (45) | 4 (38) |
| J12 | 5 (50) | 3 (21) | 1 (18) | 5 (41) | 2 (29) |
| J13 | 1 (16) | 2 (52) | 1 (52) | 3 (38) | 4 (54) |
| J14 | 1 (37) | 1 (54) | 4 (57) | 3 (74) | 2 (62) |
| J15 | 1 (57) | 2 (61) | 1 (81) | 3 (30) | 4 (68) |

Fonte : Adaptado de OR-Library (BEASLEY, 1990).

As configurações das demais instâncias estão disponíveis em OR-Library (BEASLEY, 1990), conforme mencionado anteriormente (Seção 4.2).

## 4.4 Algoritmo Genético Híbrido Proposto

O BRKGA tem sido utilizado com sucesso em diversos problemas de otimização combinatória nos quais foram obtidos bons resultados, como por exemplo, no problema de minimização de troca de ferramentas (CHAVES *et al.*, 2016), em problemas de empacotamento 2D e 3D (GONÇALVES; RESENDE, 2012; GONÇALVES; RESENDE, 2013; ZUDIO *et al.*, 2018), problema de restrição de *clustering* (OLIVEIRA; CHAVES; LORENA, 2017), problemas de otimização na área da telecomunicação (RESENDE, 2012), bem como nos problemas de *scheduling* (CABO *et al.*, 2018; GONÇALVES; MENDES; RESENDE, 2005; GONÇALVES; RESENDE, 2014; LI; ZHANG, 2018).

Como apresentado na Seção 2.5 do Capítulo 2, o BRKGA possui duas partes distintas: uma parte independente do problema que consiste no Algoritmo Genético com seus métodos e gerações de cromossomos e, a parte dependente do problema que consiste no decodificador. Dessa forma, o BRKGA pode ser definido especificando-se como as soluções são codificadas e decodificadas, facilitando a sua adaptação para resolver problemas específicos de otimização combinatória como é o caso dos problemas de *scheduling* (GONÇALVES; RESENDE, 2011; TOSO; RESENDE, 2015).

Uma forma bem sucedida de obter metaheurísticas híbridas diz respeito à combinação de vários algoritmos de busca com forte especialização em intensificação e/ou diversificação. Diversificação e intensificação são duas questões principais para o desenvolvimento de um método de busca global (LOZANO; GARCÍA-MARTÍNEZ, 2010).

A diversificação geralmente refere-se à capacidade de visitar muitas e diferentes regiões do espaço de busca, enquanto a intensificação refere-se à capacidade de obter soluções de alta qualidade nessas regiões (Capítulo 2, Seção 2.5.2).

Metaheurísticas híbridas com algoritmos de busca que combinam as estratégias de diversificação e intensificação têm sido utilizadas para prover melhores resultados nos problemas de otimização, como pode ser visto em Glover e Laguna (1997), Jia e Hu (2014), Li e Gao (2016) e Yan, Sohn e Reyes (2017).

O Algoritmo Genético Híbrido (HGA) proposto é composto por uma busca local e mais dois procedimentos como segue:

a) BRKGA com uma Busca Local;

b) o procedimento para a diversificação populacional denominado $d$HGA que utiliza a Análise de Componentes Principais (PCA), conforme apresentado na Seção 4.4.1;

c) o procedimento para a intensificação das buscas denominado $i$HGA que utiliza um

Algoritmo Genético Binário Bidimensional (GAB) e o método com base no *Path Relinking*, conforme apresentado na Seção 4.4.2.

A Figura 4.2 ilustra o fluxograma principal do HGA.

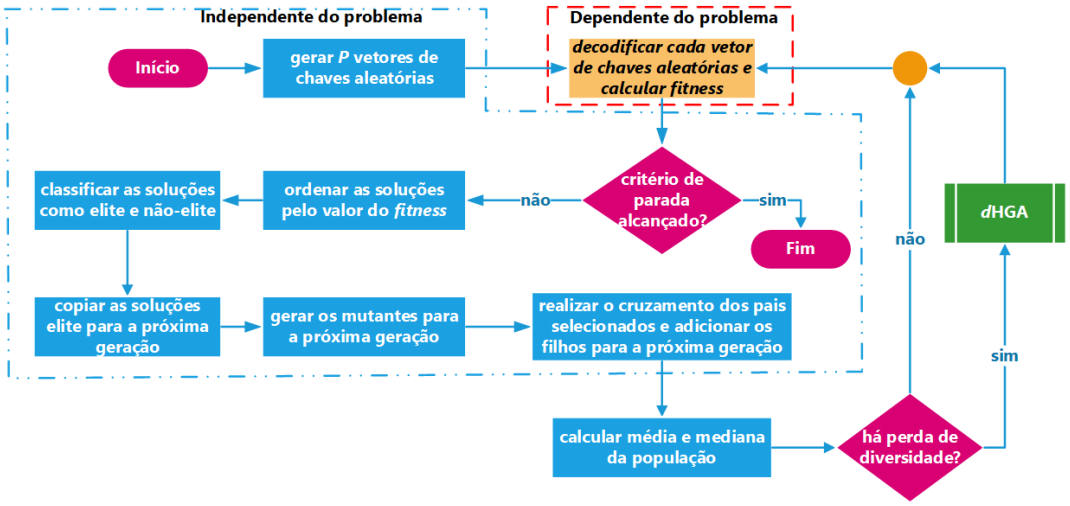



**Figura 4.2:** *Fluxograma do programa principal do Algoritmo Genético Híbrido - HGA.*
Fonte : A autora.

Vale ressaltar que durante a execução do HGA as medidas estatísticas média e mediana da população são calculadas a cada iteração para determinar se houve perda da diversidade populacional. Quando é identificada essa perda, então é realizada a chamada ao procedimento $d$HGA que por sua vez realiza a chamada ao procedimento $i$HGA.

A Figura 4.3 ilustra o fluxograma do $d$HGA e a chamada ao procedimento $i$HGA. A parte destacada com as linhas tracejadas na cor vermelha refere-se ao $d$HGA. Os procedimentos $d$HGA e $i$HGA, com o detalhamento de cada uma das instruções do fluxograma (Figura 4.3) são apresentados separadamente nas Seções 4.4.1 e 4.4.2, respectivamente.

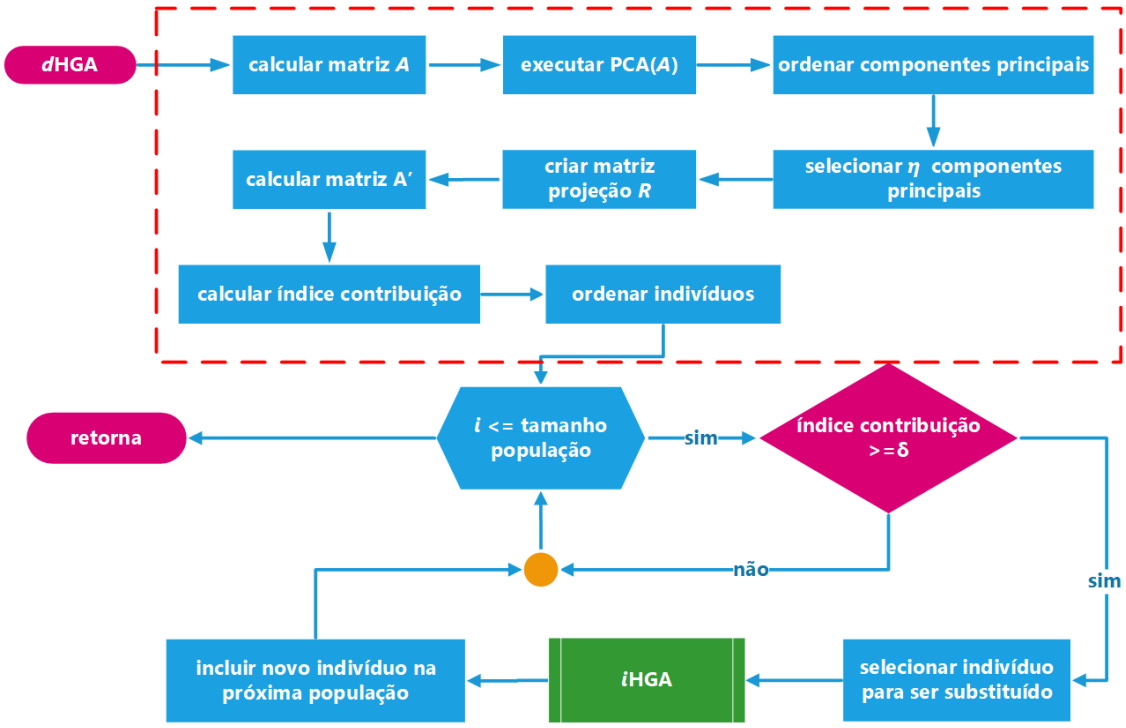



**Figura 4.3:** *Fluxograma do procedimento dHGA e a chamada ao procedimento iHGA*
Fonte : A autora.

### 4.4.1 Abordagem de Diversificação

Nesta abordagem utiliza-se a Análise de Componentes Principais (PCA) para avaliar qual ou quais indivíduos da população contribuem para a perda da diversidade populacional e são candidatos a serem substituídos na população.

A diversidade populacional é estimada com base nas medidas estatísticas de centralidade média e mediana da população, a cada geração, durante a evolução do HGA. Se a perda de diversidade populacional é identificada então o procedimento *d*HGA é executado, no qual, calcula-se o índice de contribuição ($CI$) de cada indivíduo à partir da sua relação entre todos os indivíduos no contexto da população. O procedimento *d*HGA é apresentado a seguir (Seção 4.4.1.1).

#### 4.4.1.1 Análise de componentes principais do *fitness landscape*

A PCA tem sido utilizada com sucesso na redução de dimensionalidade de um conjunto de dados de variáveis inter-relacionadas, bem como, em abordagens de seleção de atributos. Nessas abordagens o objetivo é selecionar um subconjunto de variáveis a partir de um conjunto de observações, geralmente para propósitos de regressão ou classificação. (BAIR *et al.*, 2006; JOLLIFFE, 2002; SADOWSKI; NIKOO; NIKOO, 2015; SCHIMIT; PEREIRA, 2018).

Assim sendo, a PCA foi utilizada neste trabalho devido a similaridade entre os problemas de redução de dimensionalidade e seleção de atributos com o problema de extração de características na análise do *fitness landscape* para o JSSP.

A aplicação da PCA em um conjunto de dados qualquer resulta em uma representação desses dados no subespaço dos componentes principais a qual é diferente dos dados originais (Capítulo 2 Seção 2.6). Essa abordagem usual não apenas modificaria os valores dos indivíduos da população mas também não levaria em conta o conceito de vizinhança entre esses indivíduos, que é um dos elementos fundamentais do *fitness landscape*.

Portanto, na abordagem de diversificação aplica-se a PCA não diretamente aos dados originais, mas em uma matriz $A = a_{i,j}$ definida a partir da relação entre os indivíduos da população. Essa matriz é calculada com base nos coeficiente de regressão padronizados entre pares de indivíduos da população. Denotando-se por $X_i$ e $X_j$ dois indivíduos da população, o coeficiente de regressão é calculado conforme a expressão 4.1:

$$a_{i,j} = \left(X_j^T X_i\right) / \sqrt{(X_j^T X_j)} \tag{4.1}$$

Assim, a PCA é executada sobre a matriz $A$ e ordena os componentes principais em ordem decrescente pelos autovalores. Os primeiros componentes principais são dispostos como colunas de uma matriz de projeção $R$, a qual é usada para calcular uma matriz $A'$ como a representação de $A$ no espaço dos componentes principais. Finalmente, é calculado um índice de contribuição $CI$ de cada indivíduo $X_j$ como a soma dos produtos escalares entre o $j$-ésima coluna de $A = A_j$, e cada coluna de $A'$, ponderada pelos autovalores correspondentes. O índice de contribuição considera apenas os primeiros componentes principais, suficientes para reter 98% da variação contida em $A$, como definido em Schimit e Pereira (2018). Vale ressaltar que cada coluna da matriz $A$ representa a relação de um indivíduo da população com todos os outros.

Considerando um JSSP com $n \times m$ operações, uma população de $\zeta$ indivíduos (soluções) denotadas por $X_1, X_2, ..., X_\zeta$, e um conjunto com os $\eta$ componentes principais $u_1, u_2, ..., u_\eta$ com os correspondentes autovalores $\lambda_1, \lambda_2, ..., \lambda_\eta$, $\eta < \zeta$, o procedimento proposto pode ser descrito como segue:

1. Calcular a matriz $A$ com os coeficientes de regressão padronizados $a_{i,j}$ para cada par de indivíduos da população $X_i$ e $X_j$;

2. Executar a PCA sobre a matriz $A$;

3. Ordenar os componentes principais em ordem decrescente em função dos autovalores;

4. Selecionar os $\eta$ primeiros componentes principais, $u_1, u_2, ..., u_\eta$ para criar uma matriz de projeção $R$;

5. Calcular a matriz $A'$ como uma representação de $A$ no espaço dos componentes principais: $A' = A_{\zeta \times \zeta} R_{\zeta \times \eta}$;
6. Calcular um índice de contribuição individual, da forma $CI(X_j) = \sum_{i=1}^{\eta} (A_j^T A'_j) * \lambda_i$;
7. Ordenar os indivíduos em ordem crescente com base no índice $CI$ calculado.

O fluxograma do procedimento $d$HGA descrito acima, para calcular o índice de contribuição dos indivíduos está em destaque com linhas tracejadas na cor vermelha da Figura 4.3 apresentada anteriormente na Seção 4.4.

É importante destacar que a matriz $A$ calculada conforme descrito permite que a PCA considere não apenas os indivíduos da população em seu cálculo, mas também uma noção de vizinhança entre os indivíduos da população. Ou seja, existe uma correlação entre o índice de contribuição $CI$ e a distância entre os indivíduos no espaço de busca, a qual é fundamental no conceito do *fitness landscape*.

Para exemplificar, considere a matriz $A$ dos coeficiente de regressão que foi calculada conforme equação 4.1 para uma população contendo 18 indivíduos. A Figura 4.4 apresenta os valores parciais dessa matriz.

$$A = \begin{bmatrix} 4,292 & 4,172 & 4,190 & 4,177 & 4,191 & 4,177 & 4,157 & 3,356 & \dots & 4,164 \\ 4,172 & 4,292 & 4,213 & 4,196 & 4,215 & 4,207 & 4,190 & 3,441 & \dots & 4,196 \\ 4,190 & 4,213 & 4,292 & 4,220 & 4,253 & 4,256 & 4,246 & 3,410 & \dots & 4,250 \\ 4,177 & 4,196 & 4,220 & 4,292 & 4,224 & 4,218 & 4,202 & 3,363 & \dots & 4,207 \\ 4,191 & 4,215 & 4,253 & 4,224 & 4,292 & 4,269 & 4,261 & 3,364 & \dots & 4,264 \\ 4,177 & 4,207 & 4,256 & 4,218 & 4,269 & 4,292 & 4,287 & 3,369 & \dots & 4,288 \\ 4,157 & 4,190 & 4,246 & 4,202 & 4,261 & 4,287 & 4,292 & 3,340 & \dots & 4,291 \\ 3,331 & 3,415 & 3,384 & 3,338 & 3,338 & 3,343 & 3,315 & 4,259 & \dots & 3,324 \\ 3,687 & 3,673 & 3,649 & 3,634 & 3,660 & 3,696 & 3,708 & 3,566 & \dots & 3,705 \\ 3,184 & 3,229 & 3,256 & 3,147 & 3,231 & 3,233 & 3,244 & 3,359 & \dots & 3,241 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 4,164 & 4,196 & 4,205 & 4,207 & 4,264 & 4,288 & 4,297 & 3,349 & \dots & 4,292 \end{bmatrix}$$

**Figura 4.4:** *Valores parciais da matriz A dos coeficientes de regressão de uma população com 18 indivíduos.*

Fonte : A autora.

Em seguida, aplica-se a PCA à matriz $A$ gerando-se a sua representação no espaço dos componentes principais, $A'$. Finalmente, calcula-se os índices de contribuição $CI$ de cada indivíduo.A Figura 4.5 é uma representação do índice $CI$ de cada um dos indivíduos.

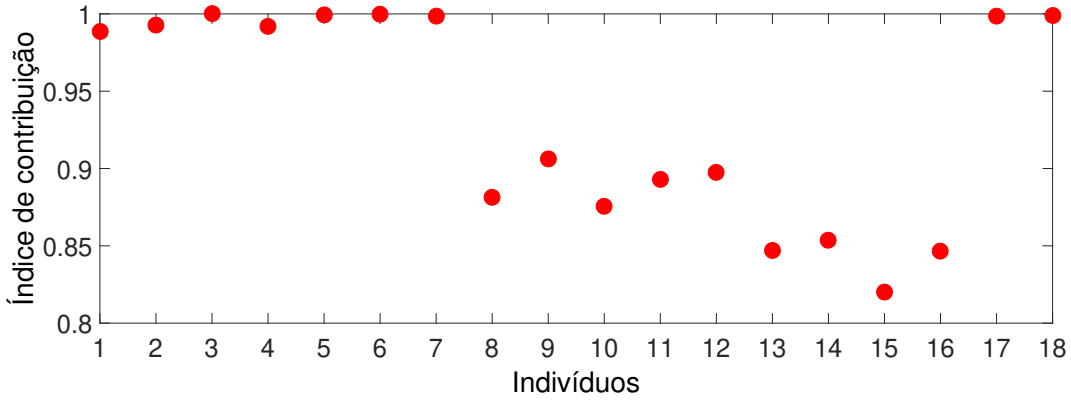



**Figura 4.5:** *Representação dos 18 indivíduos da população considerando-se o seu índice de contribuição*

Fonte : A autora.

Analisando a correlação entre o índice $CI$ e a distância média entre cada individuo da população e os demais, observa-se neste exemplo, um coeficiente de correlação de -0,9567, o que mostra que quanto maior o índice $CI$, menor é a distância média entre os indivíduos da população. Portanto, o índice $CI$ pode ser utilizado como uma indicação da perda da diversidade populacional, visto que, indivíduos próximos indicam baixa diversidade.

Ressalta-se que esse índice $CI$ representa uma relação de um-para-muitos e não de um-para-um como ocorre com a matriz de distância ou a própria matriz $A$. Ademais, essa mesma correlação não é observada entre os valores da matriz $A$ e as distâncias médias supracitadas o que demonstra a importância da aplicação da PCA.

Esse tipo de classificação poderia ser obtida com a utilização de outras técnicas como, por exemplo, k-*means*. A escolha do PCA se deu em função dos bons resultados obtidos em outros trabalhos (BAIR *et al.*, 2006), e a comparação de técnicas para este propósito está fora do escopo deste trabalho.

### 4.4.2 Abordagem de Intensificação

A abordagem de intensificação também é realizada com base nos resultados gerados pela PCA e de forma concomitante à diversificação. Para tal, considera-se os índices de contribuição $CI$ calculados para os indivíduos da população e os respectivos valores de aptidão (*makespan*), para definir quais soluções classificadas pela PCA ou serão substituídas por indivíduos aleatórios, ou passarão pelo processo de intensificação. O fluxograma do procedimento de intensificação $i$HGA é apresentado na Figura 4.6.

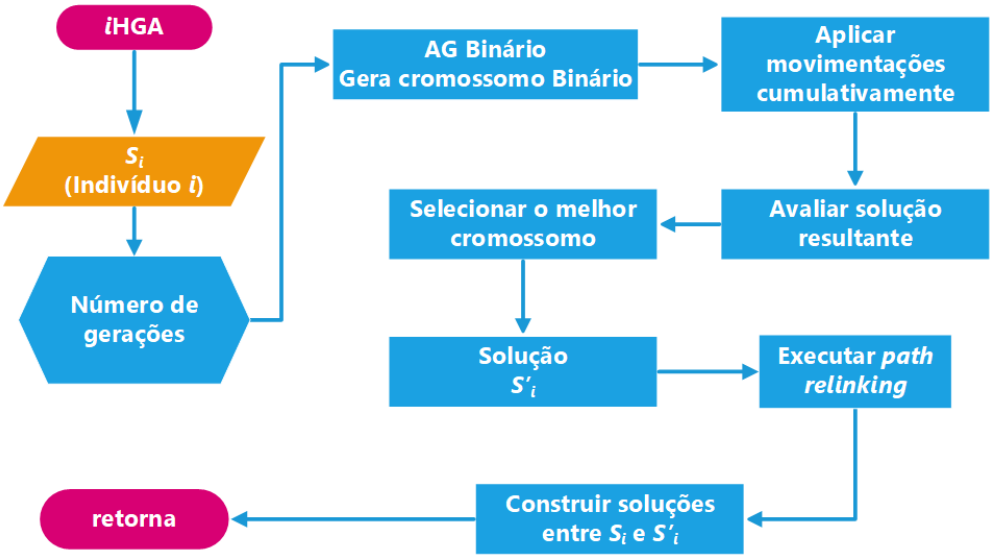



**Figura 4.6:** *Fluxograma do procedimento computacional iHGA para intensificação.*
Fonte : A autora.

Indivíduos que apresentam índices $CI$ próximos de 1,00 fazem parte de um grupo com baixa diversidade populacional e são, portanto, candidatos a serem substituídos na etapa de diversificação. Ocorre que a simples substituição desses indivíduos, sem considerar o valor do *makespan*, pode eliminar soluções potenciais. Nesse caso, antes da sua substituição, o indivíduo é submetido à etapa de intensificação.

Assim, apesar do HGA preservar ao menos 30% da população elite, a abordagem proposta pode remover indivíduos subótimos importantes na estrutura do *fitness landscape*, devido ao conceito do "*big valley*" (Capítulo 2, Seção 2.3) segundo o qual soluções ótimas e subótimas estão, eventualmente, conectadas no espaço de busca. Essa característica pode fazer com que a aplicação do *d*HGA produza uma piora na solução.

Visando atenuar esse efeito, a etapa de intensificação amplia a vizinhança do indivíduo selecionado na busca por uma solução de melhor qualidade. Para tanto, foi desenvolvido um Algoritmo Genético Binário (GAB) adaptado de Grassi, Schimit e Pereira (2016), conforme apresentado na Seção 4.4.2.1. O GAB permite que mais de uma movimentação seja feita na solução simultaneamente, diferente da forma tradicional na qual um vizinho de uma solução é gerado por meio de um único movimento ou permutação.

Posteriormente, o indivíduo selecionado e a melhor solução obtida pelo GAB (em caso de melhoria) são enviados para o método *path relinking* que reconecta o caminho entre essas duas soluções, explorando assim o conceito do "*big valley*", conforme apresentado na Seção 4.4.2.2.

#### 4.4.2.1 Algoritmo Genético Binário Bidimensional

O Algoritmo Genético Binário Bidimensional (GAB) desenvolvido para a etapa de intensificação tem por base o método de semente dinâmica proposto por Grassi, Schimit e Pereira (2016). A ideia do GAB é criar um cromossomo bidimensional binário que induz permutações em uma solução previamente definida.

Na abordagem proposta por Grassi, Schimit e Pereira (2016) as movimentações são realizadas da seguinte forma: sendo um gene igual a 1 na posição $(l, c)$ do cromossomo binário, indica uma permutação entre o *job* da posição $c$ com o *job* da posição $c + 1$ na máquina $l$.

A forma como essas movimentações são realizadas é o que difere, essencialmente, no GAB proposto. Neste trabalho, sendo o gene com valor igual a 1, significa a antecipação da operação $O_{jl}$ para a primeira posição na máquina $l$, sendo $j$ o *job* da posição $c$. A mudança na movimentação de troca para inserção é comprovadamente mais eficiente (GRABOWSKI; WODECKI, 2005).

Dada uma solução $s_i$, de tamanho $n \times m$, classificada pelo $d$HGA como candidata a ser eliminada/substituída, define-se um GAB com $m$ linhas e $n - 1$ colunas. Então, cada cromossomo binário gerado é usado para induzir movimentações na ordem das operações da solução $s_i$ original, sendo uma linha do cromossomo para cada máquina.

Durante o procedimento do GAB, as movimentações são aplicadas cumulativamente, para cada cromossomo, e a solução resultante é então avaliada e, ao final dessa etapa, o melhor cromossomo encontrado produz uma nova solução $s'_i$. Adicionalmente, o melhor cromossomo é utilizado em um procedimento similar ao método *path relinking* para construir todas as soluções entre $s_i$ e $s'_i$ na busca por um substituto melhor para $s_i$, conforme ilustrado na Figura 4.7. Nesse caso, as movimentações induzidas pelo cromossomo binário são realizadas uma-a-uma, seguida da avaliação de cada solução intermediária gerada.

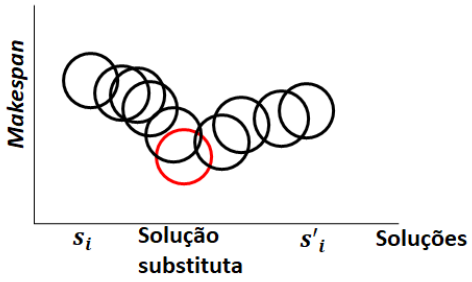



**Figura 4.7:** *Representação da intensificação da busca entre a solução $s_i$ e a solução $s'_i$. Na Figura cada círculo representa uma solução.*

Fonte : A autora.

Para ilustrar o procedimento, retoma-se o exemplo apresentado na Seção 2.2.2.1 do Capítulo 2, com 3 *jobs* e 3 máquinas. Considere a solução apresentada na Figura 4.8 já

decodificada para a representação do JSSP (conforme Seção 4.6.1), selecionada pelo seu índice $CI$ para ser substituída na população.

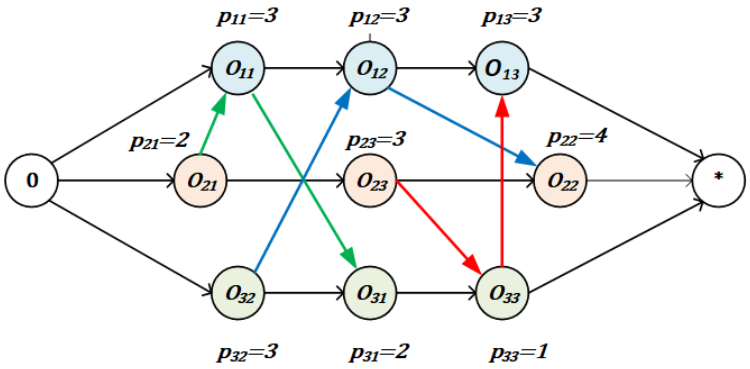



**Figura 4.8:** *Grafo conjuntivo da representação da solução s.*

Fonte : A autora.

Essa solução também pode ser representada por uma matriz $s$ na qual a primeira linha define a sequência dos *jobs* que serão processados na máquina 1, a segunda linha define a ordem dos *jobs* que serão processados na máquina 2 e, assim por diante. A Figura 4.9 representa a matriz $s$.

$$s = \begin{bmatrix} 2 & 1 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix}$$

**Figura 4.9:** *Matriz da representação da solução s*

Fonte : A autora.

Considerando o cromossomo binário bidimensional, conforme ilustrado na Figura 4.10, gerado pelo GAB, cada linha do cromossomo corresponde à uma máquina, então, a linha 1 do cromossomo corresponde às permutações que deverão ocorrer nas operações da máquina 1, e assim por diante. Quando um gene desse cromossomo é igual a 1, significa que deverá ocorrer a antecipação da operação daquele determinado *job* para a primeira posição na respectiva máquina.

$$\text{cromossomo} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$$

**Figura 4.10:** *Ilustração de um cromossomo binário bidimensional gerado pelo GAB*

Fonte : A autora.

No exemplo apresentado acima (Figura 4.10), o primeiro gene do cromossomo na linha 1 é igual a 1 e, indica que deverá ocorrer a antecipação da segunda operação da máquina

1 para a primeira operação na mesma máquina; o segundo gene do cromossomo, também na linha 1, indica que a terceira operação da máquina 1 deverá ser antecipada para a primeira da operação da máquina 1, e assim por diante; quando o gene é igual a zero, não ocorre permutação. O resultado das permutações cumulativas que ocorreram na solução $s$ do exemplo acima é apresentado na Figura 4.11.

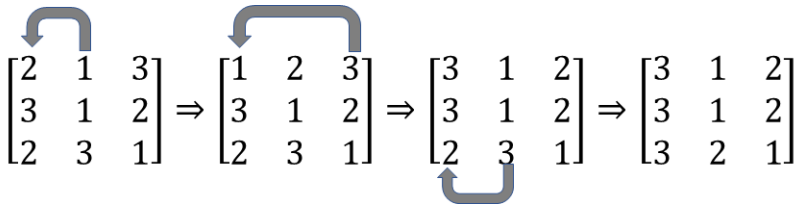


**Figura 4.11:** *Permutações cumulativas geradas na solução s à partir do cromossomo binário bidimensional gerado pelo GAB*

Fonte : A autora.

As soluções neste GAB são avaliadas da seguinte forma: todas as permutações induzidas pelo cromossomo binário, no qual o gene tem valor igual a 1, são aplicadas ao mesmo tempo, gerando uma nova solução $s'$ a qual é então avaliada calculando-se o *makespan*.

Dessa forma, à partir de uma solução $s$ e um cromossomo binário bidimensional gerado pelo GAB, obtém-se uma solução final $s'$. A Figura 4.12 apresenta a solução inicial $s$, um cromossomo binário e solução final $s'$, para o exemplo acima, após todas as movimentações que foram realizadas de acordo com o cromossomo binário.

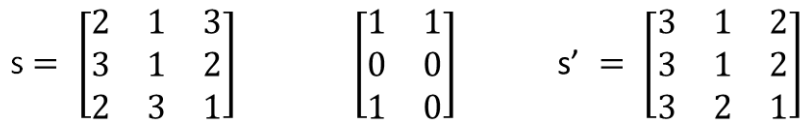


**Figura 4.12:** *Solução inicial s, cromossomo binário bidimensional gerado pelo GAB, e solução final $s'$*

Fonte : A autora.

Se ao final desse processo o GAB encontrar uma ou mais soluções melhores do que aquela solução $s$ preestabelecida inicialmente, então o cromossomo que gerou essa solução, juntamente com a solução inicial $s$ são enviados para o método *path relinking*, para assim então, recriar o caminho entre as duas soluções.

#### 4.4.2.2 Método Path Relinking

O *Path Relinking* é uma estratégia de intensificação de busca para explorar trajetórias que conectam soluções de elite, ou seja, soluções de alta qualidade em problemas de otimização combinatória. Como um aprimoramento dos métodos de busca heurística, sua

hibridização com outras metaheurísticas levou a melhorias significativas na qualidade da solução e nos tempos de execução das heurísticas híbridas (RESENDE; RIBEIRO, 2016c). O *Path Relinking* tem sido utilizado com sucesso em problemas de otimização como pode ser visto em Nowicki e Smutnicki (2005a) e Peng, Lü e Cheng (2015), por exemplo.

Neste trabalho, o próprio cromossomo binário gerado pelo GAB já define a trajetória que o *Path Relinking* deverá recriar o que difere das abordagens comumente utilizadas (Seção 2.7 do Capítulo 2). Por exemplo, Aiex, Binato e Resende (2003) considera a solução com o melhor *makespan* no caminho como a solução de referência, enquanto que Nasiri e Kianfar (2012) parte de uma solução inicial, para em uma iteração específica e retorna a solução corrente como sendo a solução referência. Peng, Lü e Cheng (2015) utilizaram uma estratégia dedicada com base no mecanismo de controle de distância adaptativo para obter a solução mais promissora.

No procedimento *i*HGA, após a etapa realizada pelo GAB, o *Path Relinking* recebe a solução inicial $s$ e o melhor cromossomo binário do GAB, ou os melhores cromossomos caso exista mais de um, para criar o caminho entre a solução inicial $s$ e a solução $s'$ com base no cromossomo binário do GAB, sendo que, nesta etapa, todas as solução são geradas e avaliadas. Dado o exemplo apresentado anteriormente, a Figura 4.13 ilustra a construção das soluções na trajetória do *Path Relinking*, isto é, o *Path Relinking* realiza todas as permutações indicadas pelo cromossomo binário do GAB, uma-a-uma e calcula-se o *makespan* de cada uma delas.

$$s = \begin{bmatrix} 2 & 1 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix} s_{1=} \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix} s_2 = \begin{bmatrix} 3 & 1 & 2 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix} s' = \begin{bmatrix} 3 & 1 & 2 \\ 3 & 1 & 2 \\ 3 & 2 & 1 \end{bmatrix}$$

**Figura 4.13:** *Soluções geradas e avaliadas pelo Path Relinking*

Fonte : A autora.

Ao terminar a construção do caminho entre a solução inicial $s$ e a solução final $s'$, o *Path Relinking* retornará a melhor solução encontrada no caminho, conforme ilustrado na Figura 4.7, a qual substituirá a solução $s$.

É importante ressaltar que, os exemplos apresentados anteriormente, são com base em uma solução já decodificada para a representação do JSSP para facilitar o entendimento, porém, no algoritmo são utilizados os vetores de chaves aleatórias como é a representação do BRKGA e são realizadas todas as operações necessárias para chegar-se à solução final.

## 4.5 Planejamento das Etapas Experimentais

Para a realização dos experimentos computacionais, a fim de atingir os objetivos propostos neste trabalho, o planejamento dos experimentos computacionais foi dividido em

três etapas, conforme apresentado a seguir.

- **Etapa 1: BRKGA com Busca Local -** A primeira etapa dos experimentos foi realizada utilizando o BRKGA com um método de Busca Local, doravante referenciado como GABL. O método de busca local utilizado em todas as etapas é baseado na estrutura de vizinhança conhecida N1, na qual todas as operações críticas em uma mesma máquina são permutadas, duas a duas (AMIRGHASEMI; ZAMANI, 2015; WANG *et al.*, 2018). Ressalta-se ainda, que a Busca Local é realizada em todas as etapas dos experimentos, ou seja, nas três abordagens.

  Apesar da busca local com vizinhança N1 não ser a mais eficiente do ponto de vista computacional, ela foi adotada por garantir que todos os vizinhos são avaliados (KUHPFAHL; BIERWIRTH, 2016). Consequentemente, além do volume de dados, que são gravados em arquivo para análise, a questão sobre o tempo computacional não foi considerada.

  Os resultados obtidos para o *makespan* são comparados com a literatura e servirão de referência para comparação com a abordagem proposta. O objetivo nesta etapa é investigar as características do *fitness landscape* por meio dos resultados obtidos com o GABL para definir, por exemplo, medidas para a perda de diversidade populacional com base nessas características.

- **Etapa 2: HGA com diversificação -** Esta etapa, refere-se à abordagem de diversificação proposta e utiliza o procedimento computacional $d$HGA conforme apresentado na Seção 4.4.1, explorando-se as características do *fitness landscape* com base na análise dos componentes principais.

  Nesta etapa de diversificação da população são utilizadas as medidas estatísticas de centralidade média e mediana dos valores do *makespan* de cada indivíduo da população, a cada geração/iteração do HGA. Quando o valor da razão entre a média e a mediana for maior ou igual a noventa e nove ($media/mediana \geq 99$), então considera-se como perda de diversidade e o procedimento $d$HGA é executado. O valor da razão entre a média e a mediana foi definido empiricamente. O objetivo nesta etapa é verificar/validar a abordagem de diversificação comparando-se os resultados com aqueles da etapa GABL.

- **Etapa 3: HGA com diversificação e intensificação -** Esta etapa, refere-se a abordagem de diversificação (Seção 4.4.1) com a abordagem de intensificação das buscas conforme apresentado na Seção 4.4.2. Aqui, aplica-se então tanto o procedimento $d$HGA quanto o $i$HGA. Nesta etapa, se o índice de contribuição do indivíduo $CI$ é maior ou igual à noventa e oito ($CI \geq 98$), então o procedimento $i$HGA é executado. Aqui também, o valor de ($CI \geq 98$) foi definido empiricamente. Ob-

jetivo nesta etapa é verificar/validar o HGA proposto. Os resultados do HGA são comparados com aqueles obtidos nas duas etapas anteriores e com a literatura.

Ressalta-se ainda, que o custo computacional (tempo de processamento do algoritmo) não está sendo considerado devido à geração e armazenamento de um grande volume de dados para análise. Ademais, foi realizada somente uma execução de cada instância nos experimentos computacionais. As etapas elencadas acima estão representadas no diagrama da Figura 4.14.

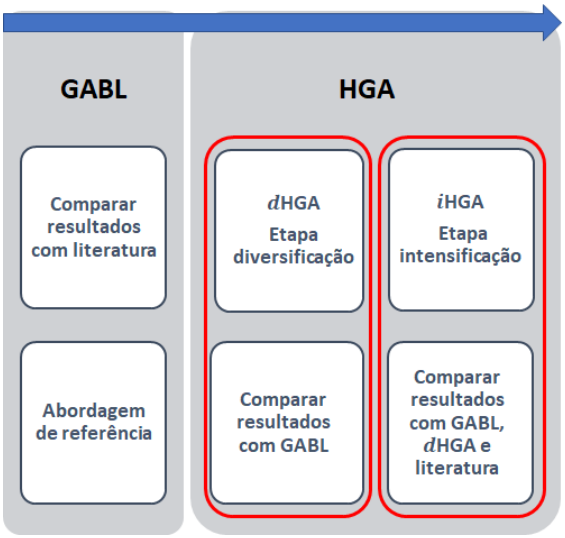



**Figura 4.14:** *Diagrama da representação das etapas para realização dos experimentos computacionais.*

Fonte : A autora.

As instâncias dos problemas que foram utilizadas nos experimentos, bem como suas configurações foram apresentadas na Seção 4.3

## 4.6 Definição dos Parâmetros do BRKGA e do Algoritmo Genético Binário

Os parâmetros do BRKGA utilizados nos experimentos foram configurados com base na literatura (GONÇALVES; MENDES; RESENDE, 2005; GONÇALVES; RESENDE, 2011) e são apresentados na Tabela 4.6. Esses parâmetros são aplicados para todas as instâncias dos problemas testadas e em todas as etapas definidas anteriormente (Seção 4.5).

Ademais, vale ressaltar que o critério de parada do algoritmo HGA e/ou do BRKGA com busca local (GABL) é de no máximo 200 gerações ou quando a solução ótima conhecida (BKS) é alcançada (Tabela 4.1)

**Tabela 4.6:** *Configurações dos parâmetros do HGA e/ou BRKGA com busca local.*

| **Parâmetro** | **Definição** |
|---|---|
| Tamanho do cromossomo | $n \times m$ |
| Tamanho da população $p$ | duas vezes o tamanho do cromossomo |
| População elite $pe$ | 10% de $pop$ |
| População mutante $pm$ | 20% de $pop$ |
| Probabilidade de herança | 70% de $pe$ |
| Número máximo de gerações | 200 |

Fonte : Gonçalves, Mendes e Resende (2005) e Gonçalves e Resende (2011).

O trabalho de Grassi, Schimit e Pereira (2016) mostra que o AG binário apresenta uma convergência alta nas primeiras gerações e depois fica estagnado, o que justifica a utilização de um número pequeno de gerações como critério de parada. Ademais, essa decisão também tem um impacto significativo no custo computacional, visto que esse procedimento de intensificação $i$HGA é aplicado em várias soluções.

Os parâmetros do GAB utilizados nos experimentos foram configurados com base no trabalho de Grassi, Schimit e Pereira (2016) e são apresentados na Tabela 4.7.

**Tabela 4.7:** *Configurações dos parâmetros do Algoritmo Genético Binário - GAB.*

| **Parâmetro** | **Definição** |
|---|---|
| Tamanho do cromossomo | $m \times (n-1)$. |
| Tamanho da população | 20 |
| Taxa de mutação | 5% |
| Taxa de cruzamento (*crossover*) | 90% |
| Taxa de substituição dos indivíduos | 50% |
| Critério de parada | 2 gerações sem melhora |

Fonte : A autora.

### 4.6.1 Codificação e decodificação de uma solução do JSSP

O BRKGA utiliza vetores de chaves aleatórias, conforme apresentado na Seção 2.5 do Capítulo 2, para codificação dos indivíduos da população. Esses cromossomos precisam ser decodificados para uma solução do problema.

Para exemplificar o processo de decodificação, retoma-se o exemplo apresentado na Seção 2.2.2.1, de 3 *jobs* e 3 máquinas, com o vetor $X$ de 9 chaves aleatórias apresentado na Seção 2.5.1 do Capítulo 2, o qual é replicado aqui em sua versão já ordenada.

Os valores do vetor $X$ e seus índices são apresentados a seguir.

| *índice* | 2 | 7 | 0 | 8 | 3 | 1 | 5 | 4 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| $X =$ | [0,145577 | 0,192037 | 0,50934 | 0,648522 | 0,713488 | 0,756746 | 0,797062 | 0,817017 | 0,90824] |

À partir dos índices do vetor $X$ é gerado um vetor $Y$ que resultará em uma solução para o JSSP. O vetor $Y$ é apresentado a seguir.

$$Y = [2 \quad 7 \quad 0 \quad 8 \quad 3 \quad 1 \quad 5 \quad 4 \quad 6]$$

O vetor $Y$ é então decodificado utilizando-se a representação baseada em operações (OB, do inglês *operation-based representation*) para gerar uma solução do JSSP. Para cada elemento de $Y$ calcula-se o módulo (resto da divisão) entre o valor do elemento e o número de *jobs* da instância do problema acrescido de uma unidade. Lembrando que no exemplo tem-se 3 *jobs*, o resultado da decodificação, gera uma solução $s$ para ser interpretada/avaliada, conforme apresentado a seguir.

$$s = \{3, \quad 2, \quad 1, \quad 3, \quad 1, \quad 2, \quad 3, \quad 2, \quad 1\}.$$

A vantagem da utilização da representação baseada em operações é que a solução resultante é sempre factível(WANG; LI, 2011).

Para cada ocorrência dos valores na solução $s$, é verificado na matriz de operações $T_{jk}$ (conforme apresentado no Capítulo 2, Seção 2.1) a sequência tecnológica dos *jobs*. Neste exemplo, a primeira ocorrência do valor 3 em $s$ representa a primeira operação do *job*3, que deverá ser processada na máquina 2, $O_{32}$; a segunda ocorrência do valor 3 em $s$, ou seja, o quarto valor, representa a segunda operação do *job*3 que deverá ser processada na máquina 1, $O_{31}$, e assim por diante. Dessa forma, conforme discutido na Seção 2.2.2.3 do Capítulo 2, a solução do JSSP é encontrada no grafo conjuntivo acíclico, aplicando-se uma direção aos arcos disjuntivos do grafo. A Figura 4.15 apresenta a interpretação da solução $s$ para o grafo conjuntivo do exemplo abordado.

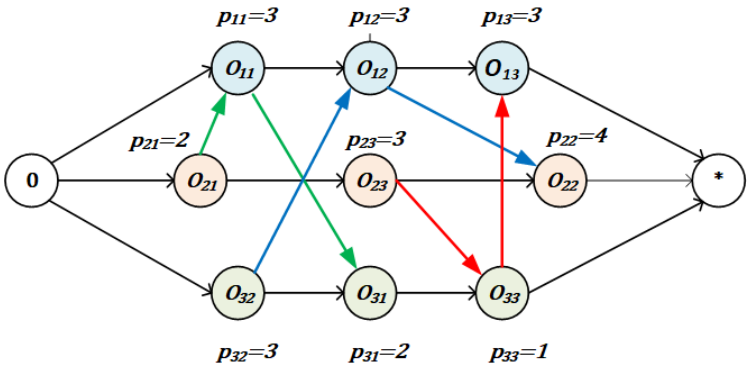



**Figura 4.15:** *Grafo conjuntivo da representação da solução s.*

Fonte : A autora.

À partir da solução $s$, o gráfico de Gantt pode ser elaborado, conforme apresentado na Figura 4.16.

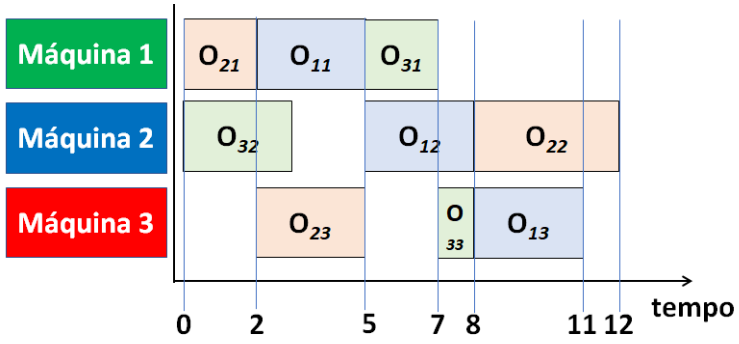



**Figura 4.16:** *Gráfico de Gantt para a representação da solução s.*

Fonte : A autora.

Ainda, vale ressaltar que à partir de uma solução, um vizinho pode ser alcançado quando aplica-se uma mudança de direção no arco conjuntivo de um par de operações realizadas na mesma máquina, respeitando as relações de precedência conforme determinado na sequência tecnológica.

Capítulo 5

# Resultados e Discussão

Neste Capítulo são apresentados os resultados obtidos para as instâncias FT06, FT10 e LA01 até LA33 perfazendo um total de 35 instâncias. São discutidos mais detalhadamente os resultados para as instâncias FT06, FT10, LA03 e LA07. Os resultados obtidos em cada etapa dos experimentos, conforme mencionado na Seção 4.5 no Capítulo 4, são discutidos como segue.

a) Primeiramente, na Seção 5.1 deste Capítulo são apresentados os resultados referente à etapa 1 do planejamento, os quais referem-se ao BRKGA com Busca Local, GABL e que serviram de referência para comparação com as abordagens propostas de diversificação e, diversificação com intensificação.

b) Os resultados obtidos pelo HGA com diversificação referente à etapa 2 do planejamento são apresentados na Seção 5.2. Nesta etapa utiliza-se a estratégia de diversificação da população implementada no procedimento *d*HGA. Os resultados obtidos foram comparados com os resultados do GABL (etapa 1).

c) Na Seção 5.3 são apresentados os resultados do HGA que contempla a abordagem de diversificação com intensificação (*d*HGA com *i*HGA) e comparados àqueles obtidos nas duas etapas anteriores. Ainda, compara-se os resultados do HGA com outras abordagens disponíveis na literatura (Seção 5.3.3).

## 5.1 Análise dos resultados experimentais do BRKGA com Busca Local

Os resultados obtidos nos experimentos utilizando o GABL são apresentados na Tabela 5.1. São listados os dados das instâncias, a solução ótima conhecida (BKS), o valor do *makespan* encontrado pelo GABL e o número de gerações necessárias. Vale ressaltar que o critério de parada do algoritmo é o número máximo de 200 gerações ou quando o valor do *makespan* ótimo é alcançado (Tabela 4.6 da Seção 4.6 do Capítulo 4).

As linhas da Tabela 5.1 em negrito destacam quais as instâncias que o algoritmo convergiu para uma solução ótima e o número de gerações necessárias. Como pode ser observado, das 35 instâncias testadas o GABL encontrou uma solução ótima para 18 instâncias, ou seja, aproximadamente 52% das instâncias listadas.

**Tabela 5.1:** *Resultados dos experimentos do GABL*

| Instância | Tamanho | Operações | BKS | GABL | |
|---|---|---|---|---|---|
| | | | | MKP | Gerações |
| **FT06** | **6 × 6** | **36** | **55** | **55** | **8** |
| FT10 | 10 × 10 | 100 | 930 | 1019 | 200 |
| **LA01** | **10 × 5** | **50** | **666** | **666** | **8** |
| **LA02** | **10 × 5** | **50** | **655** | **655** | **20** |
| LA03 | 10 × 5 | 50 | 597 | 604 | 200 |
| LA04 | 10 × 5 | 50 | 590 | 598 | 200 |
| **LA05** | **10 × 5** | **50** | **593** | **593** | **1** |
| **LA06** | **15 × 5** | **75** | **926** | **926** | **7** |
| **LA07** | **15 × 5** | **75** | **890** | **890** | **76** |
| **LA08** | **15 × 5** | **75** | **863** | **863** | **16** |
| **LA09** | **15 × 5** | **75** | **951** | **951** | **8** |
| **LA10** | **15 × 5** | **75** | **958** | **958** | **3** |
| **LA11** | **20 × 5** | **100** | **1222** | **1222** | **16** |
| **LA12** | **20 × 5** | **100** | **1039** | **1039** | **12** |
| **LA13** | **20 × 5** | **100** | **1150** | **1150** | **10** |
| **LA14** | **20 × 5** | **100** | **1292** | **1292** | **1** |
| **LA15** | **20 × 5** | **100** | **1207** | **1207** | **34** |
| LA16 | 10 × 10 | 100 | 945 | 982 | 200 |
| LA17 | 10 × 10 | 100 | 784 | 793 | 200 |
| LA18 | 10 × 10 | 100 | 848 | 861 | 200 |
| LA19 | 10 × 10 | 100 | 842 | 874 | 200 |
| LA20 | 10 × 10 | 100 | 902 | 907 | 200 |
| LA21 | 15 × 10 | 150 | 1046 | 1086 | 200 |
| LA22 | 15 × 10 | 150 | 927 | 982 | 200 |
| **LA23** | **15 × 10** | **150** | **1032** | **1032** | **70** |
| LA24 | 15 × 10 | 150 | 935 | 1002 | 200 |
| LA25 | 15 × 10 | 150 | 977 | 1029 | 200 |
| LA26 | 20 × 10 | 200 | 1218 | 1240 | 200 |
| LA27 | 20 × 10 | 200 | 1235 | 1311 | 200 |
| LA28 | 20 × 10 | 200 | 1216 | 1251 | 200 |
| LA29 | 20 × 10 | 200 | 1157 | 1250 | 200 |
| LA30 | 30 × 10 | 300 | 1355 | 1369 | 200 |
| **LA31** | **30 × 10** | **300** | **1784** | **1784** | **118** |
| **LA32** | **30 × 10** | **300** | **1850** | **1850** | **138** |
| **LA33** | **30 × 10** | **300** | **1719** | **1719** | **195** |

Considerando-se as instâncias que o algoritmo convergiu para o valor ótimo do *makespan* (Tabela 5.1), pode-se destacar que o tamanho da instância (número de operações, $n \times m$ ) não é uma medida suficiente para determinar a dificuldade do problema. Por exemplo, para as instâncias do LA11 até o LA15 (100 operações) e do LA31 até o LA33 (300 operações) o algoritmo convergiu para o valor ótimo do *makespan*, enquanto que para as instâncias LA03 e LA04, com 50 operações apenas, uma solução ótima não foi encontrada. Portanto, o tamanho não é a única característica do problema e do *fitness landscape* a afetar o desempenho do algoritmo.

Importante mencionar que, de acordo com Streeter e Smith (2006), a razão entre o número de *jobs* e o de máquinas $\frac{n}{m}$ também influencia no *fitness landscape* e consequentemente no desempenho do algoritmo. Os JSSP quadrados, aqueles no qual a razão entre *job* e máquina é igual a um, ($\frac{n}{m} = 1$) são mais difíceis de serem resolvidos do que os retangulares (GRABOWSKI; WODECKI, 2005; STREETER; SMITH, 2006).

Ainda, conforme estudos realizados por Pérez, Herrera e Hernández (2003) e Pérez, Posada e Herrera (2012), o número de diferentes soluções ótimas globais distribuídas no espaço de soluções pode indicar a dificuldade do problema, conforme apresentado na Tabela 5.2.

**Tabela 5.2:** *Número de diferentes soluções ótimas globais conhecidos.*

| FT06 | FT10 | LA01 | LA02 | LA03 | LA04 | LA05 |
|---|---|---|---|---|---|---|
| 42 | 4 | 26813 | 5321 | 706 | 143 | 484714 |

Fonte : Adaptado de Pérez, Herrera e Hernández (2003), Pérez, Posada e Herrera (2012).

Após os experimentos e observação dos dados gerados pelo GABL, foram realizados testes estatísticos de centralidade, utilizando-se a média e a mediana dos valores do *makespan* de todos os indivíduos da população, a cada geração do GABL, para as instâncias que são objetos de estudo mais detalhado (Capítulo 4, Seção 4.3). Foram observadas algumas similaridades no comportamento dos dados estatísticos dessas instâncias, principalmente nos casos em que a solução ficou presa em um ótimo local ou o algoritmo demandou um número de gerações relativamente alto para alcançar o ótimo global.

As Seções a seguir discutem as análises dos resultados do GABL para as instâncias FT06, FT10, LA03 e LA07, conforme mencionado anteriormente.

### 5.1.1 Resultados dos experimentos do GABL para as instâncias FT06 e LA07

As instâncias FT06 e LA07 foram classificadas anteriormente neste trabalho como instâncias fáceis de serem resolvidas (Capítulo 4, Seção 4.5). Os resultados dos experimentos

computacionais são discutidos, primeiramente para o FT06 e a seguir, para a instância LA07.

No caso da instância FT06, foram necessárias apenas oito gerações para o GABL encontrar uma solução ótima, sendo o valor do *makespan* ótimo igual a 55. O gráfico apresentado na Figura 5.1 traz informações referente às medidas estatísticas média e mediana, e também os valores referente à evolução da população durante as gerações do GABL.

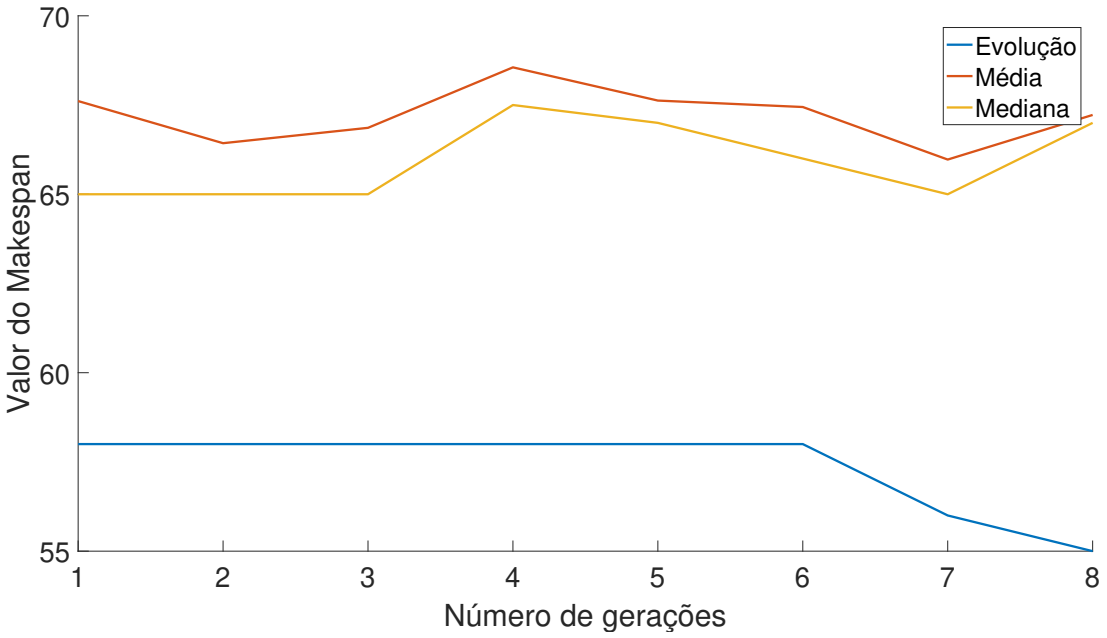



**Figura 5.1:** *Média, mediana e evolução da população da instância FT06 a cada geração do GABL.*

Fonte : A autora.

Apesar dos poucos dados para essa instância, visto que o GABL encontrou uma solução ótima do problema em oito gerações, nota-se no gráfico (Figura 5.1) que a média e mediana têm comportamento parecido. Por sua vez, olhando para a linha da evolução da população no gráfico, nota-se algoritmo ficou estagnado em um ótimo local até a sexta geração e depois evoluiu para o ótimo global.

Para o FT06, mesmo sendo um problema pequeno (6 *jobs* e 6 máquinas, 36 operações), observou-se que existe um número considerável de indivíduos com valores de *makespan* 64 e 66 (aproximadamente 14% e 12% dos indivíduos da população, respectivamente) e, aproximadamente 10% dos indivíduos da população com valores de *makespan* 57 e 59. Esse fato pode ser verificado no histograma apresentado na Figura 5.2. Mesmo o FT06 sendo resolvido em poucas gerações e de fácil solução, percebe-se a tendência do GABL gerar indivíduos com valores de *makespan* cada vez mais parecidos. Vale frisar que indivíduos com o mesmo valor de *makespan* não são necessariamente iguais, ou seja, pode haver mais de um sequenciamento diferente mas com o mesmo do valor *makespan* ótimo.

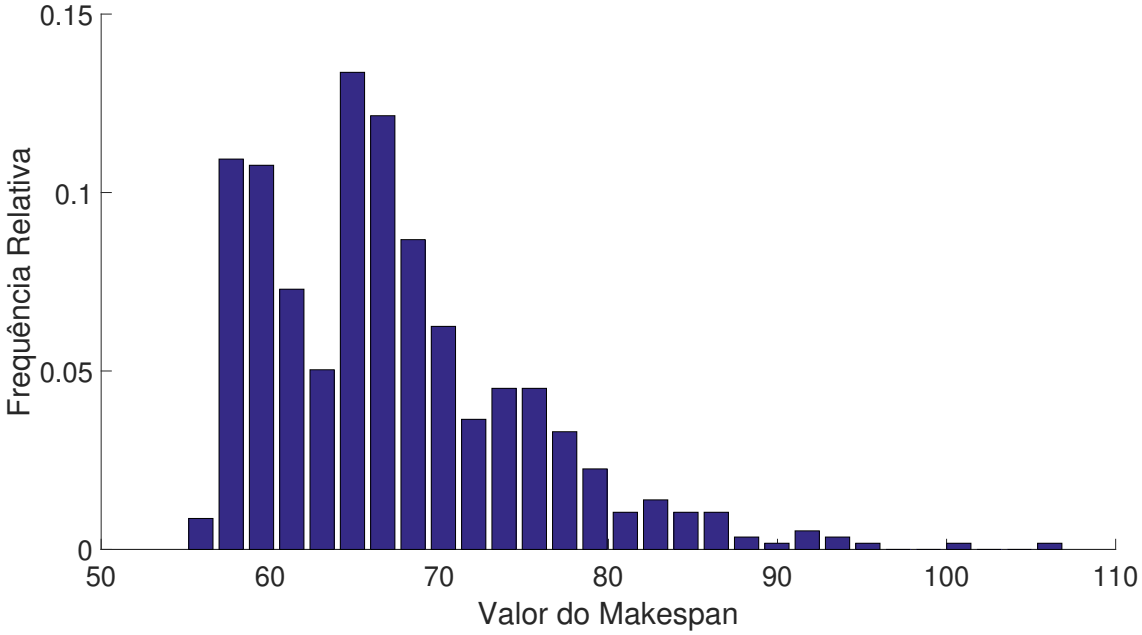



**Figura 5.2:** *Histograma dos indivíduos da população durante evolução do GABL para a instância FT06.*

Fonte : A autora.

O LA07 é um problema de maior dimensão quando comparado ao FT06, com 15 *jobs*, 5 máquinas, 75 operações e com valor do *makespan* ótimo conhecido igual a 890 (Tabela 4.1). A solução ótima para o LA07 foi encontrada pelo GABL em 76 gerações (Tabela 5.1). Os valores do *makespan* por geração durante a evolução da população pelo GABL, bem como os valores das medidas estatísticas média e mediana são apresentados no gráfico da Figura 5.3.

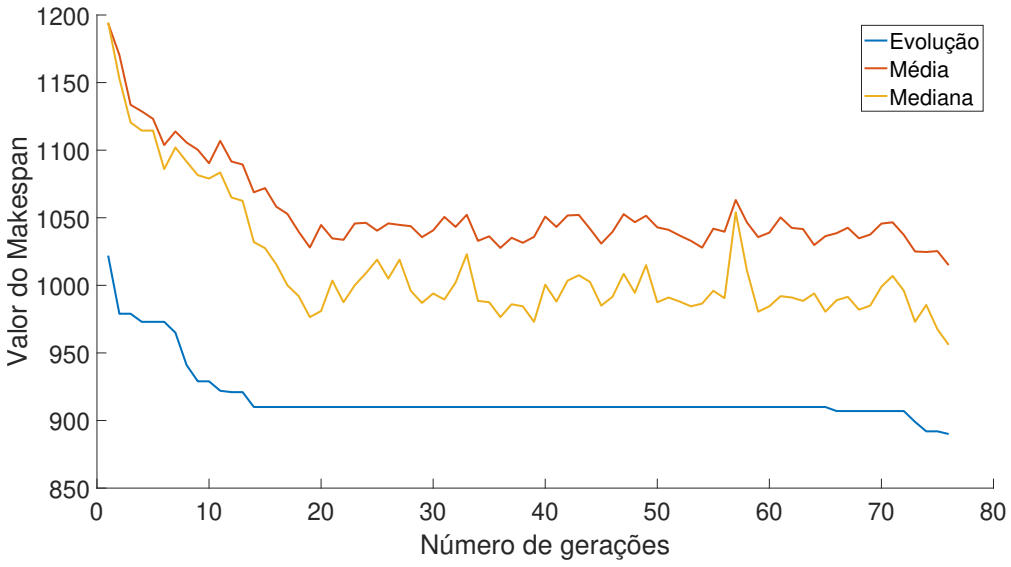



**Figura 5.3:** *Média, mediana e evolução da população da instância LA07 a cada geração do GABL.*

Fonte : A autora.

Nos resultados do LA07 apresentados no gráfico da Figura 5.3, observa-se que durante as primeiras dez gerações, a média e a mediana estão próximas e depois há um afastamento da mediana tornando-se mais pronunciado ao longo das gerações. Nota-se que próximo a décima terceira geração, os valores do *makespan* permanecem iguais por várias gerações indicando que houve estagnação na evolução do GABL, o que provavelmente indica que o algoritmo não conseguiu sair de uma região do espaço de busca que contém bacias atratoras. À partir da 66ª geração, observa-se que o algoritmo saiu do ponto de estagnação, explorando regiões mais promissoras e até alcançar o ótimo global. A Tabela 5.3 apresenta a geração e o valor do *makespan* que foi melhorado durante a evolução do algoritmo.

**Tabela 5.3:** *Evolução da população para a instância LA07: número da geração e valor do makespan no qual o algoritmo melhorou a solução.*

| **Geração** | 1 | 2 | 4 | 7 | 8 | 9 | 11 | 12 | 14 | 66 | 73 | 74 | 75 | 76 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **MKP** | 1022 | 979 | 973 | 965 | 941 | 929 | 922 | 921 | 910 | 907 | 899 | 892 | 892 | 890 |

Como mencionado, o algoritmo ficou estagnado durante 52 gerações, nas quais foram gerados cerca de 20% dos indivíduos da população, o que corresponde a mais de 2000 indivíduos, com valor de *makespan* igual à 919, como pode ser observado no histograma (Figura 5.4).Esse comportamento pode indicar perda de diversidade populacional (Capítulo 2, Seção 2.5.2).

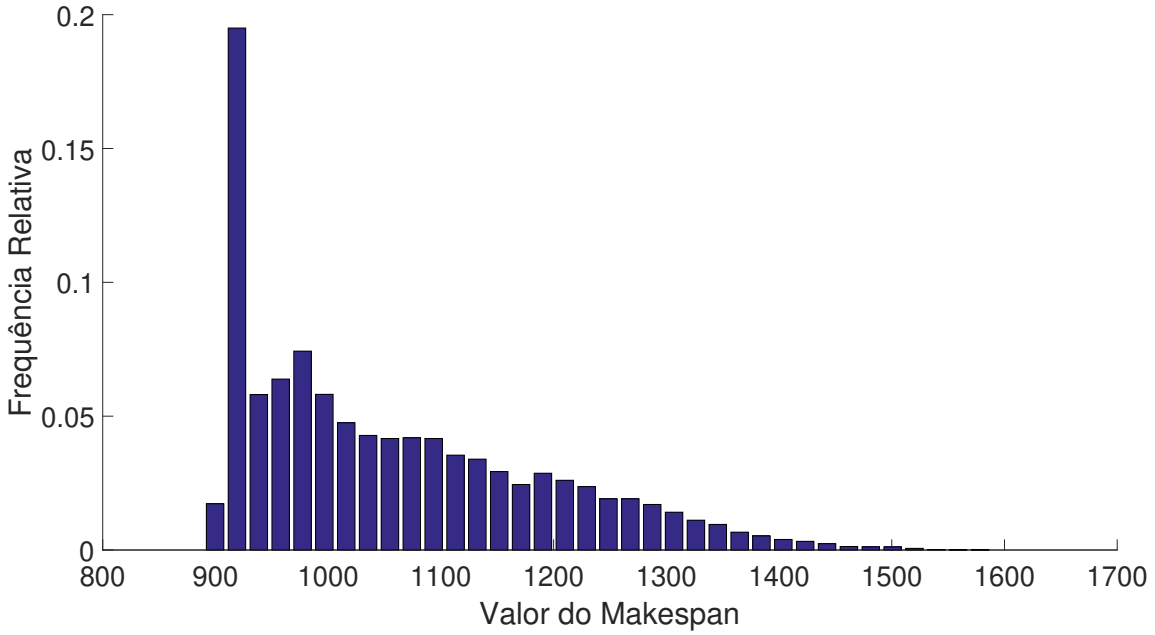



**Figura 5.4:** *Histograma dos indivíduos da população durante evolução do GABL para a instância LA07*

Fonte : A autora.

Embora perceba-se a perda de diversidade populacional para a instância LA07, o

GABL convergiu para o ótimo global com relativa facilidade devido às características do problema e do seu *fitness landscape*.

Na próxima Seção discute-se os resultados obtidos para as instâncias LA03 e FT10 utilizando o GABL, as quais foram classificadas como instâncias difíceis de serem resolvidas (Seção 4.5 do Capítulo 4).

### 5.1.2 Resultados dos experimentos do GABL para as instâncias LA03 e FT10

Nesta Seção são discutidos primeiramente os resultados dos experimentos referente à instância LA03 e a seguir os da instância FT10.

O LA03 é um problema composto por 10 *jobs* e 5 máquinas, portanto, 50 operações e, o seu *makespan* ótimo conhecido é igual a 597. Esse problema é considerado difícil devido às suas características. De acordo com Pérez, Posada e Herrera (2012), o LA03 possui 706 diferentes ótimos globais conhecidos distribuídos no espaço de soluções.

Conforme apresentado na Tabela 5.1, o valor do *makespan* encontrado pelo GABL foi de 604 em 200 gerações. A análise dos dados são apresentados na Figura 5.5. Essa Figura mostra a evolução da população com os valores do *makespan* ótimo local encontrado a cada geração, bem como, os valores da média e da mediana dos valores do *makespan* de toda população por geração.

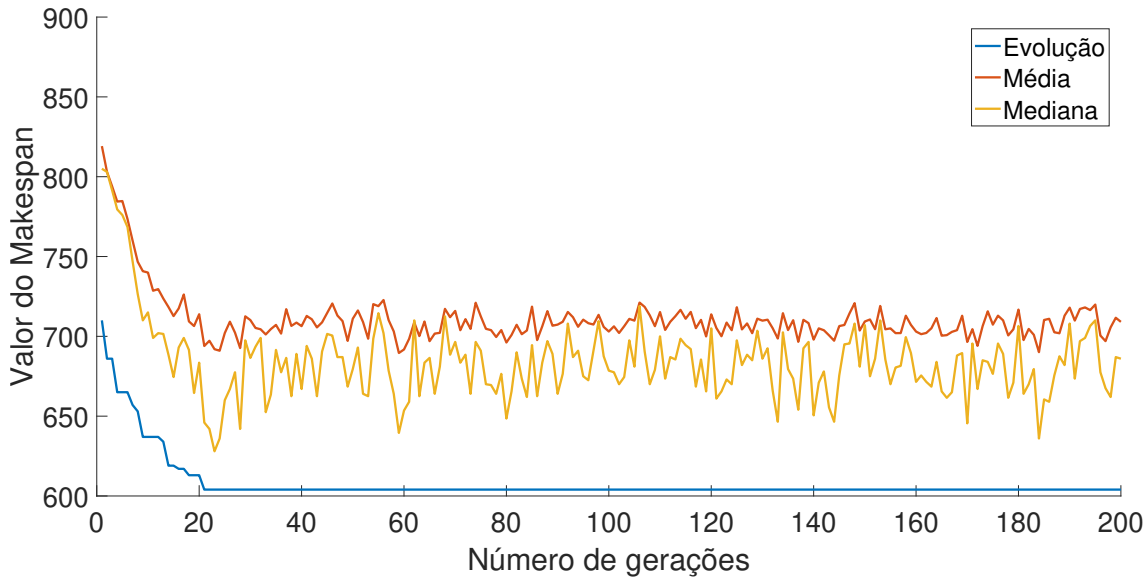



**Figura 5.5:** *Média, mediana e evolução da população da instância LA03 a cada geração do GABL.*

Fonte : A autora.

Nota-se que o algoritmo ficou estagnado, convergindo prematuramente para um ótimo local na 22ª geração com um valor de *makespan* igual à 604, indicando que o algoritmo

ficou "preso" em uma região ótima local do espaço de busca. Também observa-se que os valores da mediana começam a se afastar da média na 10ª geração. A Tabela 5.4 apresenta o número da geração e os valores do *makespan* das soluções que foram melhoradas durante a evolução da população para a instância LA03 conforme apresentado na Figura 5.5.

**Tabela 5.4:** *Evolução da população para a instância LA03: número da geração e valor do makespan no qual o GABL melhorou a solução.*

| Geração | 1 | 2 | 3 | 5 | 8 | 9 | 10 | 14 | 15 | 17 | 19 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MKP | 720 | 710 | 686 | 665 | 657 | 653 | 637 | 634 | 619 | 617 | 613 | 604 |

Dando sequência na análise dos dados do LA03, o histograma apresentado na Figura 5.6 revela a presença de aproximadamente 26,5% de indivíduos da população com valor de *makespan* igual a 604. Esses indivíduos predominaram durante a evolução da população, indicando a presença de muitos indivíduos semelhantes, e que essas soluções podem estar em regiões do espaço de busca no qual estão localizadas as bacias de atração (Seção 2.3.1 do Capitulo 3).

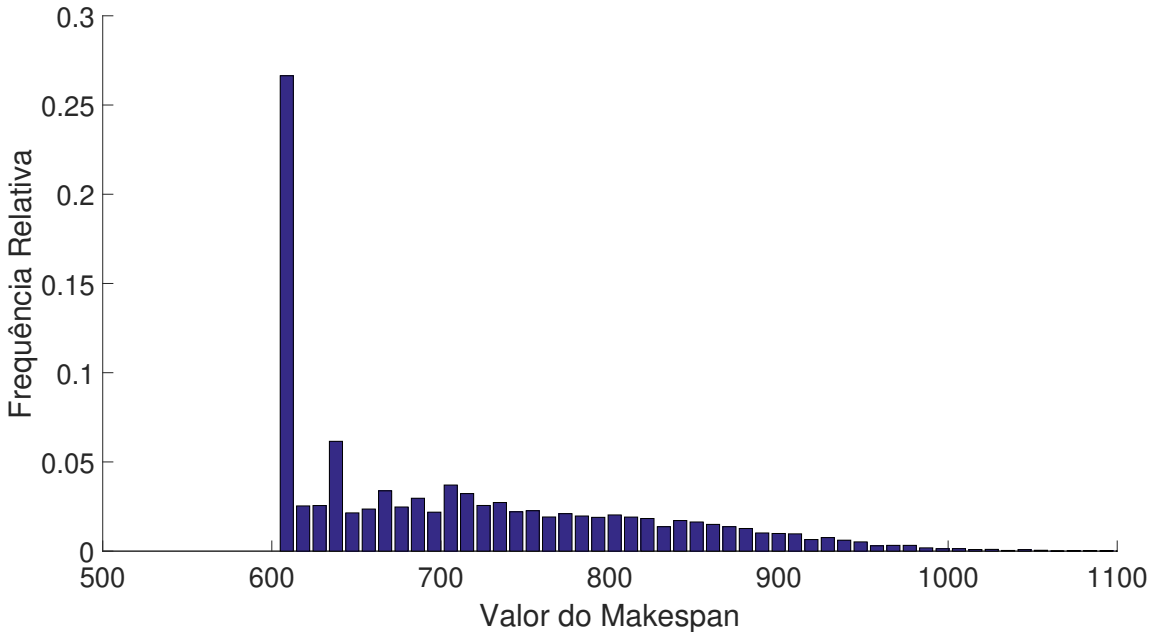



**Figura 5.6:** *Histograma dos indivíduos da população durante evolução do GABL para a instância LA03*

Fonte : A autora.

Como mencionado anteriormente nas configurações do algoritmo (Capítulo 4, Seção 4.6), o tamanho da população é igual a duas vezes o tamanho do cromossomo, que por sua vez é igual à $n \times m$. No caso do LA03, o cromossomo é igual à 50 e, portanto, a população é formada por 100 indivíduos (cromossomo) que foram evoluídos e/ou modificados pelos

operadores genéticos ao longo das 200 gerações. Foram avaliados vinte mil indivíduos e, aproximadamente, 25% deles tem valor de *makespan* igual a 604, ou seja, foram gerados 5000 indivíduos com valor de *makespan* iguais, ficando evidente a perda da diversidade populacional do GABL.

Analisando os dados da instância FT10, o mesmo comportamento da média e mediana observados nos problemas anteriores se repete também para o FT10 conforme apresentado na Figura 5.7.

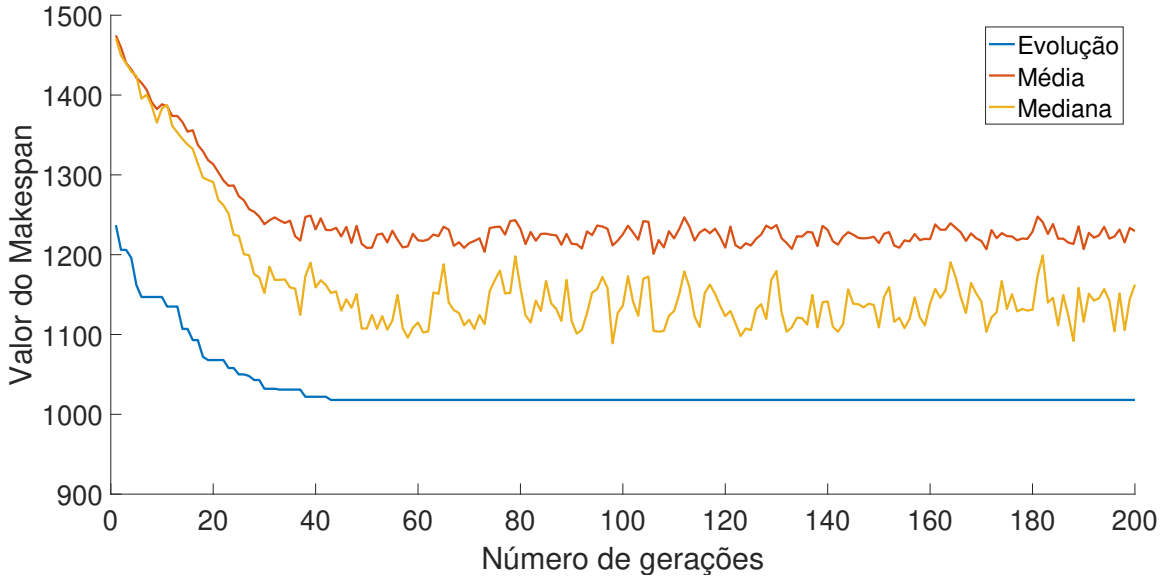



**Figura 5.7:** *Média, mediana e evolução da população da instância FT10 a cada geração do GABL.*

Fonte : A autora.

O método GABL apresentou uma convergência prematura da solução para um ótimo local com valor de *makespan* igual à 1018 na 43ª geração. A mediana começa a se afastar da média por volta da trigésima geração e, à partir da 43ª geração o algoritmo ficou estagnado. Esse fato conduz para a interpretação da falta de diversidade populacional assim como nas instâncias analisadas anteriormente, e também para possíveis bacias de atração na região do espaço de busca.

Para um resumo da evolução do GABL para o FT10, a Tabela 5.5 apresenta o número da geração e os valores do *makespan* das soluções que foram melhoradas conforme apresentado na Figura 5.7.

**Tabela 5.5:** *Evolução da população para a instância FT10: número da geração e valor do makespan no qual o GABL melhorou a solução.*

| **Geração** | 1 | 2 | 4 | 5 | 6 | 11 | 14 | 16 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|
| **MKP** | 1237 | 1206 | 1196 | 1162 | 1147 | 1135 | 1107 | 1093 | 1072 | 1068 |
| **Geração** | 23 | 24 | 25 | 27 | 28 | 30 | 33 | 38 | 43 | |
| **MKP** | 1058 | 1058 | 1050 | 1048 | 1043 | 1032 | 1031 | 1022 | 1018 | |

O GABL gerou cerca de 7800 indivíduos com *makespan* igual à 1018, o que significa aproximadamente 20% dos indivíduos da população em todas as gerações. A Figura 5.8 apresenta o histograma dos indivíduos da população durante evolução do GABL para a instância FT10.

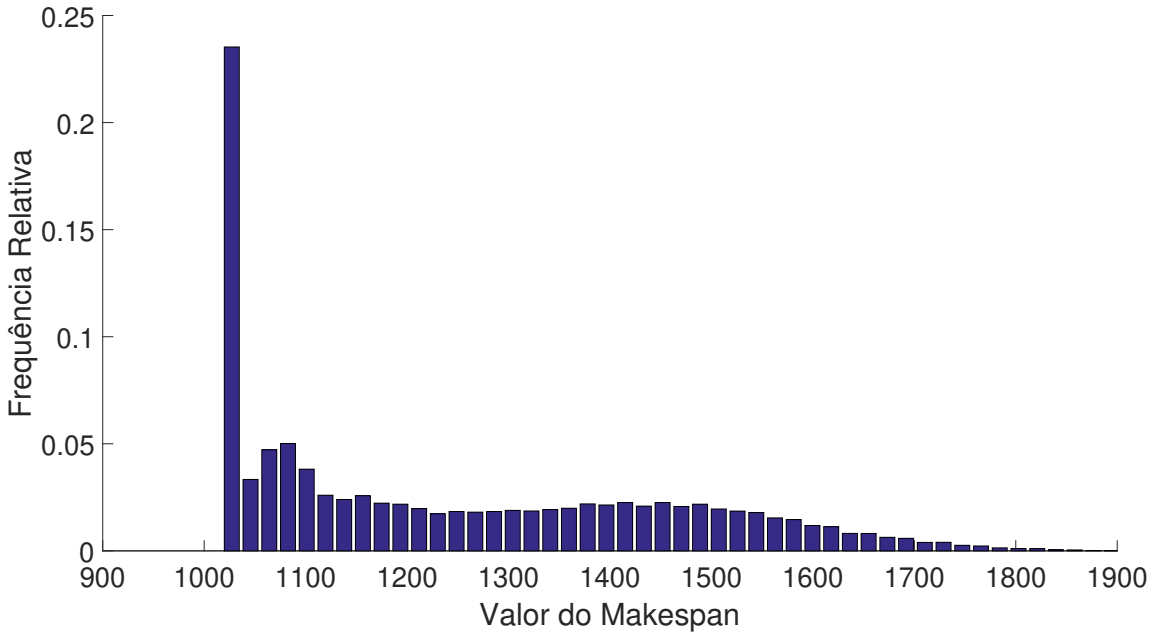



**Figura 5.8:** *Histograma dos indivíduos da população durante evolução do GABL para a instância FT10*

Fonte : A autora.

Por meio da análise das medidas estatísticas de centralidade média e mediana realizadas para as instâncias FT06, LA07, LA03 e FT10 foi possível observar que quando a mediana dos valores de *makespan* de todos os indivíduos da população por geração se afasta da média, pode-se relacionar com a perda da diversidade populacional.

Ainda, para as instâncias que o GABL não convergiu para o *makespan* ótimo conhecido (Tabela 5.1), fica claro que somente a estratégia de busca local não foi suficiente para encontrar melhores soluções. Ademais, foi observado a perda de diversidade do algoritmo justificando assim, a aplicação de métodos para diversificação da população para melhorar o processo de busca considerando-se tanto as características do *fitness landscape* quanto

do próprio problema.

Como mencionado, neste trabalho foi proposto o procedimento denotado por $d$HGA para melhorar a diversidade populacional do algoritmo (Capítulo 4, Seção 4.4.1). Os resultados dos experimentos computacionais são apresentados à seguir (Seção 5.2).

## 5.2 Análise dos resultados experimentais do HGA com abordagem de diversificação

A abordagem de diversificação proposta foi implementada no procedimento computacional $d$HGA. Os resultados apresentados a seguir referem-se à segunda etapa dos experimentos conforme mencionado na Seção 4.5 do Capítulo 4. O objetivo aqui é testar a proposta de diversificação comparando os resultados aos obtidos anteriormente (GABL).

Como foi observado na análise dos resultados do GABL existe a necessidade de introduzir uma forma de diversidade populacional para que a solução não fique parada em um ótimo local. Ademais, vale frisar que, nesta etapa dos experimento está sendo testado somente o procedimento $d$HGA, ou seja, sem a parte que realiza a intensificação das buscas, o que será apresentado nos resultados da Seção 5.3.

No procedimento $d$HGA os indivíduos são classificados de acordo com um índice de contribuição $CI$ conforme apresentado na Seção 4.4.1.1 do Capítulo 4. Dependendo do valor desse índice $CI$, o indivíduo poderá ser substituído na população por um outro gerado aleatoriamente, o que acontece somente nesta etapa a fim de comparação de resultados. Porém, nesse processo podem existir indivíduos que pertencem à população elite e foram classificados para serem substituídos. Para não eliminar indivíduos bons, 30% dos indivíduos do conjunto elite são mantidos.

Nesta Seção são discutidos os resultados utilizando a abordagem de diversificação proposta para todas as instâncias conforme informado anteriormente. Após essa discussão mais abrangente, apresenta-se a discussão das instâncias selecionadas para serem objetos de estudo mais detalhado FT06, FT10, LA03 e LA07.

Os resultados obtidos pelo procedimento $d$HGA para todas as instâncias são apresentados na Tabela 5.6 e são comparados com os resultados do GABL (Tabela 5.1 apresentado anteriormente).

Na Tabela 5.6 são listados os nomes das instâncias, o tamanho ($n \times m$), o valor do *makespan* da solução ótima conhecido (BKS), os valores do *makespan* (MKP) e o número de gerações realizadas pelo GABL e por fim o valor do *makespan* (MKP) e número de gerações realizadas pelo $d$HGA. Os valores coloridos de azul da coluna $d$HGA (MKP e Gerações) desta Tabela indicam que houve melhora nos valores do *makespan* e/ou número de gerações em relação ao GABL; da mesma forma, os valores coloridos de vermelho indicam que houve uma piora do *makespan* e/ou no número de gerações em relação ao GABL.

**Tabela 5.6:** *Comparação dos resultados entre o GABL e dHGA.*

| Instância | Tamanho | Operações | BKS | GABL | | *d*HGA | |
|---|---|---|---|---|---|---|---|
| | | | | MKP | Gerações | MKP | Gerações |
| **FT06** | **6 × 6** | **36** | **55** | **55** | **8** | **55** | **2** |
| FT10 | 10 × 10 | 100 | 930 | 1019 | 200 | **968** | 200 |
| **LA01** | **10 × 5** | **50** | **666** | **666** | **8** | **666** | **7** |
| **LA02** | **10 × 5** | **50** | **655** | **655** | **20** | **655** | **25** |
| LA03 | 10 × 5 | 50 | 597 | 604 | 200 | 604 | 200 |
| LA04 | 10 × 5 | 50 | 590 | 598 | 200 | **602** | 200 |
| **LA05** | **10 × 5** | **50** | **593** | **593** | **1** | **593** | **1** |
| **LA06** | **15 × 5** | **75** | **926** | **926** | **7** | **926** | **1** |
| **LA07** | **15 × 5** | **75** | **890** | **890** | **76** | **890** | **16** |
| **LA08** | **15 × 5** | **75** | **863** | **863** | **16** | **863** | **10** |
| **LA09** | **15 × 5** | **75** | **951** | **951** | **8** | **951** | **5** |
| **LA10** | **15 × 5** | **75** | **958** | **958** | **3** | **958** | **1** |
| **LA11** | **20 × 5** | **100** | **1222** | **1222** | **16** | **1222** | **6** |
| **LA12** | **20 × 5** | **100** | **1039** | **1039** | **12** | **1039** | **6** |
| **LA13** | **20 × 5** | **100** | **1150** | **1150** | **10** | **1150** | **10** |
| **LA14** | **20 × 5** | **100** | **1292** | **1292** | **1** | – | – |
| **LA15** | **20 × 5** | **100** | **1207** | **1207** | **34** | **1207** | **36** |
| LA16 | 10 × 10 | 100 | 945 | 982 | 200 | **945** | **69** |
| LA17 | 10 × 10 | 100 | 784 | 793 | 200 | **787** | 200 |
| LA18 | 10 × 10 | 100 | 848 | 861 | 200 | **848** | **98** |
| LA19 | 10 × 10 | 100 | 842 | 874 | 200 | **868** | 200 |
| LA20 | 10 × 10 | 100 | 902 | 907 | 200 | **913** | 200 |
| LA21 | 15 × 10 | 150 | 1046 | 1086 | 200 | **1099** | 200 |
| LA22 | 15 × 10 | 150 | 927 | 982 | 200 | **967** | 200 |
| **LA23** | **15 × 10** | **150** | **1032** | **1032** | **70** | **1032** | **115** |
| LA24 | 15 × 10 | 150 | 935 | 1002 | 200 | **996** | 200 |
| LA25 | 15 × 10 | 150 | 977 | 1029 | 200 | **1040** | 200 |
| LA26 | 20 × 10 | 200 | 1218 | 1240 | 200 | **1256** | 200 |
| LA27 | 20 × 10 | 200 | 1235 | 1311 | 200 | **1300** | 200 |
| LA28 | 20 × 10 | 200 | 1216 | 1251 | 200 | **1271** | 200 |
| LA29 | 20 × 10 | 200 | 1157 | 1250 | 200 | **1252** | 200 |
| LA30 | 30 × 10 | 300 | 1355 | 1369 | 200 | **1400** | 200 |
| **LA31** | **30 × 10** | **300** | **1784** | **1784** | **118** | **1784** | **163** |
| **LA32** | **30 × 10** | **300** | **1850** | **1850** | **138** | **1850** | **147** |
| **LA33** | **30 × 10** | **300** | **1719** | **1719** | **195** | **1719** | **156** |

Dentre as trinta e cinco instâncias listadas, o GABL encontrou *makespan* ótimo para dezoito delas (Tabela 5.1). Por outro lado, o procedimento *d*HGA foi utilizado em trinta e quatro instâncias, pois a instância LA14 foi resolvida otimamente sem a necessidade de invocar a diversificação (o procedimento *d*HGA). Assim, o *d*HGA resolveu dezenove das trinta e quatro instâncias testadas, aproximadamente 56%, correspondendo e cerca de 4% à mais das instâncias quando comparado com o GABL.

O *Gap* da solução encontrada (SE) tanto pelo GABL quanto pelo *d*HGA em relação à melhor solução conhecida (BKS) é apresentado na Tabela 5.7, para as instâncias que não foi encontrada a solução ótima. Os valores realçados em negrito destacam qual das abordagens encontrou uma solução com valor do *makespan* mais próximo do *makespan* ótimo. O *Gap* foi calculado da seguinte forma: $Gap = |SE - BKS|/BKS * 100$.

**Tabela 5.7:** *Gap da solução encontrada pelo o GABL e dHGA em relação ao BKS.*

| **Instância** | **BKS** | **GABL** | | ***d*HGA** | |
|---|---|---|---|---|---|
| | | **MKP** | ***Gap* %** | **MKP** | ***Gap* %** |
| FT10 | 930 | 1019 | 9,57 | 968 | **4,08** |
| LA03 | 597 | 604 | 1,17 | 604 | 1,17 |
| LA04 | 590 | 598 | **1,35** | 602 | 2,03 |
| LA16 | 945 | 982 | 3,91 | 945 | **0,00** |
| LA17 | 784 | 793 | 1,14 | 787 | **0,38** |
| LA18 | 848 | 861 | 1,53 | 848 | **0,00** |
| LA19 | 842 | 874 | 3,80 | 868 | **3,08** |
| LA20 | 902 | 907 | **0,55** | 913 | 1,22 |
| LA21 | 1046 | 1086 | **3,82** | 1099 | 5,06 |
| LA22 | 927 | 982 | 5,93 | 967 | **4,31** |
| LA24 | 935 | 1002 | 7,16 | 996 | **6,52** |
| LA25 | 977 | 1029 | **5,32** | 1040 | 6,44 |
| LA26 | 1218 | 1240 | **1,80** | 1256 | 3,11 |
| LA27 | 1235 | 1311 | 6,15 | 1300 | **5,26** |
| LA28 | 1216 | 1251 | **2,87** | 1271 | 4,52 |
| LA29 | 1157 | 1250 | **8,03** | 1252 | 8,21 |
| LA30 | 1355 | 1369 | **1,03** | 1400 | 3,31 |

Adicionalmente, observou-se no *d*HGA houve melhora em relação ao número de gerações necessárias para doze instâncias. Os resultados correspondentes são apresentados na Tabela 5.8. Os números negativos indicam que houve uma piora do *d*HGA em número de gerações em relação ao GABL. Apesar de apresentar número de gerações mais elevado em cinco instâncias, o *d*HGA produziu resultados melhores nesse aspecto. A porcentagem de

melhora, nesse caso, foi calculada da seguinte forma: $Red = (G_{GABL} - G_{dHGA})/G_{GABL} * 100$, no qual $Red$ representa o porcentual de redução e $G$ o número de gerações.

**Tabela 5.8:** *Quantidade de gerações do dHGA em relação ao GABL.*

| **Instância** | $G$ **do GABL** | $G$ **do** $d$**HGA** | **% Melhora** |
|---|---|---|---|
| FT06 | 8 | 2 | **75,00** |
| LA01 | 8 | 7 | **12,50** |
| LA02 | 20 | 25 | -25,00 |
| LA06 | 7 | 1 | **85,72** |
| LA07 | 76 | 16 | **78,95** |
| LA08 | 16 | 10 | **37,50** |
| LA09 | 8 | 5 | **37,50** |
| LA10 | 3 | 1 | **66,66** |
| LA11 | 16 | 6 | **62,50** |
| LA12 | 12 | 6 | **50,00** |
| LA15 | 34 | 36 | -5,88 |
| LA16 | 200 | 69 | **65,50** |
| LA18 | 200 | 98 | **51,00** |
| LA23 | 70 | 115 | -64,28 |
| LA31 | 118 | 163 | -38,13 |
| LA32 | 138 | 147 | -6,52 |
| LA33 | 195 | 156 | **20,00** |

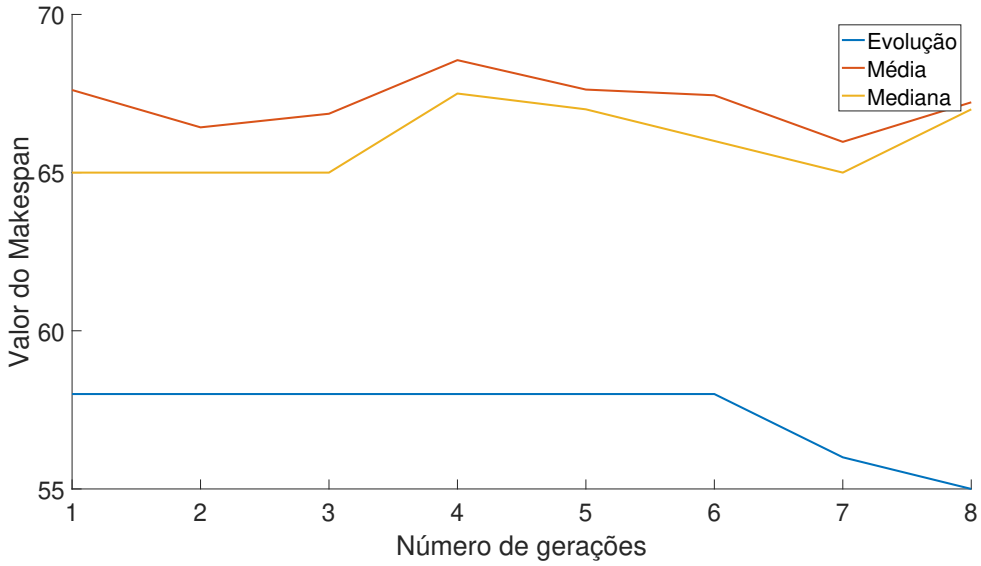
Como apresentado até o momento, o $d$HGA mostrou-se superior ao GABL. As Seções a seguir discutem os resultados das instâncias que são objetos de estudo mais detalhado, sendo FT06, FT10, LA03 e LA07.

### 5.2.1 Resultados dos experimentos do $d$HGA para as instâncias FT06 e LA07

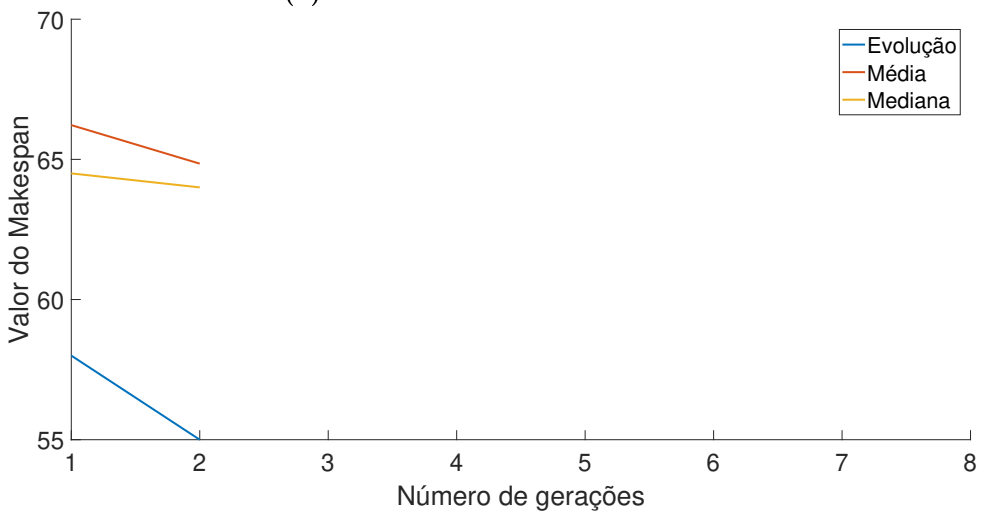
Nesta Seção são apresentados e discutidos os resultados das instâncias FT06 e LA07, nessa ordem. Apresenta-se a análise das medidas estatísticas de centralidade média e mediana bem como a evolução da população durante a execução do algoritmo. As figuras da Seção 5.1 são repetidas nesta Seção e colocadas lado-a-lado para efeitos de comparação entre $d$HGA e GABL e melhor visualização.

Para a instância FT06, o $d$HGA alcançou o *makespan* ótimo em duas gerações. Como visto anteriormente, os dados são escassos principalmente considerando a quantidade de gerações que foram necessárias para alcançar o *makespan* ótimo e, portanto, não é possível

uma análise mais detalhada. A Figura 5.9b apresenta as medidas estatísticas de centralidade obtidas com a abordagem do *d*HGA; a Figura 5.9a apresenta o gráfico do GABL (apresentado na Seção anterior).




**(a)** *Estatística do FT06 com GABL*



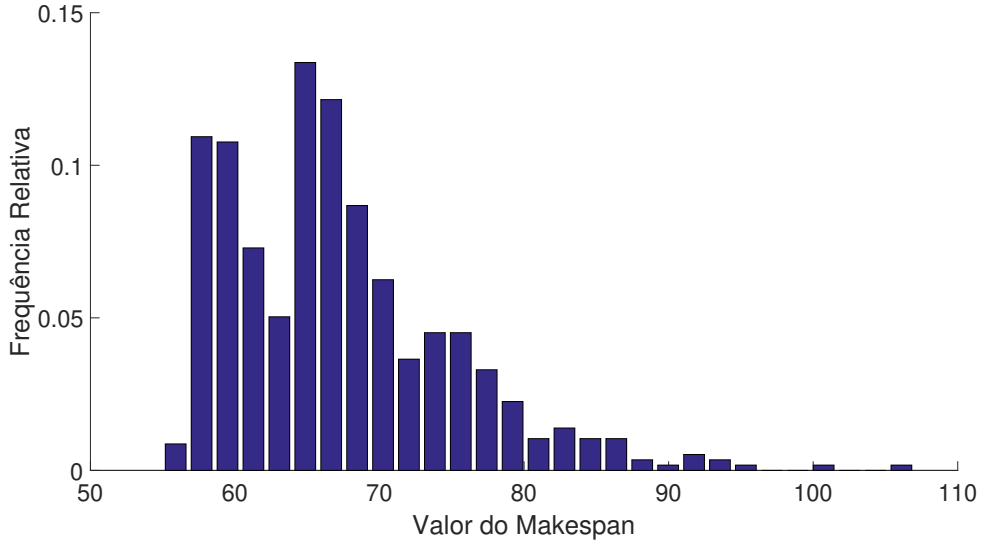

**(b)** *Estatística do FT06 com dHGA*

**Figura 5.9:** *Média, mediana e evolução da população do FT06 a cada geração com as abordagens GABL e dHGA.*

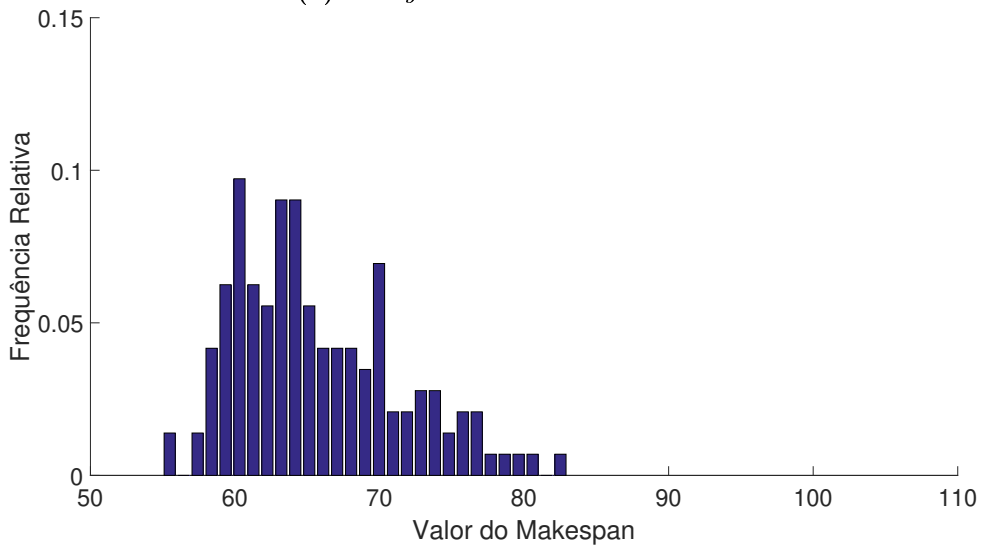
Fonte : A autora.

Analisando o histograma do FT06 apresentado na Figura 5.10b, nota-se que houve uma melhor distribuição dos indivíduos com o *d*HGA quando comparado com o histograma da Figura 5.10a do GABL. Esse fato ocorreu devido à diversidade populacional utilizada pelo *d*HGA. Neste caso, existem 14 indivíduos na população que predominaram com valor de *makespan* igual à 60, o que representa menos de 10% dos indivíduos da população e, também existem dois grupos com cerca de 9% dos indivíduos da população em cada grupo com *makespan* entre 62 e 64. Comparando as duas figuras dos histogramas (Figura 5.10)

fica evidente que houve melhora na diversificação dos indivíduos da população.




**(a)** *Histograma FT06 com GABL*

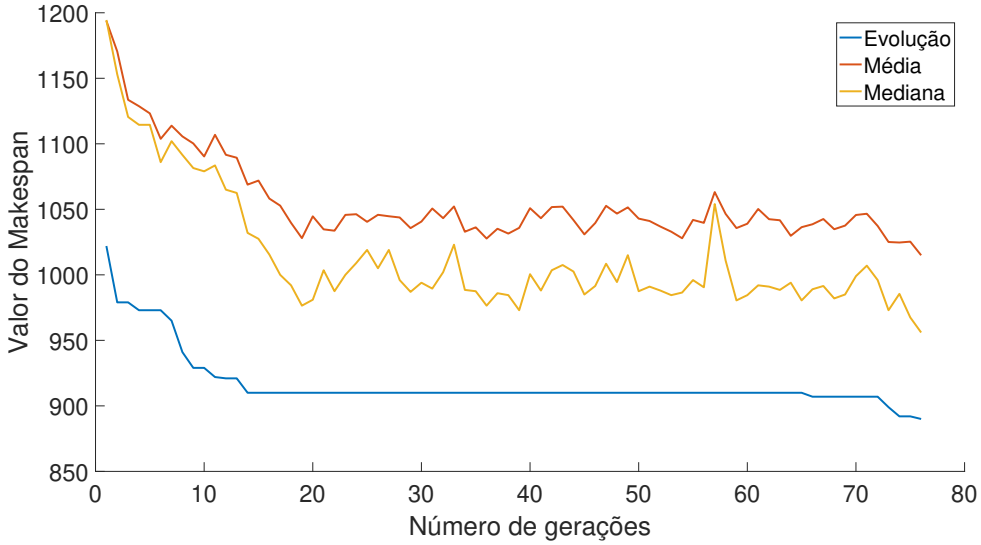



**(b)** *Histograma FT06 com dHGA*

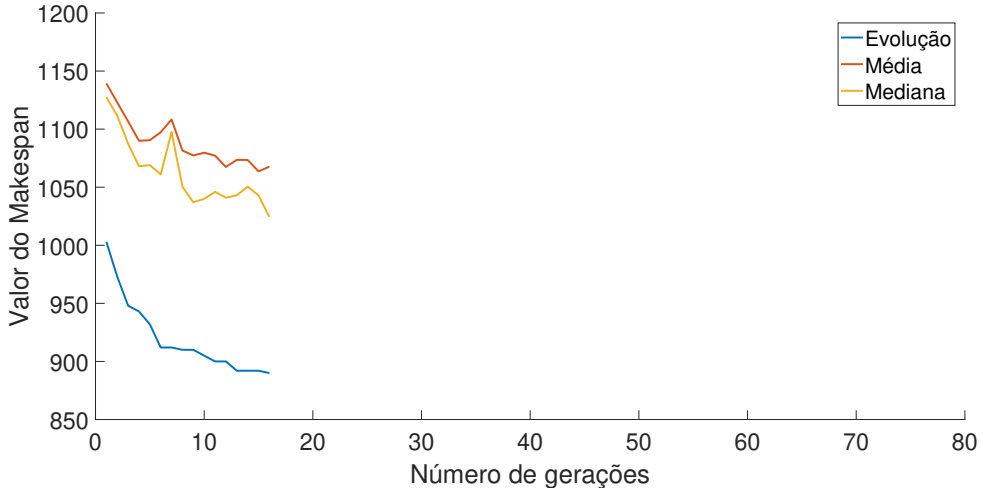
**Figura 5.10:** *Histograma dos indivíduos da população do FT06 com as abordagens GABL e dHGA.*

Fonte : A autora.

Para o LA07 foi encontrada a solução ótima em dezesseis gerações utilizando-se a abordagem de diversificação *d*HGA contra setenta e seis gerações utilizando o GABL, significando um ganho de aproximadamente 79% em número de gerações, conforme apresentado na Tabela 5.8. No gráfico apresentado na Figura 5.11b as medidas da média e mediana permanecem próximas durante a evolução da população.




**(a)** *Estatística do LA07 com GABL*

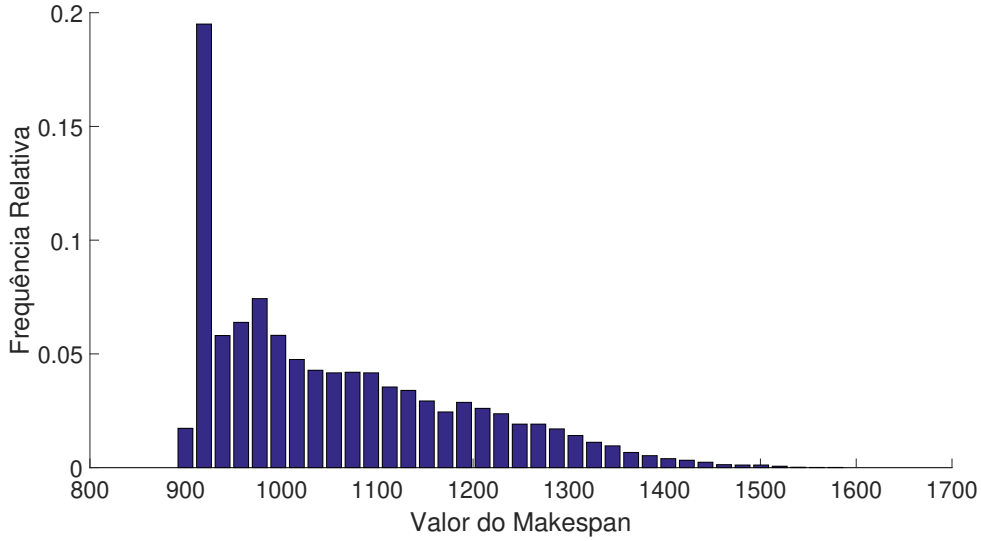



**(b)** *Estatística do LA07 com dHGA*

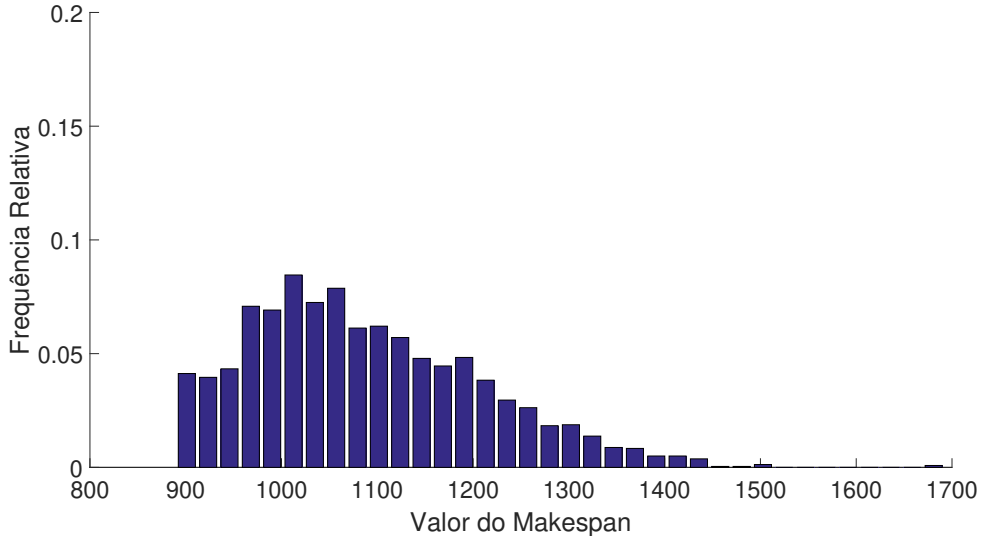
**Figura 5.11:** *Média, mediana e evolução da população do LA07 a cada geração com as abordagens GABL e dHGA.*

Fonte : A autora.

Comparando-se os dois histogramas das abordagens GABL e *d*HGA apresentados na Figura 5.12 fica evidenciado o resultado da diversificação da população utilizando o *d*HGA (Figura 5.12b) em relação ao GABL (Figura 5.12a). Enquanto que com o GABL foram gerados cerca de 20% dos indivíduos da população (aproximadamente 2000 indivíduos) com valor de *makespan* igual a 919, com o *d*HGA foram gerados pouco mais de 250 indivíduos (aproximadamente 4%) com o mesmo valor de *makespan*. Outra observação diz respeito à distribuição da frequência dos indivíduos durante a evolução do algoritmo. Essa diversificação teve resultados positivos visto que o algoritmo convergiu para solução com o *makespan* ótimo em 16 gerações como mencionado anteriormente.




**(a)** *Histograma LA07 com GABL*




**(b)** *Histograma LA07 com dHGA*

**Figura 5.12:** *Histograma dos indivíduos da população do LA07 com as abordagens e GABL e dHGA.*

Fonte : A autora.

A análise dos resultados utilizando a abordagem de diversificação da população $d$HGA para as instâncias LA03 e FT10, consideradas difíceis, são apresentadas a seguir.

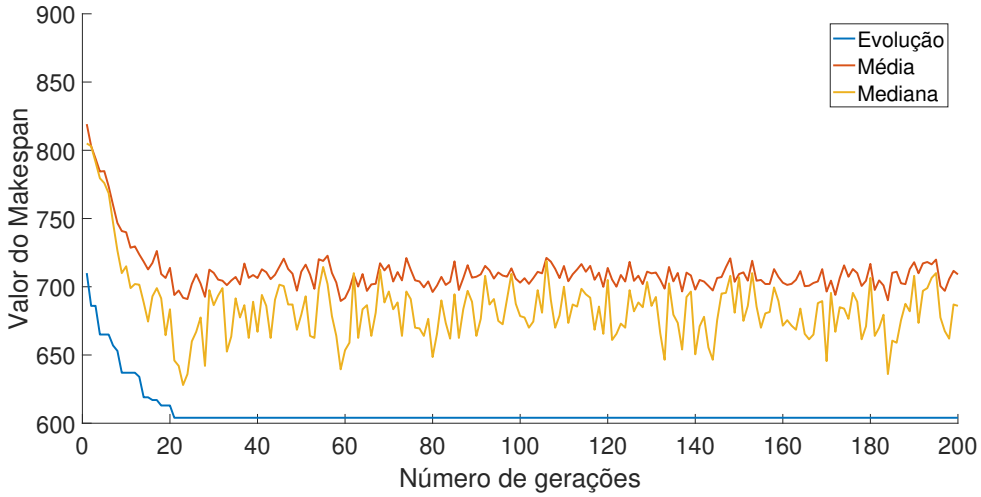
### 5.2.2 Resultados dos experimentos do $d$HGA para as instâncias LA03 e FT10

Como mencionado ao final da Seção 5.1.2 deste Capítulo, foi observado que quando o valor da mediana se afasta da média pode ser um indicador da perda de diversidade. Quando essa perda da diversidade é identificada então aplica-se o procedimento computacional $d$HGA.

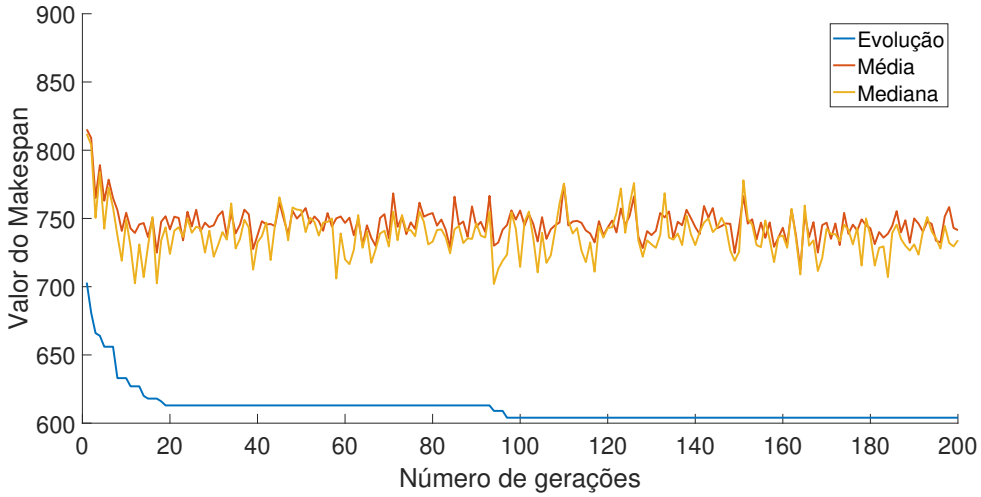
Nos dados apresentados anteriormente para o FT06 e LA07 (Figuras 5.9 e 5.11), o comportamento da média e da mediana não fica muito evidenciado devido ao algoritmo ter convergido para a solução ótima em poucas gerações. Para o LA03 e FT10, como foram gerados mais dados foi possível realizar uma análise mais detalhada conforme é apresentado a seguir.

Analisando os dados da instância LA03, conforme Tabela 5.6, tanto o *d*HGA quanto o GABL realizaram 200 gerações e terminaram com valor do *makespan* de 604, portanto, não alcançaram o ótimo global.

Na Figura 5.13b, nota-se que os valores da média e mediana ficam próximos durante a evolução do algoritmo por todas gerações, o que não acontece quando utiliza-se o GABL (Figura 5.13a), porque quando é detectado a perda da diversidade, aplica-se o procedimento *d*HGA.




**(a)** *Estatística do LA03 com GABL*




**(b)** *Estatística do LA03 com dHGA*

**Figura 5.13:** *Média, mediana e evolução da população do LA03 a cada geração com as abordagens GABL e dHGA.*

Fonte : A autora.

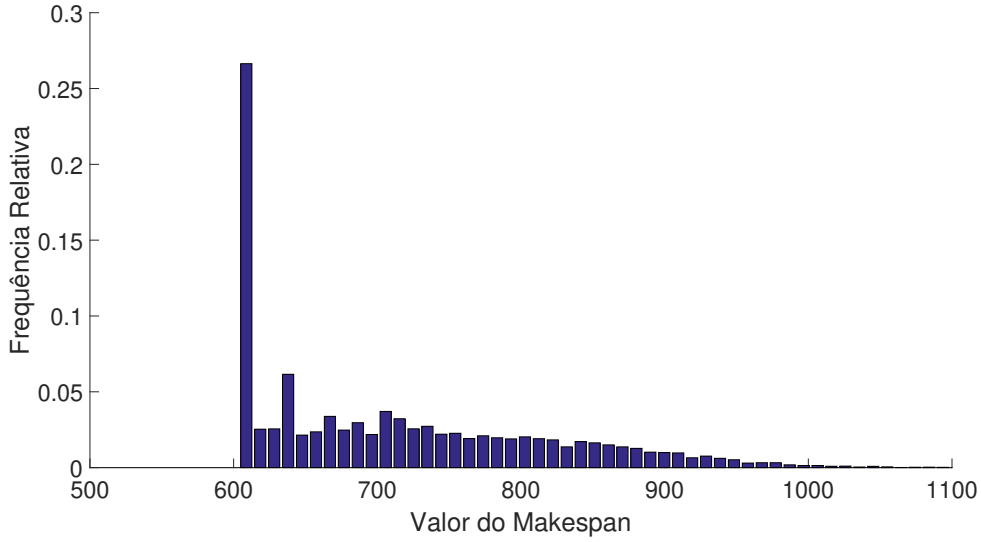
Ainda, é possível notar as diferenças na evolução da população entre o GABL e o *d*HGA (Figura 5.13). Na execução do GABL, o algoritmo ficou estagnado próximo à 20ª geração, enquanto que com o *d*HGA houve dois momentos de estagnação, sendo na 19ª e 97ª gerações. Os indivíduos que predominaram durante a evolução da população podem ser indivíduos muito semelhantes mas que têm *makespan* iguais indicando que as soluções encontram-se em regiões do espaço de busca formado por bacias de atração e, portanto, existe a necessidade de intensificar as buscas para explorar regiões mais promissoras.

A Tabela 5.9 apresenta um resumo da evolução do algoritmo utilizando o *d*HGA, destacando em quais gerações a solução foi melhorada e qual o valor do *makespan* encontrado. Observa-se que até a 19ª geração houve uma boa evolução, mas depois o algoritmo ficou estagnado até a 94ª geração e por último na 97ª geração até o final, conforme já observado anteriormente.

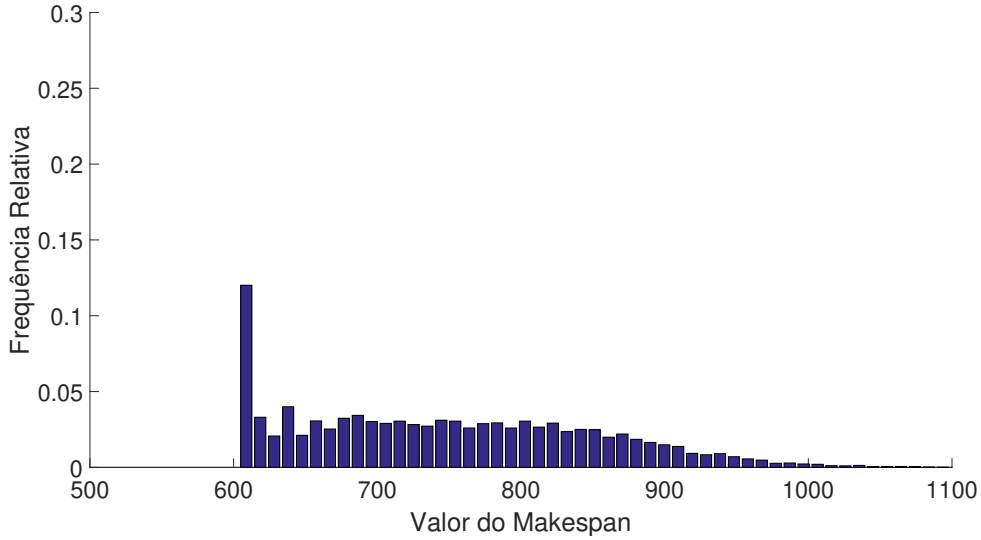
**Tabela 5.9:** *Evolução da população para a instância LA03: número da geração e valor do makespan no qual o dHGA melhorou a solução.*

| Geração | 1 | 2 | 3 | 4 | 5 | 8 | 11 | 14 | 15 | 18 | 19 | 94 | 97 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MKP | 703 | 681 | 666 | 664 | 656 | 633 | 627 | 620 | 618 | 616 | 613 | 609 | 604 |

Analisando o histograma da instância LA03 apresentado na Figura 5.14b observa-se a existência de 12% dos indivíduos da população com *makespan* iguais, indicando que mesmo com a diversificação da população aplicada pelo *d*HGA não foi suficiente para melhorar os indivíduos da população, mas ainda assim, percebe-se que houve melhora da distribuição dos outros indivíduos com a diversificação.




**(a)** *Histograma LA03 com GABL*




**(b)** *Histograma LA03 com dHGA*

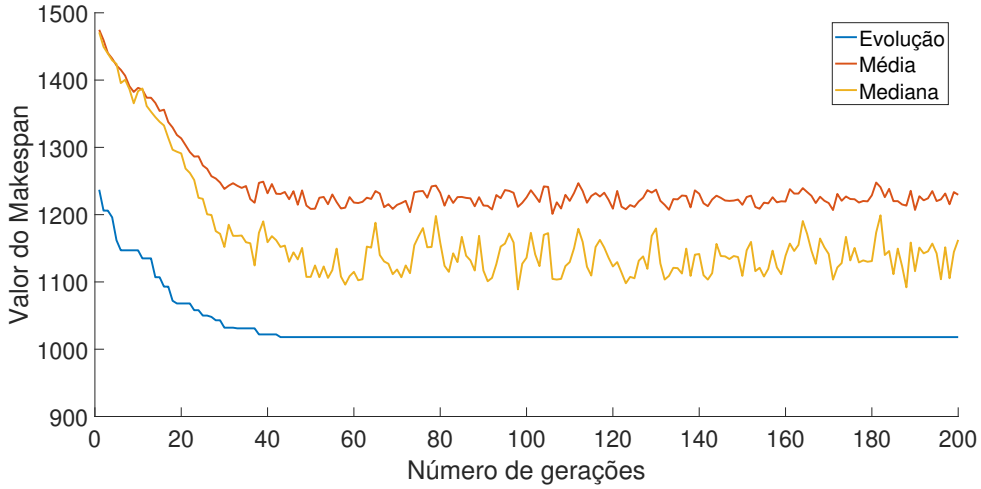
**Figura 5.14:** *Histograma dos indivíduos da população do LA03 com as abordagens e GABL e dHGA.*

Fonte : A autora.

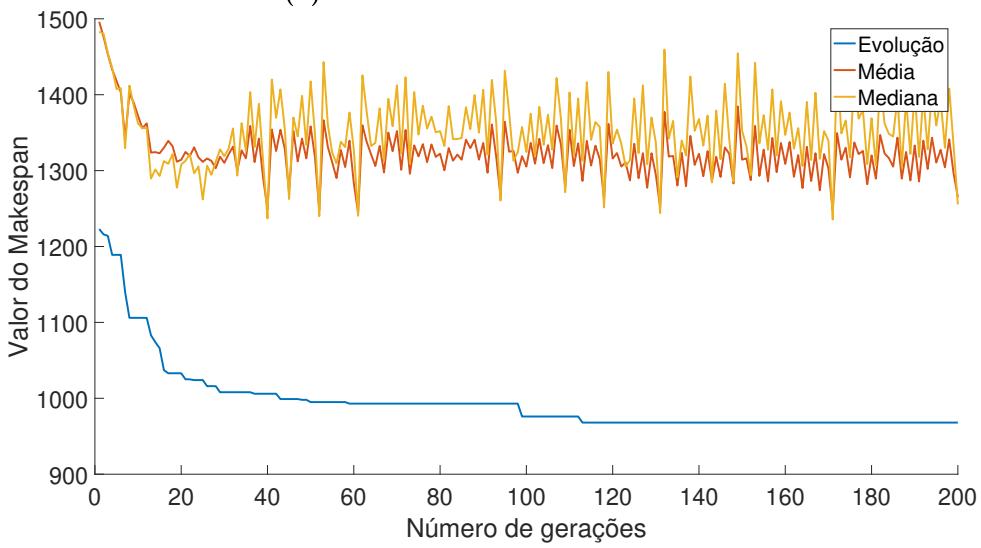
Analisa-se a seguir os dados obtidos pelo algoritmo para a instância FT10 utilizando o procedimento computacional *d*HGA para manter a diversidade populacional. Como verificado anteriormente (Tabela 5.6), o algoritmo obteve a solução final com *makespan* igual à 968, resultado melhor do que o obtido com o GABL. A partir dos dados estatísticos, bem como a evolução da população para o FT10, nota-se o mesmo comportamento relatado anteriormente referente as outras instâncias.

A aplicação do procedimento *d*HGA quando observa-se que os valores da mediana afastam-se da média da população durante as gerações, recompõe-se a diversidade populacional que é novamente refletida no valor das medidas estatísticas. A Figura 5.15b apresenta a evolução da população bem como da média e mediana durante as 200 gerações

do algoritmo.




**(a)** *Estatística do FT10 com GABL*




**(b)** *Estatística do FT10 com dHGA*

**Figura 5.15:** *Média, mediana e evolução da população do FT10 a cada geração com as abordagens GABL e dHGA.*

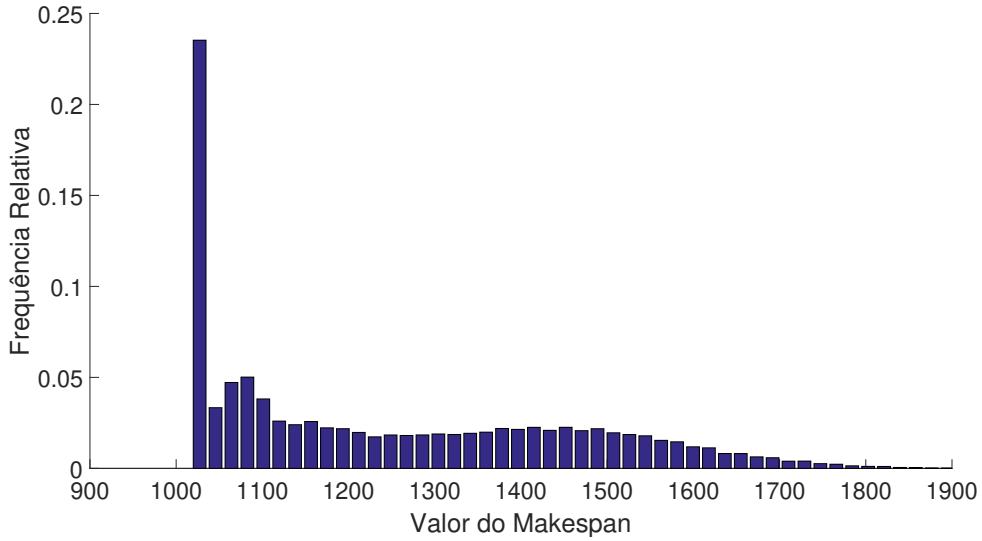
Fonte : A autora.

Como pode-se observar na Figura 5.15 existe diferença do comportamento da média e da mediana entre as abordagens GABL e *d*HGA. Com o GABL houve uma convergência prematura para um ótimo local mais cedo do que a convergência prematura do *d*HGA. Ainda, com a abordagem de diversificação, embora não tenha sido encontrado a solução ótima, mesmo assim houve melhora no resultado final em relação ao GABL como apresentado anteriormente. A Tabela 5.10 apresenta resumidamente a evolução da população do FT10 com a abordagem *d*HGA.

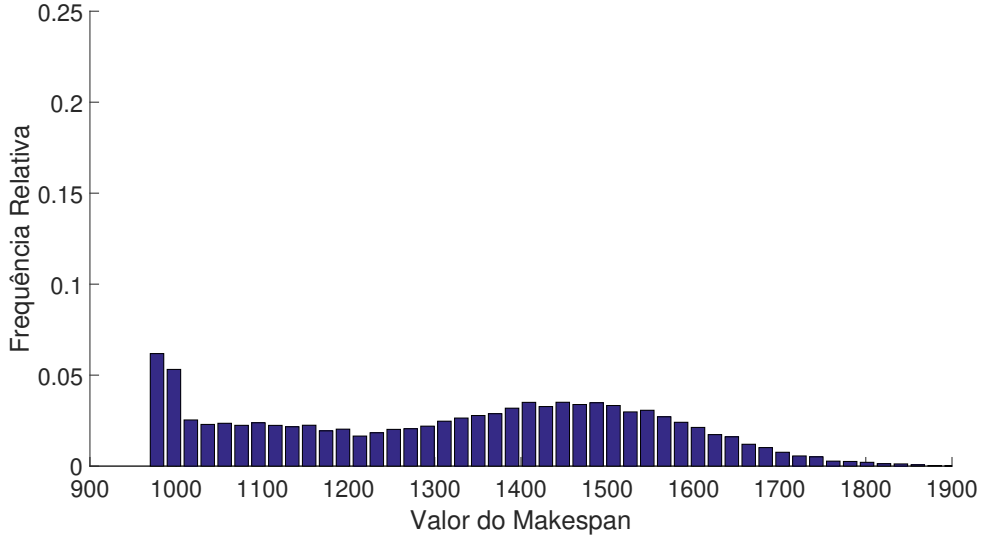
**Tabela 5.10:** *Evolução da população para a instância FT10: número da geração e valor do makespan no qual o dHGA melhorou a solução.*

| Geração | 1 | 2 | 3 | 4 | 7 | 8 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MKP | 1223 | 1216 | 1214 | 1189 | 1140 | 1106 | 1083 | 1074 | 1066 | 1037 | 1033 |
| Geração | 21 | 23 | 26 | 29 | 37 | 43 | 48 | 50 | 59 | 99 | 113 |
| MKP | 1025 | 1024 | 1016 | 1008 | 1006 | 999 | 998 | 995 | 993 | 976 | 968 |

Na Figura 5.16 apresenta-se o histograma para o FT10 utilizando as abordagens GABL e *d*HGA (Figuras 5.16a e 5.16b, respectivamente).




**(a)** *Histograma FT10 com GABL*



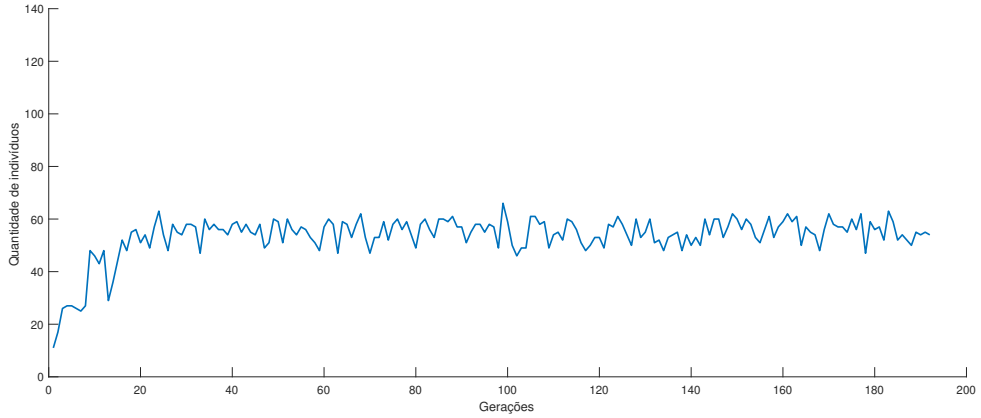

**(b)** *Histograma FT10 com dHGA*

**Figura 5.16:** *Histograma dos indivíduos da população do LA03 com as abordagens GAB e dHGA.*
Fonte : A autora.

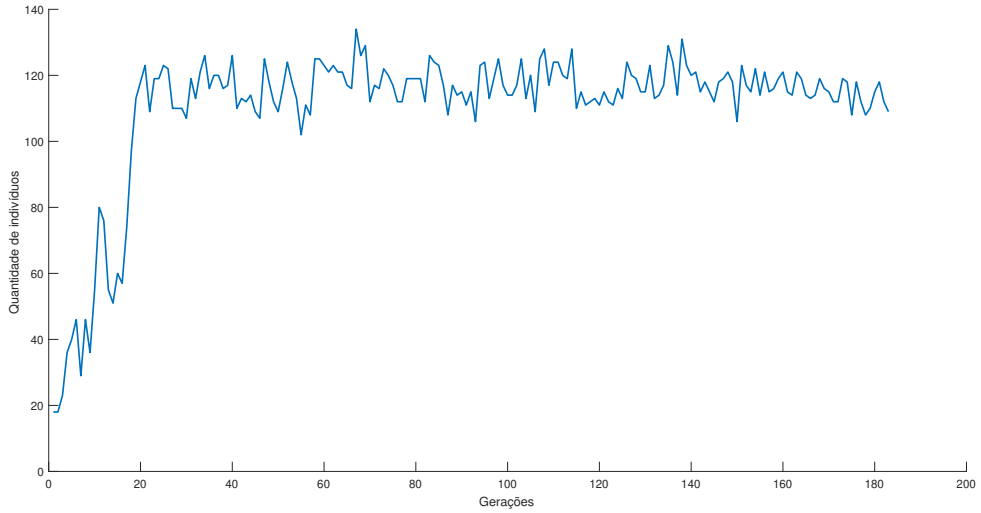
Novamente nota-se que houve melhor distribuição dos indivíduos e principalmente uma diferença em número de indivíduos que predominaram a população, sendo que com a abordagem $d$HGA foram gerados aproximadamente 6% indivíduos com valores de *makespan* na faixa de 968 e 980 e, com o GABL, foram gerados aproximadamente 23% dos indivíduos com valores de *makespan* na faixa de 1020.

Durante a execução do algoritmo utilizando o procedimento computacional $d$HGA para os problemas listados anteriormente, foi observado que no início das gerações do algoritmo, o número de indivíduos para serem substituídos na população de acordo com o seu índice de contribuição $CI$, dependendo da instância, era pequeno, o que significa que nas primeiras gerações a diversidade populacional é mantida. Com a evolução das gerações do algoritmo, notou-se que o número de indivíduos para serem substituídos aumenta.

Com essa informação relatada, foi executado o algoritmo sem efetuar a substituição dos indivíduos na população, somente para contabilizar a quantidade de indivíduos com índice de contribuição $CI$ que deveriam ser substituídos na população para manter diversidade, conforme apresentado na Figura 5.17 para as instâncias LA03 (Figura 5.17a) e FT10 (5.17b). Nota-se que para ambas as instâncias há um crescimento na quantidade de indivíduos logo nas primeiras gerações e depois esse número se mantém. Ainda vale comentar que, para o LA03, com uma população de 200 indivíduos, são classificados uma média de 55 indivíduos e, para o FT10, com 200 indivíduos formando a população, aproximadamente 120 indivíduos são classificados para serem substituídos.




**(a)** *Instância LA03*




**(b)** *Instância FT10*

**Figura 5.17:** *Quantidade de indivíduos selecionados para possível substituição na população a cada geração.*

Fonte : A autora.

## 5.3 Análise dos Resultados experimentais do HGA com abordagem de diversificação e intensificação

As abordagens de diversificação com intensificação propostas foram implementadas nos procedimentos $d$HGA e $i$HGA (Capítulo 4, Seções 4.4.1 e 4.4.2, respectivamente). Os resultados apresentados a seguir referem-se a etapa 3 do planejamento dos experimentos (Capítulo 4, Seção 4.5). O principal objetivo é analisar os resultados obtidos com o HGA e comparar com os das duas etapas anteriores.

Como visto, somente abordagens de busca local e/ou diversificação da população ainda não foram suficientes para melhorar os resultados, ou para encontrar soluções com valor de *makespan* ótimo, ou ainda reduzir o número de gerações necessárias para alcançar uma solução ótima ou subótima para o JSSP.

A abordagem de diversificação com intensificação ($d$HGA com $i$HGA), difere da abor-

dagem somente com diversificação ($d$HGA) na forma como os indivíduos classificados pelo seu índice de contribuição $CI$ são substituídos na população.

Enquanto que na abordagem somente com diversificação foram gerados indivíduos aleatoriamente para substituir àqueles selecionados pelo índice $CI$, na diversificação com intensificação o indivíduo selecionado pelo índice $CI$ é substituído pelo melhor indivíduo encontrado pelo método *Path Relinking*. Neste caso então, o indivíduo classificado de acordo com seu índice $CI$ para ser removido/substituído na população, é a solução inicial para a construção da trajetória do *Path Relinking*, conforme explicado na Seção 4.4.2 do Capítulo 4.

A Tabela 5.6 apresenta os resultados obtidos pelo HGA, bem como os resultados das abordagens anteriores, listando-se os nomes das instâncias, o tamanho ($n \times m$), o valor do *makespan* ótimo conhecido (BKS), os valores de *makespan* e número de gerações realizadas pelo GABL, os valores do *makespan* encontrados pelo $d$HGA e número de gerações e, por fim os valores do *makespan* e número de gerações do HGA. Os valores coloridos em azul nas colunas $d$HGA e HGA da Tabela, indicam que os resultados melhoraram ou o valor do *makespan* e/ou no número de gerações e, os valores coloridos em vermelho indicam que os resultados foram piores ou no valor do *makespan* e/ou em número de gerações. As comparações foram realizadas entre o HGA com diversificação e intensificação ($d$HGA e $i$HGA), HGA somente com a diversificação ($d$HGA) e GABL.

**Tabela 5.11:** *Comparação dos resultados entre o GABL, dHGA e HGA (dHGA com iHGA).*

| Instância | Tamanho | Operações | BKS | GABL | | *d*HGA | | HGA | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MKP | Gerações | MKP | Gerações | MKP | Gerações |
| **FT06** | **6 × 6** | **36** | **55** | **55** | **8** | **55** | **2** | **55** | **2** |
| FT10 | 10 × 10 | 100 | 930 | 1019 | 200 | **968** | 200 | **937** | 200 |
| **LA01** | **10 × 5** | **50** | **666** | **666** | **8** | **666** | **7** | **666** | **4** |
| **LA02** | **10 × 5** | **50** | **655** | **655** | **20** | **655** | **25** | **655** | **19** |
| LA03 | 10 × 5 | 50 | 597 | 604 | 200 | 604 | 200 | **603** | 200 |
| **LA04** | 10 × 5 | 50 | 590 | 598 | 200 | **602** | 200 | **590** | **44** |
| **LA05** | **10 × 5** | **50** | **593** | **593** | **1** | **593** | **1** | **593** | **1** |
| **LA06** | **15 × 5** | **75** | **926** | **926** | **7** | **926** | **1** | **926** | **2** |
| **LA07** | **15 × 5** | **75** | **890** | **890** | **76** | **890** | **16** | **890** | **14** |
| **LA08** | **15 × 5** | **75** | **863** | **863** | **16** | **863** | **10** | **863** | **6** |
| **LA09** | **15 × 5** | **75** | **951** | **951** | **8** | **951** | **5** | **951** | **3** |
| **LA10** | **15 × 5** | **75** | **958** | **958** | **3** | **958** | **1** | **958** | **1** |
| **LA11** | **20 × 5** | **100** | **1222** | **1222** | **16** | **1222** | **6** | **1222** | **8** |
| **LA12** | **20 × 5** | **100** | **1039** | **1039** | **12** | **1039** | **6** | **1039** | **5** |
| **LA13** | **20 × 5** | **100** | **1150** | **1150** | **10** | **1150** | **10** | **1150** | **9** |
| **LA14** | **20 × 5** | **100** | **1292** | **1292** | **1** | – | – | **1292** | **1** |
| **LA15** | **20 × 5** | **100** | **1207** | **1207** | **34** | **1207** | **36** | **1207** | **31** |
| **LA16** | 10 × 10 | 100 | 945 | 982 | 200 | **945** | **69** | **946** | **200** |
| LA17 | 10 × 10 | 100 | 784 | 793 | 200 | **787** | 200 | **789** | 200 |
| **LA18** | 10 × 10 | 100 | 848 | 861 | 200 | **848** | **98** | **855** | 200 |
| LA19 | 10 × 10 | 100 | 842 | 874 | 200 | **868** | 200 | **875** | 200 |
| LA20 | 10 × 10 | 100 | 902 | 907 | 200 | **913** | 200 | **907** | 200 |
| LA21 | 15 × 10 | 150 | 1046 | 1086 | 200 | **1099** | 200 | **1095** | 200 |
| LA22 | 15 × 10 | 150 | 927 | 982 | 200 | **967** | 200 | **946** | 200 |
| **LA23** | **15 × 10** | **150** | **1032** | **1032** | **70** | **1032** | **115** | **1032** | **134** |
| LA24 | 15 × 10 | 150 | 935 | 1002 | 200 | **996** | 200 | **990** | 200 |
| LA25 | 15 × 10 | 150 | 977 | 1029 | 200 | **1040** | 200 | **1043** | 200 |
| LA26 | 20 × 10 | 200 | 1218 | 1240 | 200 | **1256** | 200 | **1238** | 200 |
| LA27 | 20 × 10 | 200 | 1235 | 1311 | 200 | **1300** | 200 | **1320** | 200 |
| LA28 | 20 × 10 | 200 | 1216 | 1251 | 200 | **1271** | 200 | **1290** | 200 |
| LA29 | 20 × 10 | 200 | 1157 | 1250 | 200 | **1252** | 200 | **1262** | 200 |
| LA30 | 30 × 10 | 300 | 1355 | 1369 | 200 | **1400** | 200 | **1408** | 200 |
| **LA31** | **30 × 10** | **300** | **1784** | **1784** | **118** | **1784** | **163** | **1784** | **148** |
| **LA32** | **30 × 10** | **300** | **1850** | **1850** | **138** | **1850** | **147** | **1850** | **152** |
| **LA33** | **30 × 10** | **300** | **1719** | **1719** | **195** | **1719** | **156** | **1719** | **124** |

Como pode ser observado na Tabela 5.11, das 35 instâncias testadas, o HGA apresentou resultados melhores ou em número de gerações e/ou no valor do *makespan* para 17 instâncias em relação tanto ao GABL quanto ao *d*HGA, o que equivale à aproximadamente 50% das instâncias; em 14 instâncias houve piora nos resultados equivalente à aproximadamente 40% e para as outras 4 instâncias os valores permaneceram sem alteração.

Também calculou-se o *Gap* entre a melhor solução conhecida, BKS e a solução encontrada (SE) para cada abordagem (GABL, *d*HGA e HGA). A Tabela 5.12 apresenta a porcentagem aproximada do *Gap* para as abordagens e os valores realçados na cor azul destaca qual abordagem encontrou uma solução com valor do *makespan* mais próximo ótimo conhecido, BKS. O *Gap* foi calculado da seguinte forma: $Gap = |SE - BKS|/BKS * 100$ para cada uma das abordagens.

**Tabela 5.12:** *Gap da solução encontrada pelo o GABL, dHGA e HGA em relação ao BKS.*

| Instância | BKS | GABL | | *d*HGA | | HGA | |
|---|---|---|---|---|---|---|---|
| | | MKP | *Gap %* | MKP | *Gap %* | MKP | *Gap %* |
| FT10 | 930 | 1019 | 9,57 | 968 | 4,08 | **937** | **0,75** |
| LA03 | 597 | 604 | 1,17 | 604 | 1,17 | **603** | **1,00** |
| LA04 | 590 | 598 | 1,35 | 602 | 2,03 | **590** | **0,00** |
| LA16 | 945 | 982 | 3,91 | **945** | **0,00** | 946 | 0,10 |
| LA17 | 784 | 793 | 1,14 | **787** | **0,38** | 789 | 0,63 |
| LA18 | 848 | 861 | 1,53 | **848** | **0,00** | 855 | 0,82 |
| LA19 | 842 | 874 | 3,80 | **868** | **3,08** | 875 | 3,91 |
| LA20 | 902 | **907** | **0,55** | 913 | 1,22 | **907** | **0,55** |
| LA21 | 1046 | **1086** | **3,82** | 1099 | 5,06 | 1095 | 4,68 |
| LA22 | 927 | 982 | 5,93 | 967 | 4,31 | **946** | **2,04** |
| LA24 | 935 | 1002 | 7,16 | 996 | 6,52 | **990** | **5,82** |
| LA25 | 977 | **1029** | **5,32** | 1040 | 6,44 | 1043 | 6,75 |
| LA26 | 1218 | 1240 | 1,80 | 1256 | 3,11 | **1238** | **1,64** |
| LA27 | 1235 | 1311 | 6,15 | **1300** | **5,26** | 1320 | 6,88 |
| LA28 | 1216 | **1251** | **2,87** | 1271 | 4,52 | 1290 | 6,08 |
| LA29 | 1157 | **1250** | **8,03** | 1252 | 8,21 | 1262 | 9,07 |
| LA30 | 1355 | **1369** | **1,03** | 1400 | 3,31 | 1408 | 3,91 |

Como observa-se na Tabela 5.12 o HGA apresentou melhores resultados em relação ao GABL e *d*HGA, destacando-se para a instância LA04 que o HGA encontrou a solução ótima.

Comparou-se também a quantidade de número de gerações realizadas por cada abordagem, verificando-se o número de gerações do HGA em relação ao GABL e a quantidade de gerações do HGA em relação ao *d*HGA. Observou-se que a abordagem HGA foi melhor quando comparada com as outras duas abordagens. Os resultados correspondentes são apresentados na Tabela 5.13, na qual, os valores que estão na cor azul destacam os resultados que foram melhores.

**Tabela 5.13:** *Número de gerações do HGA em relação ao GABL e em relação ao dHGA.*

| Instância | GABL | *d*HGA | HGA | HGA-GABL % | HGA-dHGA % |
|---|---|---|---|---|---|
| FT06 | 8 | 2 | 2 | **75,00** | 0,00 |
| LA01 | 8 | 7 | 4 | **50,00** | **42,85** |
| LA02 | 20 | 25 | 19 | **5,00** | **24,00** |
| LA04 | 200 | 200 | 44 | **78,00** | **78,00** |
| LA06 | 7 | 1 | 2 | **71,42** | -100,00 |
| LA07 | 76 | 16 | 14 | **81,57** | **12,50** |
| LA08 | 16 | 10 | 3 | **81,25** | **70,00** |
| LA09 | 8 | 5 | 3 | **62,50** | **40,00** |
| LA10 | 3 | 1 | 1 | **66,66** | 0,00 |
| LA11 | 16 | 6 | 8 | **50,00** | -33,33 |
| LA12 | 12 | 6 | 5 | **58,33** | **16,66** |
| LA13 | 10 | 10 | 9 | **10,00** | **10,00** |
| LA15 | 34 | 36 | 31 | **8,82** | **13,88** |
| LA16 | 200 | 69 | 200 | 0,00 | -189,85 |
| LA18 | 200 | 98 | 200 | 0,00 | -104,08 |
| LA23 | 70 | 115 | 134 | -91,42 | -16,52 |
| LA31 | 118 | 163 | 148 | -25,42 | 9,20 |
| LA32 | 138 | 147 | 152 | -10,14 | -3,40 |
| LA33 | 195 | 156 | 124 | **36,41** | **20,51** |

O HGA foi melhor na questão de número de gerações quando comparado com GABL em 14 instâncias e, na comparação entre o HGA e o *d*HGA, o HGA foi melhor em 10 instâncias, com números expressivos na redução do número de gerações necessárias nas duas comparações (Tabela 5.13).

De modo geral, o HGA produziu melhores resultados quando comparado às abordagens GABL e *d*HGA. Essa melhoria está relacionada com a forma que os indivíduos semelhantes são substituídos por outros de melhor qualidade, como proposto pela abordagem de intensificação. Ao ampliar a busca a partir da vizinhança das soluções selecionadas pala PCA a intensificação aumenta a probabilidade de fugir de possíveis bacias de atração, contribuindo também para ampliar a diversidade.

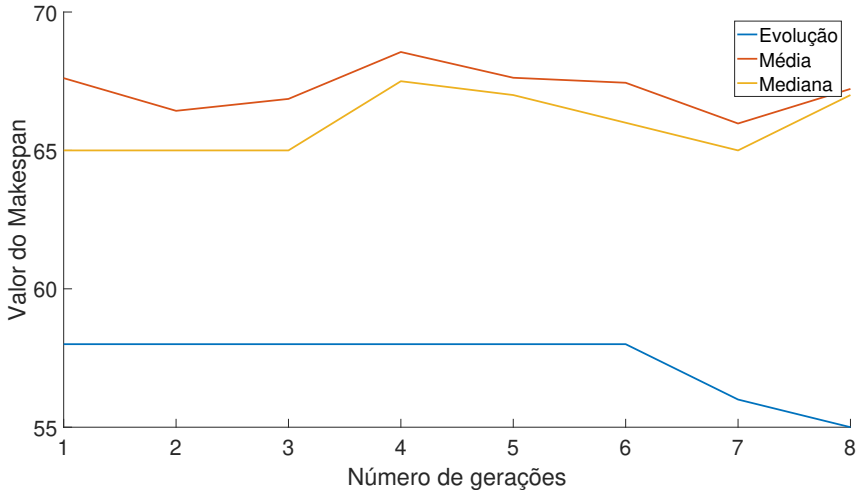
Vale frisar que os parâmetros dos algoritmos (GABL, HGA só com diversificação e HGA com diversificação e intensificação) não foram alterados em nenhuma fase dos experimentos. As Seções a seguir discutem os resultados obtidos para as instâncias FT06, FT10, LA03 e LA07.

### 5.3.1 Resultados dos experimentos do HGA para as instâncias FT06 e LA07

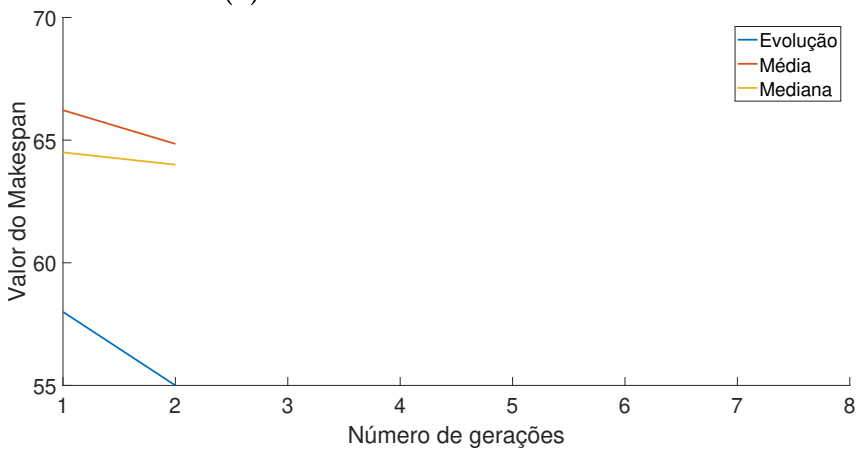
Nesta Seção são apresentados os resultados obtidos pelo HGA para as instâncias FT06 e LA07. Apresenta-se os resultados estatísticos da média e mediana bem como a evolução da população por gerações.

Os gráficos e histogramas apresentados anteriormente, referentes às abordagens GABL e HGA com diversificação ($d$HGA) são apresentados novamente para melhor visualização e comparação com os gráficos e histogramas das instâncias para a etapa da diversificação e intensificação, HGA.

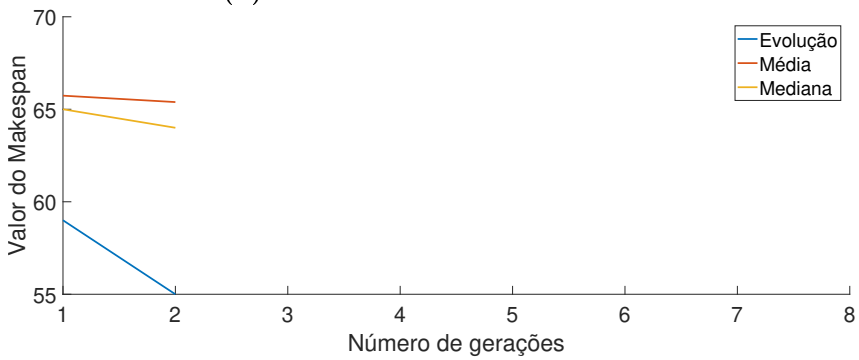
Como pode ser observado, na abordagem HGA, para a instância FT06, foram necessárias duas gerações para que o HGA encontrasse a solução ótima conhecida, ou seja, o *makespan* igual a 55 conforme apresentado na Figura 5.18c. O mesmo aconteceu com a abordagem $d$HGA (Figura 5.18b). Percebe-se na comparação entre o $d$HGA e o HGA que o comportamento das medidas de centralidade média e mediana são semelhantes.




**(a)** *Estatística do FT06 com GABL*




**(b)** *Estatística do FT06 com dHGA*

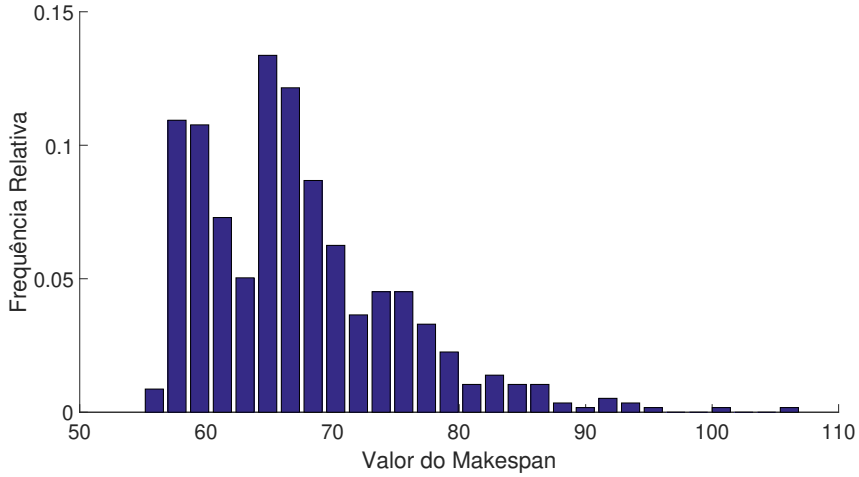



**(c)** *Estatística do FT06 com HGA*

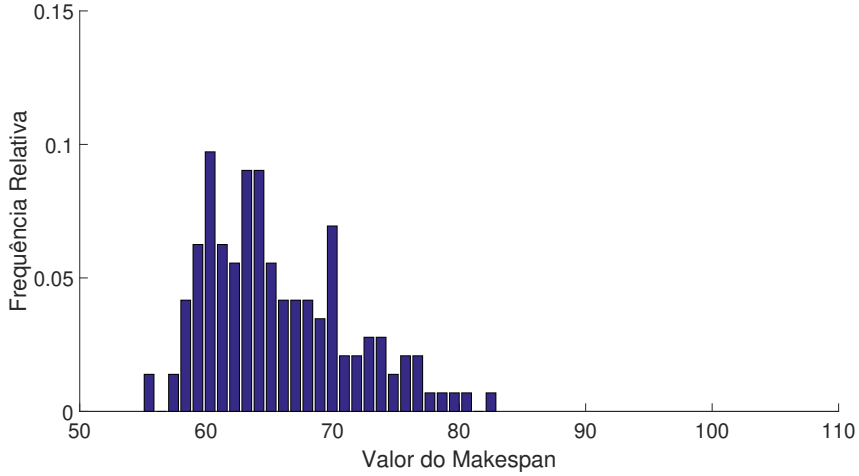
**Figura 5.18:** *Média, mediana e evolução da população do FT06 a cada geração com as abordagens GABL, dHGA e HGA.*

Fonte : A autora.

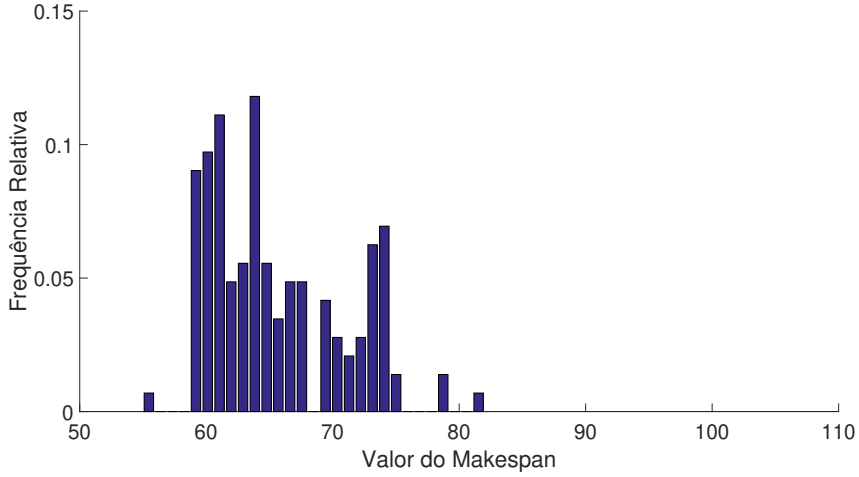
Apesar do HGA encontrar o *makespan* ótimo em duas gerações, pode-se observar um comportamento diferente da distribuição dos valores do *makespan* conforme apresentado no histograma da Figura 5.19c em relação às outras duas abordagens. Esses valores estão distribuídos de forma diferente e pode estar relacionado à forma como foi aplicado a diversificação e intensificação no HGA.




**(a)** *Histograma FT06 com GABL*




**(b)** *Histograma FT06 com dHGA*



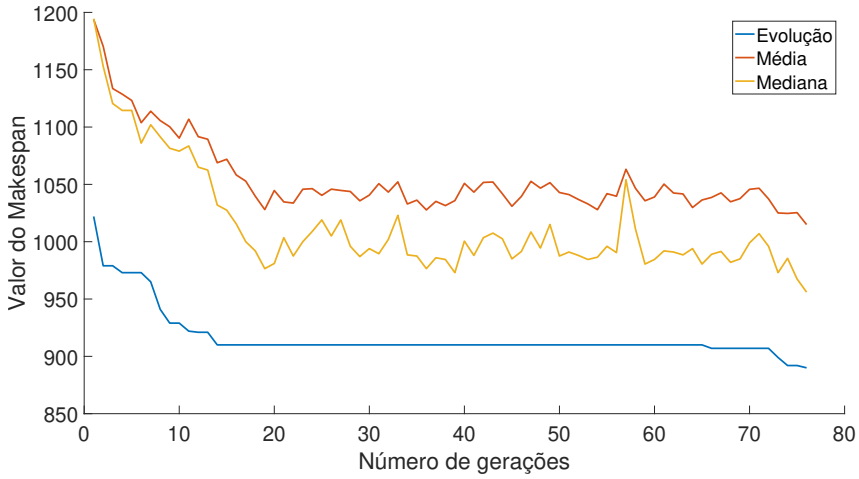

**(c)** *Histograma FT06 com HGA*

**Figura 5.19:** *Histograma dos indivíduos da população do FT06 com as abordagens GABL, dHGA e HGA.*

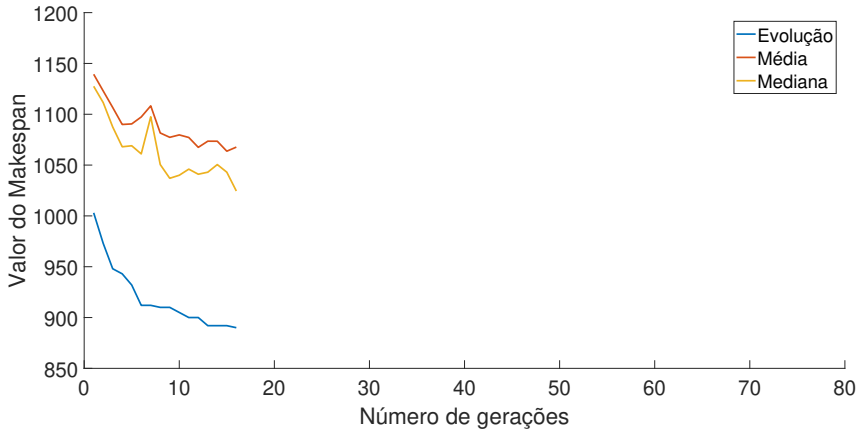
Fonte : A autora.

O HGA encontrou a solução ótima para o LA07 em 14 gerações, duas a menos do que o *d*HGA e sessenta e seis gerações a menos do que o GABL. Nota-se também nos

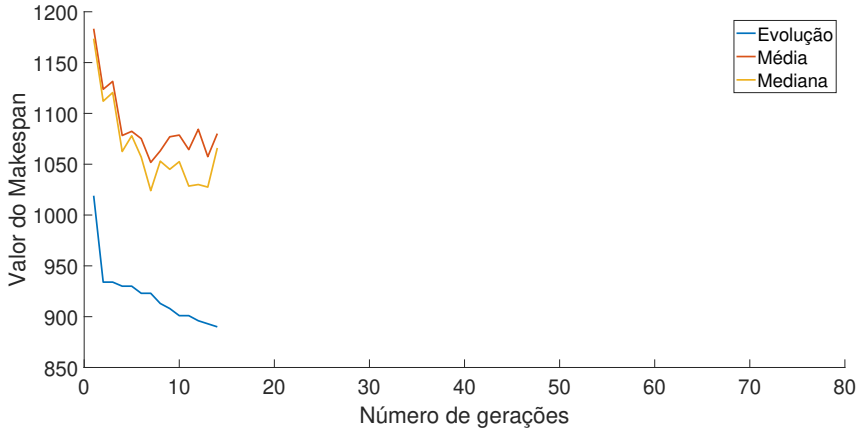
gráficos do *d*HGA e do HGA (Figuras 5.20b e 5.20c) o comportamento da mediana em relação à média, que se mantiveram próximas durante a evolução do algoritmo o que está relacionada à diversidade populacional conforme já observado anteriormente.




**(a)** *Estatística do LA07 com GABL*




**(b)** *Estatística do LA07 com dHGA*

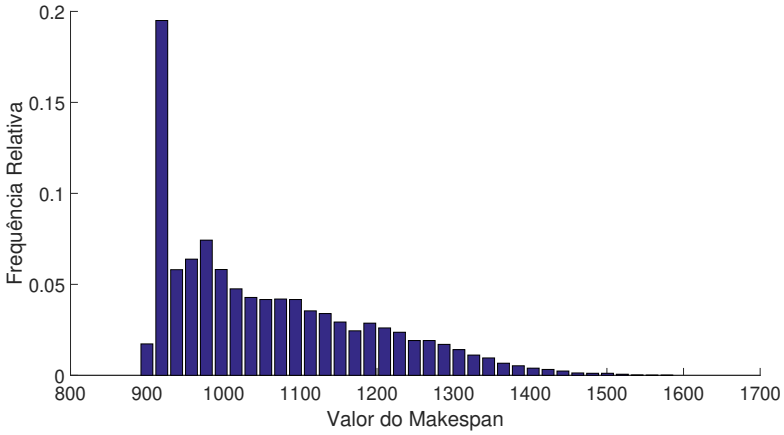



**(c)** *Estatística do LA07 com HGA*

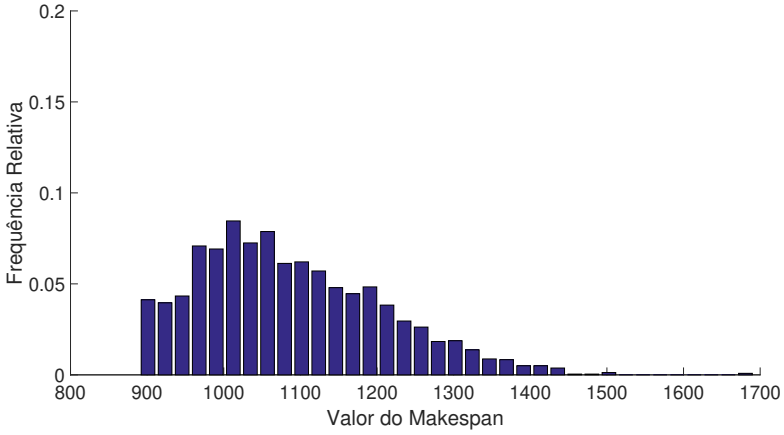
**Figura 5.20:** *Média, mediana e evolução da população do LA07 a cada geração com as abordagens GABL, dHGA e HGA.*

Fonte : A autora.

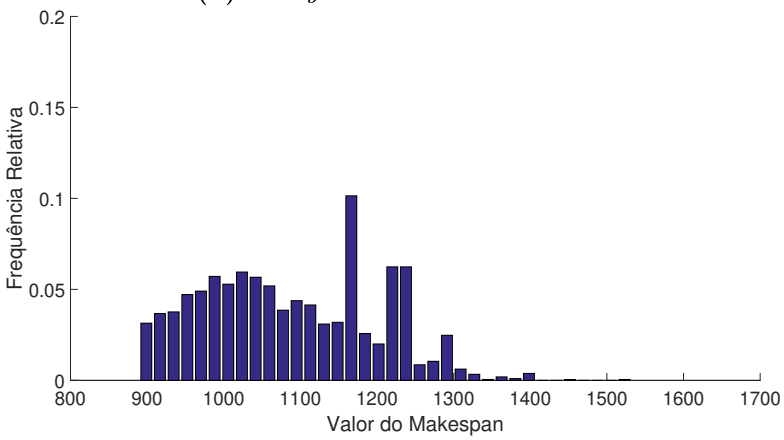
No histograma apresentado na Figura 5.21c, referente à instância LA07, observa-se que houve melhor distribuição do valor do *makespan* quando comparado ao *d*HGA e ao GABL (Figuras 5.21b e 5.21a, respectivamente). Nota-se que a frequência mais alta no histograma, cerca de 5% dos indivíduos da população com valor de *makespan* próximo de 1000, também é menor do que nas outras abordagens.




**(a)** *Histograma LA07 com GABL*




**(b)** *Histograma LA07 com dHGA*




**(c)** *Histograma LA07 com HGA*

**Figura 5.21:** *Histograma dos indivíduos da população do LA07 com as abordagens e GABL, dHGA e HGA.*

Fonte : A autora.

Como visto até então, na análise das instâncias FT06 e LA07 o HGA apresentou-se melhor tanto no que diz respeito à encontrar o valor ótimo do *makespan* para essas instâncias quanto em número de gerações e principalmente nas distribuições dos valores dos makespan quando compara-se com as abordagens GABL e *d*HGA.

A análise dos resultados das instâncias LA03 e FT10 utilizando-se o HGA são apresentados na Seção 5.3.2.

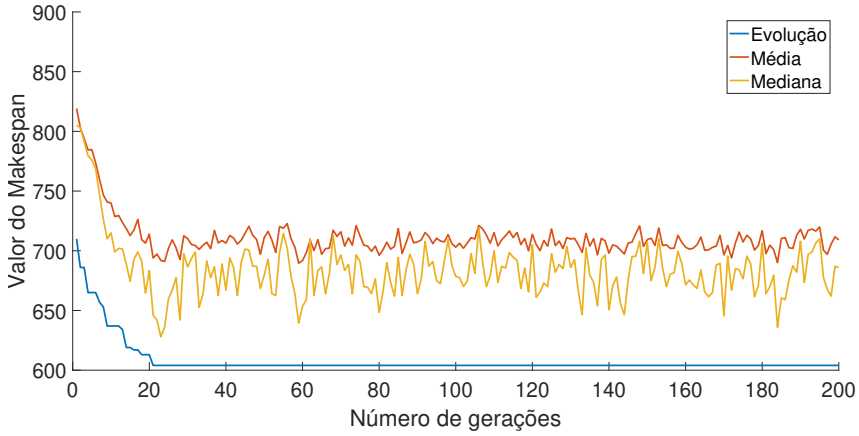
### 5.3.2 Resultados dos experimentos do HGA para as instâncias LA03 e FT10

Os resultados obtidos com a abordagem de diversificação e intensificação HGA para as instâncias LA03 e FT10, respectivamente, são apresentadas a seguir.

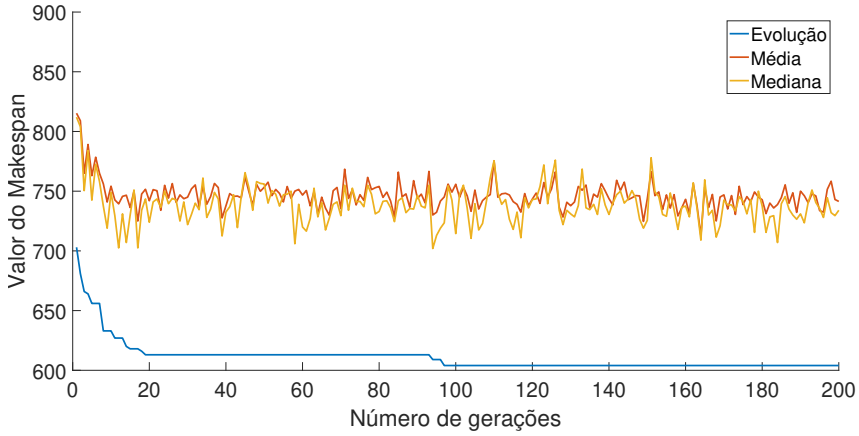
Para a instância LA03, apesar de haver uma melhora de uma unidade de tempo (*makespan* 603), em relação ao GABL e ao *d*HGA, o HGA não alcançou o valor do *makespan* ótimo.

A Figura 5.22c apresenta os valores da média, mediana e a evolução da população durante as 200 gerações. Nota-se que na 66ª geração, a melhor solução obtida pelo algoritmo ficou presa em um ótimo local. Esse comportamento indica que mesmo com a aplicação das abordagens de diversificação e intensificação a busca ficou concentrada em uma possível bacia de atração. Nesse caso, além de trabalhar em um melhor ajuste dos parâmetros do algoritmo genético binário adotado na intensificação, a seleção adequada das soluções iniciais para aplicação da intensificação pode ter efeitos significativos. Além de analisar o índice de contribuição gerado pela PCA, o valor de *makespan* também deve ser avaliado na seleção das soluções iniciais para o *path relinking*, segundo a característica do *big valey*.

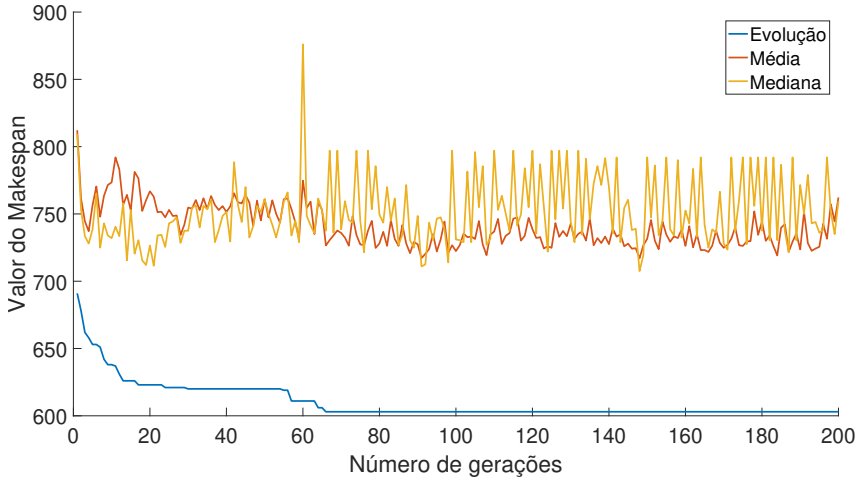
Em comparação com o *d*HGA, o HGA convergiu prematuramente para o ótimo local 603, mais cedo, como pode ser comparado com a Figura 5.22b.




**(a)** *Estatística do LA03 com GABL*




**(b)** *Estatística do LA03 com dHGA*



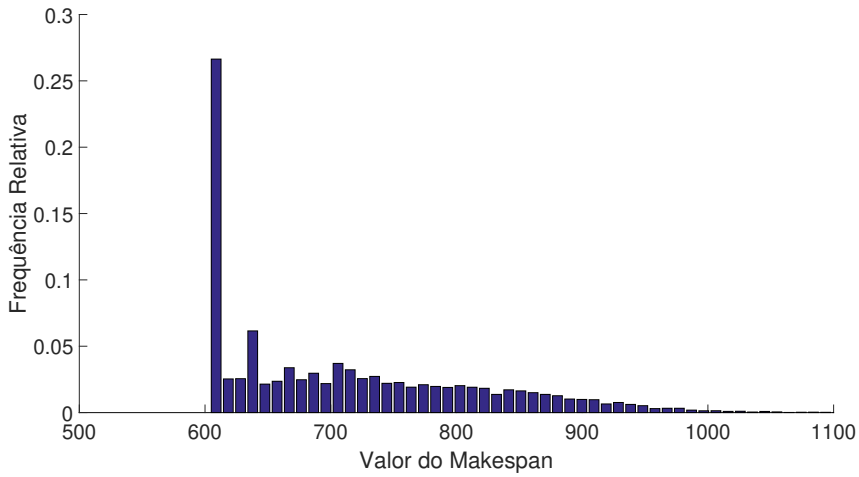

**(c)** *Estatística do LA03 com HGA*

**Figura 5.22:** *Média, mediana e evolução da população do LA03 a cada geração com as abordagens GABL, dHGA e HGA.*

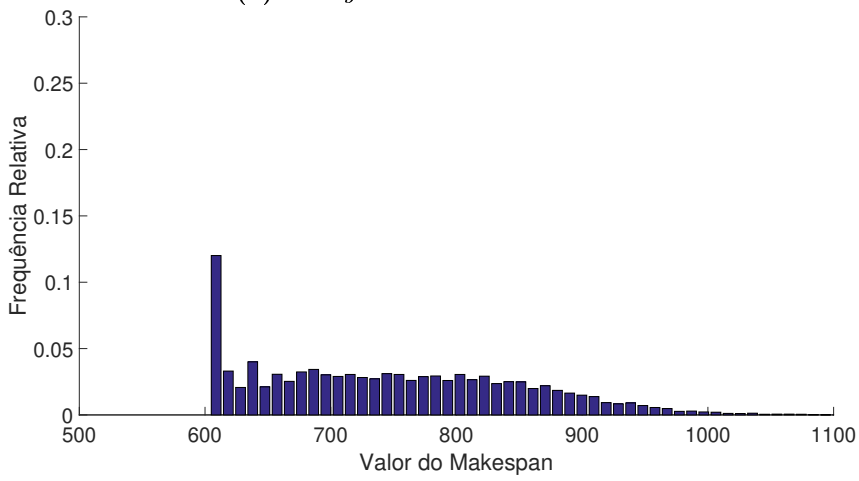
Fonte : A autora.

Analisando-se o histograma do LA03 apresentado na Figura 5.23c, nota-se um número expressivo de indivíduos, aproximadamente 20% dos indivíduos da população, com *makespan* na faixa de 790-800, e cerca de 8% dos indivíduos da população com *makespan* na faixa de 600-610. Esperava-se que os indivíduos estivessem melhor distribuídos, o que

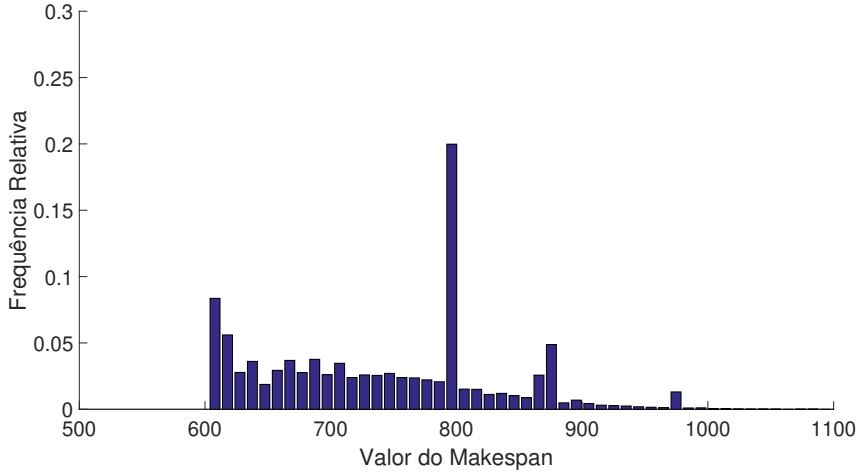
pode indicar que a substituição dos indivíduos selecionados pelo seu índice $CI$ não foram substituídos por indivíduos melhores.




**(a)** *Histograma LA03 com GABL*




**(b)** *Histograma LA03 com dHGA*

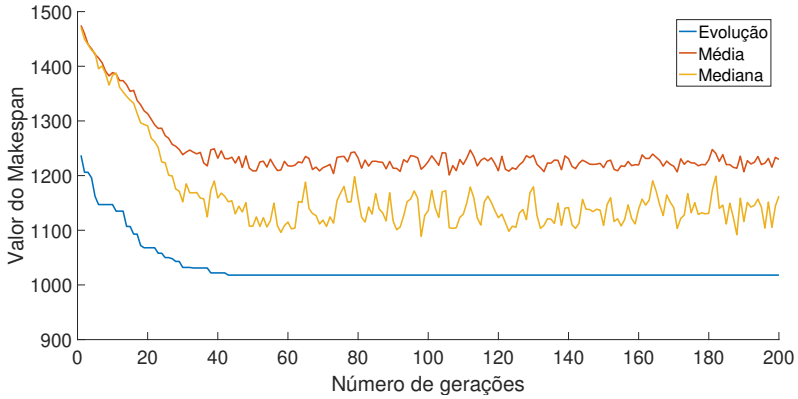



**(c)** *Histograma LA03 com HGA*

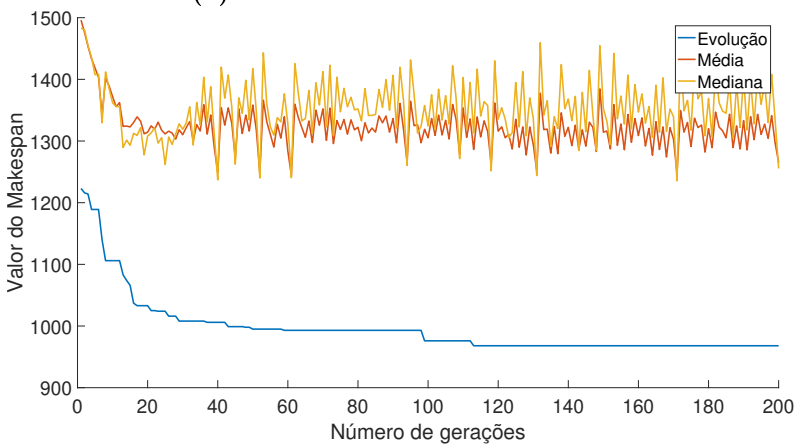
**Figura 5.23:** *Histograma dos indivíduos da população do LA03 com as abordagens e GABL, dHGA e HGA.*

Fonte : A autora.

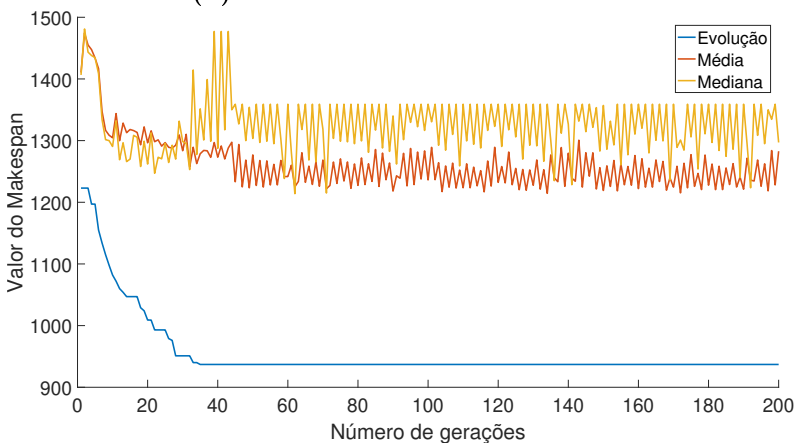
Para a instância FT10, apresenta-se na Figura 5.24c a média, mediana e a evolução da da população durante as gerações do algoritmo HGA. Nota-se que a mediana ficou acima da média quando o HGA ficou estagnado em um ótimo local, no *makespan* igual a 937, na 35ª geração, evidenciando que, mesmo aplicando-se a diversificação e intensificação, ainda não foi suficiente para que o desempenho do algoritmo fosse melhor.




**(a)** *Estatística do FT10 com GABL*




**(b)** *Estatística do FT10 com dHGA*

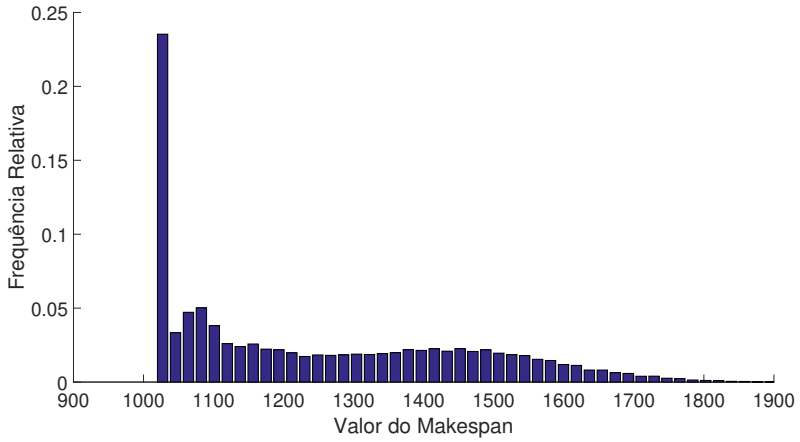



**(c)** *Estatística do FT10 com HGA*

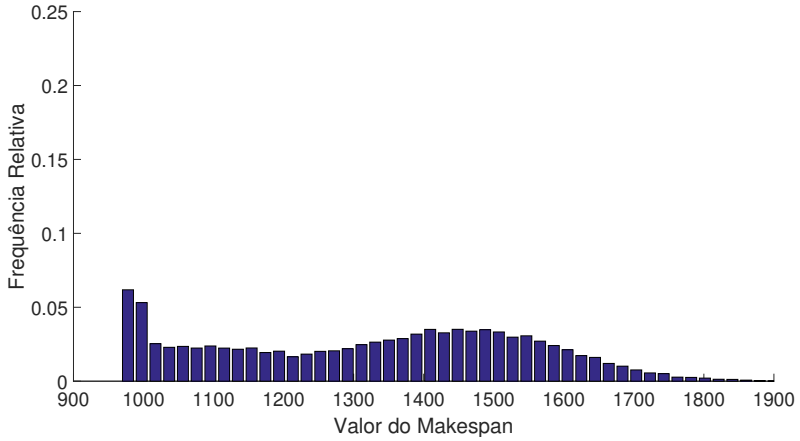
**Figura 5.24:** *Média, mediana e evolução da população do FT10 a cada geração com as abordagens GABL, dHGA e HGA.*

Fonte : A autora.

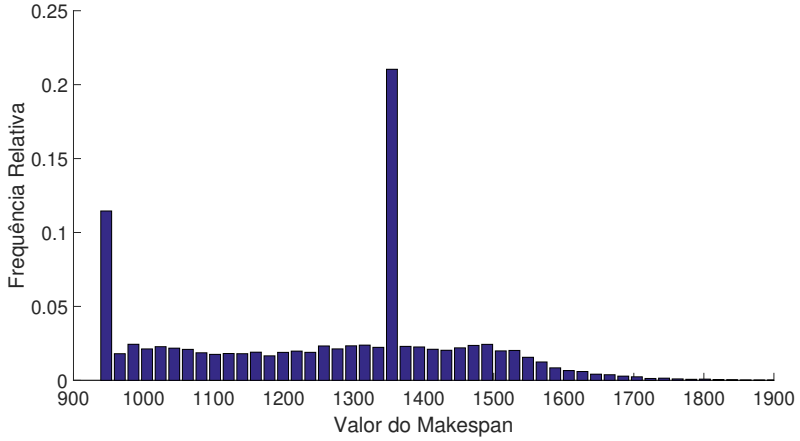
Na Figura 5.25c apresenta-se o histograma do FT10 para o HGA. Nota-se que existem aproximadamente 21% dos indivíduos da população com o *makespan* na faixa de 16350, o que não era esperado, principalmente quando comparado com o resultado do *d*HGA (Figura 5.25b), no qual houve melhor distribuição dos valores do *makespan*.




**(a)** *Histograma FT10 com GABL*




**(b)** *Histograma FT10 com dHGA*




**(c)** *Histograma FT10 com HGA*

**Figura 5.25:** *Histograma dos indivíduos da população do FT10 com as abordagens GABL, dHGA e HGA.*

Fonte : A autora.

Como visto até então, para as instâncias LA03 e FT10, o HGA não obteve um bom desempenho, apesar de melhorar os valores do *makespan* em relação ao GABL e *d*HGA, conforme apresentado na Tabela 5.11. Vários fatores podem ter contribuído para que o HGA não apresenta-se um desempenho como será discutido ao final deste Capítulo.

### 5.3.3 Análise dos resultados do HGA comparado com a literatura

Com a finalidade de avaliar/validar o algoritmo genético híbrido HGA proposto, os resultados obtidos são comparados com as abordagens listadas a seguir encontradas na literatura.

- GRASP para JSSP - (BINATO *et al.*, 2002) - GRASP-1
- GRASP Paralelo com path relinking para JSSP (AIEX; BINATO; RESENDE, 2003) - PGRA-SP-2
- Algoritmo Genético Híbrido para o JSSP (GONÇALVES; MENDES; RESENDE, 2005) - HGA-3
- Algoritmo Memético para o JSSP(HASAN *et al.*, 2009) - MA-4
- Algoritmo Genético com abordagem paralela baseada em um agente (ASADZADEH; ZAMANIFAR, 2010) - PaGA-5
- Algoritmo Memético para o JSSP(GAO *et al.*, 2011) - MA-6
- Algoritmo Genético Híbrido para o JSSP(JI; WANG, 2012) - HGA-7
- Algoritmo Híbrido baseado em biogeografia para o JSSP(WANG; DUAN, 2014) - HBBO-8
- Algoritmo Genético de busca local com agentes inteligentes para o JSSP (ASADZADEH, 2015) - aLSGA-9
- *Fast Simulated Annealing* Híbrido para o JSSP(AKRAM; KAMAL; ZEB, 2016) - HFSAQ-10
- Algoritmo Colônia de Abelhas Paralelo para o JSSP (ASADZADEH, 2016) - pABC-11

A Tabela 5.14 apresenta o nome das instâncias, o valor do *makespan* ótimo conhecido (BKS), os valores obtidos com a proposta HGA deste trabalho (terceira coluna) e os valores das abordagens listadas acima. Para melhor identificação das colunas referente aos trabalhos selecionados, cada coluna contém a sigla da proposta seguida do número conforme apresentado acima.

**Tabela 5.14:** *Comparação dos resultados do HGA com outras abordagens referenciadas na literatura.*

| Instância | BKS | **HGA** | GRASP-1 | PGRA-SP-2 | HGA-3 | MA-4 | PaGA-5 | MA-6 | HGA-7 | HBBO-8 | aLSGA-9 | HFSAQ-10 | pABC-11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FT06 | 55 | **55** | 55 | 55 | 55 | - | 55 | 55 | 55 | 55 | 55 | 55 | 55 |
| FT10 | 930 | 937 | 938 | 930 | 930 | - | 997 | 930 | 930 | 930 | 930 | 930 | 930 |
| LA01 | 666 | **666** | 666 | 666 | 666 | 666 | 666 | 666 | 666 | 666 | 666 | 666 | 666 |
| LA02 | 655 | **655** | 655 | 655 | 655 | 655 | 655 | 655 | 655 | 655 | 655 | 655 | 655 |
| LA03 | 597 | 603 | 604 | 597 | 597 | 597 | 617 | 597 | 597 | 597 | 606 | 597 | 597 |
| LA04 | 590 | **590** | 590 | 590 | 590 | 590 | 607 | 590 | 590 | 590 | 593 | 590 | 590 |
| LA05 | 593 | **593** | 593 | 593 | 593 | 593 | 593 | 593 | 593 | 593 | 593 | 593 | 593 |
| LA06 | 926 | **926** | 926 | 926 | 926 | 926 | 926 | 926 | 926 | 926 | 926 | 926 | 926 |
| LA07 | 890 | **890** | 890 | 890 | 890 | 890 | 890 | 890 | 890 | 890 | 890 | 890 | 890 |
| LA08 | 863 | **863** | 863 | 863 | 863 | 863 | 863 | 863 | 863 | 863 | 863 | 863 | 863 |
| LA09 | 951 | **951** | 951 | 951 | 951 | 951 | 951 | 951 | 951 | 951 | 951 | 951 | 951 |
| LA10 | 958 | **958** | 958 | 958 | 958 | 958 | 958 | 958 | 958 | 958 | 958 | 958 | 958 |
| LA11 | 1222 | **1222** | 1222 | 1222 | 1222 | 1222 | 1223 | 1222 | 1222 | 1222 | 1222 | 1222 | 1222 |
| LA12 | 1039 | **1039** | 1039 | 1039 | 1039 | 1039 | 1039 | 1039 | 1039 | 1039 | 1039 | 1039 | 1039 |
| LA13 | 1150 | **1150** | 1150 | 1150 | 1150 | 1150 | 1150 | 1150 | 1150 | 1150 | 1150 | 1150 | 1150 |
| LA14 | 1292 | **1292** | 1292 | 1292 | 1292 | 1292 | 1292 | 1292 | 1292 | 1292 | 1292 | 1292 | 1292 |
| LA15 | 1207 | **1207** | 1207 | 1207 | 1207 | 1207 | 1273 | 1207 | 1207 | 1207 | 1207 | 1207 | 1207 |
| LA16 | 945 | 946 | 946 | 945 | 945 | 945 | 994 | 945 | 945 | 945 | 946 | 945 | 945 |
| LA17 | 784 | 789 | 784 | 784 | 784 | 784 | 793 | 784 | 784 | 784 | 784 | 784 | 784 |
| LA18 | 848 | 855 | 848 | 848 | 848 | 848 | 860 | 848 | 848 | 848 | 848 | 848 | 848 |
| LA19 | 842 | 875 | 842 | 842 | 842 | 842 | 873 | 842 | 844 | 842 | 852 | 842 | 842 |
| LA20 | 902 | 907 | 907 | 902 | 907 | 907 | 912 | 902 | 911 | 902 | 907 | 902 | 907 |
| LA21 | 1046 | 1095 | 1091 | 1057 | 1046 | 1079 | 1146 | 1055 | 1046 | 1046 | 1068 | 1046 | 1046 |
| LA22 | 927 | 946 | 960 | 927 | 935 | 960 | 1007 | 927 | 935 | 933 | 956 | 927 | 927 |
| LA23 | 1032 | **1032** | 1032 | 1032 | 1032 | 1032 | 1033 | 1032 | 1032 | 1032 | 1032 | 1032 | 1032 |
| LA24 | 935 | 990 | 978 | 954 | 953 | 959 | 1012 | 940 | 953 | 935 | 966 | 935 | 935 |
| LA25 | 977 | 1043 | 1028 | 984 | 986 | 991 | 1067 | 984 | 984 | 977 | 1002 | 977 | 977 |
| LA26 | 1218 | 1238 | 1271 | 1218 | 1218 | 1218 | 1323 | 1218 | 1218 | 1218 | 1223 | 1218 | 1218 |
| LA27 | 1235 | 1320 | 1320 | 1269 | 1256 | 1286 | 1359 | 1261 | 1236 | 1235 | 1281 | 1235 | 1235 |
| LA28 | 1216 | 1290 | 1293 | 1225 | 1232 | 1286 | 1369 | 1216 | 1216 | 1216 | 1245 | 1216 | 1216 |
| LA29 | 1152 | 1262 | 1293 | 1203 | 1196 | 1221 | 1322 | 1190 | 1160 | 1153 | 1230 | 1164 | 1157 |
| LA30 | 1355 | 1408 | 1368 | 1355 | 1355 | 1355 | 1437 | 1355 | 1355 | 1355 | 1355 | 1355 | 1355 |
| LA31 | 1784 | **1784** | 1784 | 1784 | 1784 | 1784 | 1844 | 1784 | 1784 | 1784 | 1784 | - | 1784 |
| LA32 | 1850 | **1850** | 1850 | 1850 | 1850 | 1850 | 1907 | 1850 | 1850 | 1850 | 1850 | - | 1850 |
| LA33 | 1719 | **1719** | 1719 | 1719 | 1719 | 1719 | - | 1719 | 1719 | 1719 | 1719 | - | 1719 |

Os valores da terceira coluna da Tabela 5.14 que estão destacados em negrito indicam a instância na qual o HGA alcançou a solução ótima, ou seja, encontrou o *makespan* ótimo conhecido. Dentre as 35 instâncias testadas, o HGA encontrou uma solução ótima para 19 instâncias, aproximadamente 55% das instâncias avaliadas.

Para melhor avaliação do desempenho do HGA calculou-se o *Gap* dos resultados do HGA e das abordagens referenciadas na literatura (listadas acima) em relação à melhor solução conhecida, conforme apresentado na Tabela 5.15.

A Tabela 5.15 apresenta o nome das instâncias, o valor do *makespan* ótimo conhecido (BKS), os valores do *Gap* para o HGA (terceira coluna) para as demais abordagens que foram comparadas. A última linha da Tabela apresenta o *Gap* médio para cada abordagem.

De forma geral, quando se considera o *Gap* médio, o HGA não apresentou um bom desempenho, sendo superior apenas à abordagem PaGa-5 (Algoritmo Genético com abordagem paralela baseada em um agente). Nesse caso, o melhor desempenho foi obtido com pABC-11 (0,04%) seguido do HFSAQ-10. Contudo, quando compara-se os resultados do HGA em instâncias de forma individual, nota-se que o seu desempenho foi melhor ou equivalente às demais abordagens. Por exemplo, para as instâncias FT10, LA03, LA22, LA26, LA28, LA29 o HGA teve melhor desempenho do que GRASP-1 e, para a instância LA20 o HGA alcançou resultados iguais às abordagens GRASP-1, HGA-3, MA-4, aLSGA-9 e pABC-11.

Outro resultado que vale ser destacado é referente às instâncias FT10 e LA03, as quais são sabidamente de difícil solução. Para o FT10, o HGA teve desempenho superior em relação ao GRASP-1, PaGa-5 e, para o LA03, o HGA obteve melhores resultados do que GRASP-1, PaGa-5 e aLGA-9.

O desempenho observado para o HGA nesses experimentos são influenciados por diversos fatores. Primeiramente, a intensificação é baseada em um algoritmo genético binário bidimensional que, pela forma o qual foi idealizado, realiza uma busca local em uma vizinhança ampliada da solução inicial ao aplicar várias permutações de uma única vez nessa solução. Essa abordagem, teoricamente, aumenta as chances de exploração de outras regiões do espaço de busca, mas depende de um ajuste adequado dos parâmetros do método. Além disso, a escolha da solução inicial nesse processo também é fundamental, devendo avaliar não apenas o resultado da PCA mas também o valor de *makespan* para uma melhor exploração do conceito de "*big valey*".

**Tabela 5.15:** *Gap dos resultados do HGA e as abordagens referenciadas na literatura em relação à melhor solução conhecida, BKS.*

| Instância | BKS | **HGA** | GRASP-1 | PGRA-SP-2 | HGA-3 | MA-4 | PaGA-5 | MA-6 | HGA-7 | HBBO-8 | aLSGA-9 | HFSAQ-10 | pABC-11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FT10 | 930 | 0,75 | 0,86 | 0 | 0 | - | 7,20 | 0 | 0 | 0 | 0 | 0 | 0 |
| LA03 | 597 | 1,00 | 1,17 | 0 | 0 | 0 | 3,35 | 0 | 0 | 0 | 1,50 | 0 | 0 |
| LA04 | 590 | 0 | 0 | 0 | 0 | 0 | 2,88 | 0 | 0 | 0 | 0,50 | 0 | 0 |
| LA11 | 1222 | 0 | 0 | 0 | 0 | 0 | 0,08 | 0 | 0 | 0 | 0 | 0 | 0 |
| LA15 | 1207 | 0 | 0 | 0 | 0 | 0 | 5,46 | 0 | 0 | 0 | 0 | 0 | 0 |
| LA16 | 945 | 0,10 | 0,10 | 0 | 0 | 0 | 5,18 | 0 | 0 | 0 | 0,10 | 0 | 0 |
| LA17 | 784 | 0,63 | 0 | 0 | 0 | 0 | 1,14 | 0 | 0 | 0 | 0 | 0 | 0 |
| LA18 | 848 | 0,82 | 0 | 0 | 0 | 0 | 1,41 | 0 | 0 | 0 | 0 | 0 | 0 |
| LA19 | 842 | 3,91 | 0 | 0 | 0 | 0 | 3,68 | 0 | 0,23 | 0 | 1,18 | 0 | 0 |
| LA20 | 902 | 0,55 | 0,55 | 0 | 0,55 | 0,55 | 1,10 | 0 | 0,99 | 0 | 0,55 | 0 | 0,55 |
| LA21 | 1046 | 4,68 | 4,30 | 1,05 | 0 | 3,15 | 9,56 | 0,86 | 0 | 0 | 2,10 | 0 | 0 |
| LA22 | 927 | 2,04 | 3,55 | 0 | 0,86 | 3,55 | 8,62 | 0 | 0,86 | 0,64 | 3,12 | 0 | 0 |
| LA23 | 1032 | 0 | 0 | 0 | 0 | 0 | 0,09 | 0 | 0 | 0 | 0 | 0 | 0 |
| LA24 | 935 | 5,88 | 4,59 | 2,03 | 1,92 | 2,56 | 8,23 | 0,53 | 1,92 | 0 | 3,31 | 0 | 0 |
| LA25 | 977 | 6,75 | 5,22 | 0,71 | 0,92 | 1,43 | 9,21 | 0,71 | 0,71 | 0 | 2,55 | 0 | 0 |
| LA26 | 1218 | 1,64 | 4,35 | 0 | 0 | 0 | 8,62 | 0 | 0 | 0 | 0,41 | 0 | 0 |
| LA27 | 1235 | 6,88 | 6,88 | 2,75 | 1,70 | 4,12 | 10,04 | 2,10 | 0,08 | 0 | 3,72 | 0 | 0 |
| LA28 | 1216 | 6,08 | 6,33 | 0,74 | 1,31 | 5,75 | 12,58 | 0 | 0 | 0 | 2,38 | 0 | 0 |
| LA29 | 1152 | 9,54 | 12,23 | 4,42 | 3,81 | 5,98 | 14,75 | 3,29 | 0,69 | 0,086 | 6,77 | 1,04 | 0,43 |
| LA30 | 1355 | 3,91 | 0,95 | 0 | 0 | 0 | 6,05 | 0 | 0 | 0 | 0 | 0 | 0 |
| LA31 | 1784 | 0 | 0 | 0 | 0 | 0 | 3,36 | 0 | 0 | 0 | 0 | - | 0 |
| LA32 | 1850 | 0 | 0 | 0 | 0 | 0 | 3,08 | 0 | 0 | 0 | 0 | - | 0 |
| ***Gap* médio** | | 2.51 | 2.32 | 0.53 | 0.50 | 1.29 | 5.71 | 0.34 | 0.25 | 0.03 | 1.28 | 0.05 | 0.04 |

Os parâmetros do GAB utilizados nos experimentos foram configurados com base no trabalho de Grassi, Schimit e Pereira (2016) e são apresentados na Tabela 4.7.

No que tange às configurações dos parâmetros do GAB (Tabela 4.7, Seção 4.6 do Capítulo 4), os valores foram configurados com base no trabalho de Grassi, Schimit e Pereira (2016), conforme mencionado anteriormente, e não foram realizados testes exaustivos para definir valores adequados para cada instância o que é um aspecto muito importante. Para a instância LA03, por exemplo, apenas ajustando de 2 para 5 o valor do parâmetro "*nConvergence*", que estabelece o fim da execução do algoritmo após um determinado número de gerações sem melhora da solução corrente, o valor de *makespan* foi de 604 (Tabela 5.11) para o valor ótimo conhecido de 597. Ainda, para a instância LA04, que já havia encontrado uma solução ótima, o número de gerações foi reduzido de 44 para 38, o que significa um ganho de aproximadamente 9%.

Por outro lado, a intensificação está sendo aplica em todos os indivíduos que foram classificados pela PCA com o coeficiente maior ou igual a 0,98 ($CI \geq 0,98$). Esses indivíduos estão muito próximos e a intensificação está sendo realizada sempre à partir de pontos próximos no espaço de busca, o que requer uma melhor análise.

Por fim, vale lembrar que tanto os parâmetros do HGA (Tabela 4.6) quanto do GAB (Tabela 4.7), conforme apresentado na Seção 4.6 do Capítulo 4, foram utilizados para todas as instâncias sem alteração.

Capítulo 6

# Conclusões, limitações, sugestões e contribuições

Neste trabalho foi proposto um Algoritmo Genético Híbrido, denotado HGA, para a resolução do problema *Job Shop Scheduling* baseado na análise de componentes principais do *fitness landscape*. Os resultados da análise de componentes principais são utilizados para dar suporte as estratégias de diversificação e intensificação desenvolvidas.

Na estratégia de diversificação, a análise de componentes principais mostrou-se eficiente em explorar as características de vizinhança do *fitness landscape*, identificando adequadamente os indivíduos mais semelhantes e que são, portanto, candidatos a serem substituídos nesta etapa. Os resultados da PCA refletem a diversidade populacional na medida em que representam uma relação de vizinhança entre um indivíduo e os demais da população.

Vale destacar que a PCA seleciona as principais características do *fitness landscape*, realizando uma análise em tempo de execução do algoritmo, para cada instância específica, na qual o resultado é utilizado diretamente no processo de solução. Assim, a análise do *fitness landscape* é realizada de forma *online* para cada instância do problema, para o método utilizado e os operadores escolhidos.

A estratégia de intensificação atua nos indivíduos que são identificados na diversificação, ampliando a exploração do espaço de soluções em busca uma solução substituta de melhor qualidade. Nesta etapa utiliza-se um algoritmo genético binário bidimensional (GAB) e o método *path relinking*, ambos trabalhando em conjunto.

A utilização do GAB permite identificar um conjunto de movimentações que aplicadas a solução corrente produzem uma solução substituta de melhor qualidade. Esse conjunto de movimentações determina o caminho que o *path relinking* deverá reconstruir, explorando assim o conceito de "*big valley*".

Assim, a aplicação da intensificação da busca a partir dos indivíduos selecionados pela PCA mostrou-se capaz de encontrar melhores soluções substitutas em comparação à substituição aleatória.

Em geral, o HGA foi capaz de resolver de maneira ótima 19 das 35 instâncias testadas, conforme apresentado na Tabela 5.14. Porém, apesar de não encontrar o ótimo em 16 instâncias e de apresentar um *Gap* médio superior quando comparado às abordagens referenciadas na Seção 5.3.3 do Capítulo 5, a abordagem proposta apresentou ganhos significativos em instâncias individuais, tanto em relação ao valor de *makespan* quanto no que se refere ao número de gerações. Os ganhos foram ainda mais significativos quando comparado com o GABL conforme apresentado nas Tabelas 5.6, 5.7 e 5.8. Portanto, acredita-se no potencial das estratégias de diversificação e intensificação propostas.

Ressalta-se, no entanto, que o desempenho do HGA é dependente de um adequado ajuste dos seus parâmetros, e que a aplicação da intensificação em soluções com índices

de contribuição semelhantes pode resultar em perda de diversidade e limitar a exploração de outras regiões do espaço de busca, além de elevar o custo computacional.

Outra limitação encontrada diz respeito à utilização do BRKGA que, apesar de ser indicado para problemas de otimização combinatória, como é o caso do JSSP, demanda uma série de conversões de representação para gerar uma solução para o problema, visto que o BRKGA utiliza vetores de números reais.

Para continuidade do trabalho, sugere-se a investigação de cenários e estratégias para selecionar as soluções iniciais para a abordagem de intensificação, analisando-se não somente o índice de contribuição $CI$ do indivíduo, mas também a correlação desse índice com o seu valor de *makespan* o que pode favorecer a exploração do conceito de "*big valey*".

## 6.1 Contribuições da Pesquisa

Pode-se considerar como as principais contribuições desta pesquisa as abordagens de diversificação e intensificação propostas a partir da análise *online* do *fitness landscape*.

A utilização da PCA e a forma como ela é aplicada na etapa de diversificação também representa uma contribuição. A análise de componentes principais considera uma matriz de coeficientes de regressão que carrega informações de cada um dos indivíduos da população em relação a todos os demais, refletindo a diversidade populacional. O resultado dessa análise é um índice de contribuição individual que apresenta uma forte correlação negativa com a distância média entre o indivíduo e os demais, possibilitando uma classificação em relação à diversidade.

Portanto, a partir do índice de contribuição individual os indivíduos semelhantes subótimos podem ser submetidos à intensificação e explorar outras características do *fitness landscape* como, por exemplo, o conceito de "*big valley*" no qual soluções (sub)ótimas estão conectadas.

A abordagem de intensificação *i*HGA aprimora as movimentações realizadas no algoritmo genético binário de Grassi, Schimit e Pereira (2016). No GAB proposto neste trabalho, os movimentos são realizados antecipando-se e ajustando-se as operações nas máquinas de forma a não formar ciclos no grafo conjuntivo e, portanto, sempre produzindo soluções factíveis. Ademais, o GAB expande a busca na vizinhança da solução corrente contribuindo para a diversificação.

Por fim, o cromossomo binário ótimo gerado pelo GAB é utilizado para indicar ao método *path relinking* o caminho a ser reconstruído a partir da solução inicial até a solução final, o que também difere esta proposta das abordagens tradicionais do *path relinking*.

# Referências Bibliográficas

ADAMS, J.; BALAS, E.; ZAWACK, D. The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science*, v. 34, n. 3, p. 391–401, 1988. Citado na pág. 18.

AIEX, R.; BINATO, S.; RESENDE, M. G. C. Parallel GRASP with path-relinking for job shop scheduling. *Parallel Computing*, v. 29, n. 4, p. 393–430, 2003. ISSN 01678191. Citado na pág. 47, 74, 119.

AKKAN, C.; KARABATI, S. The two-machine flowshop total completion time problem: Improved lower bounds and a branch-and-bound algorithm. *European Journal of Operational Research*, v. 159, n. 2 SPEC. ISSUE, p. 420–429, 2004. Citado na pág. 18.

AKRAM, K.; KAMAL, K.; ZEB, A. Fast simulated annealing hybridized with quenching for solving job shop scheduling problem. *Applied Soft Computing Journal*, Elsevier B.V., v. 49, p. 510–523, 2016. ISSN 15684946. Disponível em: <https://doi.org/10.1016/j.asoc.2016.08.037>. Citado na pág. 51, 52, 119.

AMIRGHASEMI, M.; ZAMANI, R. A synergetic combination of small and large neighborhood schemes in developing an effective procedure for solving the job shop scheduling problem. *SpringerPlus*, v. 3, n. 1, p. 193, 2014. ISSN 2193-1801. Citado na pág. 34.

AMIRGHASEMI, M.; ZAMANI, R. An effective asexual genetic algorithm for solving the job shop scheduling problem. *Computers & Industrial Engineering*, Elsevier Ltd, v. 83, p. 123–138, 2015. ISSN 03608352. Citado na pág. 75.

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. *Pesquisa Operacional*. [S.l.]: Elsevier Editora Ltda., 2011. 542 p. Citado na pág. 17.

ASADZADEH, L. A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Computers & Industrial Engineering*, Elsevier Ltd, v. 85, p. 376–383, 2015. ISSN 03608352. Disponível em: <https://doi.org/10.1016/j.cie.2015.04.006>. Citado na pág. 23, 35, 52, 53, 60, 119.

ASADZADEH, L. A parallel artificial bee colony algorithm for the job shop scheduling problem with a dynamic migration strategy. *Computers & Industrial Engineering*, Elsevier Ltd, v. 102, p. 359–367, 2016. ISSN 03608352. Disponível em: <https://doi.org/10.1016/j.cie.2016.06.025>. Citado na pág. 51, 119.

ASADZADEH, L.; ZAMANIFAR, K. An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. *Mathematical and Computer Modelling*, 2010. Citado na pág. 119.

BAIR, E.; HASTIE, T.; PAUL, D.; TIBSHIRANI, R. Prediction by supervised principal components. *Journal of the American Statistical Association*, v. 101, n. 473, p. 119–137, 2006. ISSN 01621459. Citado na pág. 47, 56, 66, 69.

BALAS, E. Machine Sequencing Via Disjunctive Graphs: An Implicit Enumeration Algorithm. *Operations Research*, v. 17, n. 6, p. 941–957, dec 1969. ISSN 0030-364X. Citado na pág. 11, 26, 34.

BARNETT, L. *Evolutionary search on fitness landscapes with neutral networks*. Tese (Doutorado) — University of Sussex, 2003. Citado na pág. 56.

BASSEUR, M.; GOËFFON, A. Climbing combinatorial fitness landscapes. *Applied Soft Computing*, Elsevier, v. 30, p. 688–704, may 2015. Citado na pág. 18, 31, 54.

BAYKASOĞLU, A.; HAMZADAYI, A.; KÖSE, S. Y. Testing the performance of teaching-learning based optimization (TLBO) algorithm on combinatorial problems: Flow shop and job shop scheduling cases. *Information Sciences*, v. 276, p. 204–218, 2014. Disponível em: <https://doi.org/10.1016/j.ins.2014.02.056>. Citado na pág. 18, 30, 33.

BEAN, J. C. Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing*, v. 6, n. 2, p. 154–160, may 1994. ISSN 0899-1499. Citado na pág. 41.

BEASLEY, J. *OR-Library*. 1990. Citado na pág. 60, 61, 62, 63, 64.

BERTRAND, J. W. M.; FRANSOO, J. C. Operations management research methodologies using quantitative modeling. *International Journal of Operations & Production Management*, v. 22, n. 2, p. 241–264, feb 2002. ISSN 0144-3577. Citado na pág. 58.

BIERWIRTH, C.; KUHPFAHL, J. Extended GRASP for the job shop scheduling problem with total weighted tardiness objective. *European Journal of Operational Research*, Elsevier B.V., v. 261, n. 3, p. 835–848, 2017. ISSN 03772217. Citado na pág. 47.

BIERWIRTH, C.; MATTFELD, D. C.; WATSON, J.-P. Landscape Regularity and Random Walks for the Job-Shop Scheduling Problem. In: GOTTLIEB, J.; RAIDL, G. R. (Ed.). *Evolutionary Computation in Combinatorial Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 21–30. ISBN 978-3-540-24652-7. ISSN 03029743. Disponível em: <https://doi.org/10.1007/978-3-540-24652-7_3>. Citado na pág. 19, 30, 31, 55.

BINATO, S.; HERY, W. J.; LOEWENSTERN, D. M.; RESENDE, M. G. C. A Grasp for Job Shop Scheduling. In: *Essays and Surveys in Metaheuristics*. Boston, MA: Springer US, 2002. p. 59–79. ISBN 978-1-4615-1507-4. Citado na pág. 119.

BLUM, C. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, v. 2, n. 4, p. 353–373, 2005. ISSN 15710645. Citado na pág. 22.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys*, v. 35, n. 3, p. 189–213, 2003. ISSN 02545330. Citado na pág. 22, 45.

BOŻEJKO, W.; GNATOWSKI, A.; SMUTNICKI, C.; UCHROŃSKI, M.; WODECKI, M. Local Search Metaheuristics with Reduced Searching Diameter. In: R., M.-D.; F., P.; QUESADA-ARENCIBIA, A. (Ed.). *Computer Aided Systems Theory – EUROCAST 2017*. [S.l.]: Springer International Publishing, 2018. v. 1, p. 447–454. ISBN 978-3-319-74718-7. Citado na pág. 19, 20, 30, 56.

BOŻEJKO, W.; SMUTNICKI, C.; UCHROŃSKI, M.; WODECKI, M. Big valley in scheduling problems landscape - Metaheuristics with reduced searching area. *2017 22nd International Conference on Methods and Models in Automation and Robotics, MMAR 2017*, p. 458–462, 2017. Citado na pág. 20, 56.

BRIZUELA, C. A.; SANNOMIYA, N. A Diversity Study in Genetic Algorithms for Job Shop Scheduling Problems. *Proc. of the {G}enetic and {E}volutionary {C}omputation {C}onf. {GECCO}-99*, p. 75–82, 1999. Citado na pág. 37, 45.

BÜRGY, R. A neighborhood for complex job shop scheduling problems with regular objectives. *Journal of Scheduling*, Springer US, v. 20, n. 4, p. 391–422, 2017. ISSN 10946136. Citado na pág. 19.

CABO, M.; GONZÁLEZ-VELARDE, J. L.; POSSANI, E.; Ríos Solís, Y. Bi-objective scheduling on a restricted batching machine. *Computers and Operations Research*, v. 100, p. 201–210, 2018. ISSN 03050548. Citado na pág. 64.

ÇALIŞ, B.; BULKAN, S. A research survey: review of AI solution strategies of job shop scheduling problem. *Journal of Intelligent Manufacturing*, v. 26, n. 5, p. 961–973, 2015. ISSN 15728145. Citado na pág. 17, 36.

CARLIER, J.; PINSON, E. An Algorithm for Solving the Job-Shop Problem. *Management Science*, v. 35, n. 2, p. 164–176, feb 1989. ISSN 0025-1909. Citado na pág. 18.

CHAVES, A. A.; LORENA, L. A.; SENNE, E. L.; RESENDE, M. G. C. Hybrid method with CS and BRKGA applied to the minimization of tool switches problem. *Computers and Operations Research*, Elsevier, v. 67, p. 174–183, 2016. ISSN 03050548. Citado na pág. 42, 64.

CHWIF, L.; MEDINA, A. C. *Modelagem e Simulação de Eventos Discretos: Teoria e Aplicações*. 2ª edição. ed. São Paulo - SP: Editora dos Autores, 2007. Citado na pág. 58.

CZOGALLA, J.; FINK, A. Fitness landscape analysis for the no-wait flow-shop scheduling problem. *Journal of Heuristics*, Springer Nature, v. 18, n. 1, p. 25–51, jan 2011. Citado na pág. 55.

DAMM, R. B.; RESENDE, M. G. C.; RONCONI, D. P. A biased random key genetic algorithm for the field technician scheduling problem. *Computers and Operations Research*, Elsevier, v. 75, p. 49–63, 2016. ISSN 03050548. Citado na pág. 52.

de Oliveira Jr, J. I.; Da Rocha, J. C. F.; GUIMARÃES, A. M.; da Fonseca, A. F. A PCA and SPCA based procedure to variable selection in agriculture. *Revista Brasileira de Computação Aplicada*, v. 7, n. 1, p. 30–41, 2015. Citado na pág. 47.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, v. 26, n. 1, p. 29–41, 1996. Citado na pág. 19.

EL-DESOKY, I. M.; EL-SHORBAGY, M. A.; NASR, S. M.; HENDAWY, Z. M.; MOUSA, A. A. A Hybrid Genetic Algorithm for Job Shop Scheduling Problems. *IJAETCS Research Article International Journal of Advancement in Engineering Technology and Computer Sciences*, v. 3, n. 1, p. 6–17, 2016. Citado na pág. 52, 53.

FISHER, H.; THOMPSON, G. L. Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules. In: MUTH, J.; THOMPSON, G. (Ed.). *Industrial Scheduling*. [S.l.]: Prentice Hall, 1963. p. 225–251. Citado na pág. 14, 18, 30, 60.

FUCHIGAMI, H. Y.; MOURA, M. A. S.; BRANCO, F. J. C. Modelos matemáticos para programação de job shop com tempos de setup independentes da sequência. *Revista Produção Online*, v. 17, n. 1, p. 245–267, mar 2017. ISSN 16761901. Citado na pág. 18.

GAO, L.; LI, X.; WEN, X.; LU, C.; WEN, F. A hybrid algorithm based on a new neighborhood structure evaluation method for job shop scheduling problem. *Computers & Industrial Engineering*, Elsevier Ltd, v. 88, p. 417–429, 2015. ISSN 03608352. Citado na pág. 18, 25, 51, 60.

GAO, L.; ZHANG, G.; ZHANG, L.; LI, X. An efficient memetic algorithm for solving the job shop scheduling problem. *Computers & Industrial Engineering*, Elsevier Ltd, v. 60, n. 4, p. 699–705, 2011. ISSN 03608352. Citado na pág. 17, 18, 19, 33, 45, 51, 119.

GHAEDI, M.; GHAEDI, A. M.; ABDI, F.; ROOSTA, M.; SAHRAEI, R.; DANESHFAR, A. Principal component analysis-artificial neural network and genetic algorithm optimization for removal of reactive orange 12 by copper sulfide nanoparticles-activated carbon. *Journal of Industrial and Engineering Chemistry*, The Korean Society of Industrial and Engineering Chemistry, v. 20, n. 3, p. 787–795, 2014. ISSN 22345957. Citado na pág. 46.

GLOVER, F. Tabu Search and Adaptive Memory Programming - Advances, Applications and Challenges. In: BARR, R. S.; HELGASON, R. V.; KENNINGTON, J. L. (Ed.). *Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies*. Boston, MA: [s.n.], 1997. cap. 1, p. 1–75. Citado na pág. 47.

GLOVER, F.; LAGUNA, M. *Tabu Search*. Boston, MA: Springer US, 1997. 382 p. ISBN 978-0-7923-8187-7. Citado na pág. 18, 33, 45, 64.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675. Citado na pág. 36.

GONÇALVES, J. F.; MENDES, J. J. d. M.; RESENDE, M. G. C. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, v. 167, n. 1, p. 77–95, nov 2005. ISSN 03772217. Citado na pág. 35, 52, 53, 64, 76, 77, 119.

GONÇALVES, J. F.; RESENDE, M. G. C. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, v. 17, n. 5, p. 487–525, 2011. ISSN 13811231. Citado na pág. 42, 43, 64, 76, 77.

GONÇALVES, J. F.; RESENDE, M. G. C. A parallel multi-population biased random-key genetic algorithm for a container loading problem. *Computers and Operations Research*, v. 39, n. 2, p. 179–190, 2012. ISSN 03050548. Citado na pág. 64.

GONÇALVES, J. F.; RESENDE, M. G. C. A biased random key genetic algorithm for 2D and 3D bin packing problems. *International Journal of Production Economics*, Elsevier, v. 145, n. 2, p. 500–510, 2013. ISSN 09255273. Citado na pág. 64.

GONÇALVES, J. F.; RESENDE, M. G. C. An extended Akers graphical method with a biased random-key genetic algorithm for job-shop scheduling. *International Transactions in Operational Research*, v. 21, n. 2, p. 215–246, 2014. ISSN 14753995. Citado na pág. 35, 42, 52, 64.

GRABOWSKI, J.; WODECKI, M. A Very Fast Tabu Search Algorithm for Job Shop Problem. *Metaheuristic Optimization via Memory and Evolution - Tabu Search and Scatter Search*, v. 30, n. August, p. 117–144, 2005. ISSN 01464833. Citado na pág. 18, 71, 82.

GRASSI, F.; SCHIMIT, P. H. T.; PEREIRA, F. H. Dynamic Seed Genetic Algorithm to Solve Job Shop Scheduling Problems. In: *IFIP Advances in Information and Communication Technology*. [S.l.: s.n.], 2016. v. 488, p. 170–177. ISBN 9783319511320. Citado na pág. 70, 71, 77, 123, 125.

HASAN, S. M.; SARKER, R.; ESSAM, D.; CORNFORTH, D. Memetic algorithms for solving job-shop scheduling problems. *Memetic Computing*, v. 1, n. 1, p. 69–83, 2009. ISSN 18659284. Citado na pág. 119.

HAUPT, R. L.; HAUPT, S. E. The Binary Genetic Algorithm. In: *Practical Genetic Algorithms*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2004. p. 27–50. ISBN 0471455652. Citado na pág. 36.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. Massachusetts: MIT Press, 1975. Citado na pág. 19, 36.

Hussein Al-Arashi, W.; IBRAHIM, H.; Azmin Suandi, S. Optimizing principal component analysis performance for face recognition using genetic algorithm. *Neurocomputing*, Elsevier, v. 128, p. 415–420, 2014. ISSN 09252312. Citado na pág. 46.

JI, R. Qing-dao-er; WANG, Y. A new hybrid genetic algorithm for job shop scheduling problem. *Computers & Operations Research*, Elsevier, v. 39, n. 10, p. 2291–2299, 2012. ISSN 03050548. Citado na pág. 45, 119.

JIA, S.; HU, Z. H. Path-relinking Tabu search for the multi-objective flexible job shop scheduling problem. *Computers and Operations Research*, Elsevier, v. 47, p. 11–26, 2014. ISSN 03050548. Disponível em: <http://dx.doi.org/10.1016/j.cor.2014.01.010>. Citado na pág. 64.

JOLLIFFE, I. T. Discarding Variables in a Principal Component Analysis. I: Artificial Data. *Applied Statistics*, v. 21, n. 2, p. 160–173, 1972. ISSN 00359254. Citado na pág. 47, 56.

JOLLIFFE, I. T. Discarding Variables in a Principal Component Analysis. II: Real Data. *Journal of the Royal Statistical Society*, v. 22, n. 1, p. 21–31, 1973. Citado na pág. 47, 56.

JOLLIFFE, I. T. *Principal Component Analysis*. 2nd.. ed. New York: Springer-Verlag, 2002. 488 p. (Springer Series in Statistics). ISBN 0-387-95442-2. Disponível em: <https://doi.org/10.1007/b98835>. Citado na pág. 46, 66.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Piscataway: IEEE, 1995. v. 4, p. 1942–1948. ISBN 0-7803-2768-3. Citado na pág. 19.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by Simulated Annealing. *Science*, v. 220, n. 4598, p. 671–680, 1983. Citado na pág. 18.

KUHPFAHL, J.; BIERWIRTH, C. A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective. *Computers & Operations Research*, Elsevier, v. 66, p. 44–57, 2016. ISSN 03050548. Citado na pág. 18, 75.

KURDI, M. A new hybrid island model genetic algorithm for job shop scheduling problem. *Computers & Industrial Engineering*, Elsevier Ltd, v. 88, p. 273–283, 2015. ISSN 03608352. Citado na pág. 50, 53.

KURDI, M. An effective new island model genetic algorithm for job shop scheduling problem. *Computers & Operations Research*, Elsevier, v. 67, p. 132–142, 2016. ISSN 03050548. Citado na pág. 52, 53.

KUVAT, G.; ADAR, N. The analysis of the interaction of the migration, diversity and permeability in parallel genetic algorithms. *An International Journal of Optimization and Control: Theories & Applications (IJOCTA)*, v. 6, n. 1, p. 23–31, 2016. ISSN 2146-5703. Citado na pág. 45.

LAARHOVEN, P. J. M. van; AARTS, E. H. L.; LENSTRA, J. K. Job Shop Scheduling by Simulated Annealing. *Operations Research*, v. 40, n. 1, p. 113–125, feb 1992. ISSN 0030-364X. Citado na pág. 34, 59.

LAMOS-DÍAZ, H.; AGUILAR-IMITOLA, K.; PÉREZ-DÍAZ, Y. T.; GALVÁN-NÚÑEZ, S. A memetic algorithm for minimizing the makespan in the Job Shop Scheduling problem. *Revista Facultad de Ingeniería*, v. 26, n. 44, p. 111, jan 2017. ISSN 2357-5328. Citado na pág. 61.

LAWRENCE, S. *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement)*. Tese (Doutorado) — Graduate School of Industrial Administration, Carnegie-Mellon University, 1984. Citado na pág. 14, 60.

LEI, D. Simplified multi-objective genetic algorithms for stochastic job shop scheduling. *Applied Soft Computing Journal*, Elsevier B.V., v. 11, n. 8, p. 4991–4996, 2011. ISSN 15684946. Citado na pág. 52.

LENSTRA, J.; RINNOOY KAN, A.; BRUCKER, P. Complexity of Machine Scheduling Problems. In: *Journal of the Operational Research Society*. [S.l.: s.n.], 1977. v. 29, n. 8, p. 343–362. ISBN 9780720407655. Citado na pág. 17.

LI, X.; GAO, L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics*, Elsevier, v. 174, p. 93–110, 2016. ISSN 09255273. Disponível em: <http://dx.doi.org/10.1016/j.ijpe.2016.01.016>. Citado na pág. 64.

LI, X.; ZHANG, K. Single batch processing machine scheduling with two-dimensional bin packing constraints. *International Journal of Production Economics*, Elsevier B.V., v. 196, p. 113–121, 2018. ISSN 09255273. Citado na pág. 64.

LIMA, S. J. d. A.; ARAÚJO, S. A. de. A New Binary Encoding Scheme in Genetic Algorithm for Solving the Capacitated Vehicle Routing Problem. In: *Antimicrobial Compounds*. [S.l.: s.n.], 2018. p. 174–184. ISBN 9783319916415. Citado na pág. 39.

LINDEN, R. *Algoritmos Genéticos*. 3ª edição. ed. Rio de Janeiro: Editora Ciência Moderna, 2012. 496 p. ISBN 978-85-399-205-7. Citado na pág. 37, 39, 40, 44.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated Local Search. In: _____. *Handbook of Metaheuristics*. Boston, MA: Springer US, 2003. p. 320–353. ISBN 978-0-306-48056-0. Citado na pág. 18.

LOZANO, M.; GARCÍA-MARTÍNEZ, C. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress

report. *Computers and Operations Research*, Elsevier, v. 37, n. 3, p. 481–497, 2010. ISSN 03050548. Citado na pág. 45, 46, 64.

LU, H.; SHI, J.; FEI, Z.; ZHOU, Q.; MAO, K. Measures in the time and frequency domains for fitness landscape analysis of dynamic optimization problems. *Applied Soft Computing*, Elsevier B.V., v. 51, p. 192–208, 2017. ISSN 15684946. Citado na pág. 30.

LU, H.; SHI, J.; FEI, Z.; ZHOU, Q.; MAO, K. Analysis of the similarities and differences of job-based scheduling problems. *European Journal of Operational Research*, Elsevier Ltd, v. 270, n. 3, p. 809–825, nov. 2018. Citado na pág. 19, 55.

MALAN, K. M.; ENGELBRECHT, A. P. Fitness landscape analysis for metaheuristic performance prediction. In: *Recent Advances in the Theory and Application of Fitness Landscapes. Emergence, Complexity and Computation.* [S.l.]: Springer Berlin Heidelberg, 2014. v. 6, p. 103–132. Citado na pág. 19.

MATTFELD, D. C. *Evolutionary Search and the Job Shop - Investigations on Genetic Algorithms for Production Scheduling.* [S.l.]: Physica-Verlag, 1996. (Production and Logistics). Citado na pág. 30.

MATTFELD, D. C.; BIERWIRTH, C. An efficient genetic algorithm for job shop scheduling with tardiness objectives. *European Journal of Operational Research*, v. 155, n. 3, p. 616–630, 2004. ISSN 03772217. Citado na pág. 35.

MATTFELD, D. C.; BIERWIRTH, C.; KOPFER, H. A Search space analysis of the Job Shop Scheduling Problem. *Annals of Operations Research*, v. 86, p. 441–453, 1999. Citado na pág. 19, 30, 31, 32, 33, 34, 35.

MELLADO, R.; CUBILLOS, C.; CABRERA, D. A Constructive Heuristic for Solving the Job- Shop Scheduling Problem. *Ieee Latin America Transactions*, v. 14, n. 6, p. 2758–2763, 2016. ISSN 15480992. Citado na pág. 51.

MENDES, J. J. d. M. *Sistema de Apoio à Decisão para Planeamento de Sistemas de Produção Tipo Projecto.* 1–276 p. Tese (Doutorado) — Universidade do Porto, 2003. Disponível em: <http://repositorio-aberto.up.pt/bitstream/10216/12050/2/Texto integral.pdf>. Acesso em: 21-08-2015. Citado na pág. 28.

MICHALEWICZ, Z. *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.).* Berlin, Heidelberg: Springer-Verlag, 1996. ISBN 3-540-60676-9. Citado na pág. 36, 38.

MIRSHEKARIAN, S.; ŠORMAZ, D. N. Correlation of job-shop scheduling problem features with scheduling efficiency. *Expert Systems with Applications*, Elsevier Ltd, v. 62, p. 131–147, 2016. ISSN 09574174. Citado na pág. 19.

MITCHELL, M. *An Introduction to Genetic Algorithms.* Cambridge, MA, USA: MIT Press, 1998. ISBN 0262631857. Citado na pág. 36.

MITCHELL, T. M. *Machine Learning.* 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072. Citado na pág. 36, 39.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. *Computers & Operations Research*, v. 24, n. 11, p. 1097–1100, nov 1997. Citado na pág. 18.

MORALES, S. G.; RONCONI, D. P. Formulações matemáticas e estratégias de resolução para o problema job shop clássico. *Production*, v. 26, n. 3, p. 614–625, 2016. ISSN 1980-5411. Citado na pág. 17, 18.

MORITA, M.; OCHIAI, H.; TAMURA, K.; YASUDA, K. Multi-point Search Combinatorial Optimization Method Based on Neighborhood Search Using Evaluation of Big Valley Structure. *Proceedings - 2015 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2015*, p. 2835–2840, 2016. Citado na pág. 45.

MOSER, I.; GHEORGHITA, M.; ALETI, A. Identifying Features of Fitness Landscapes and Relating Them to Problem Difficulty. *Evolutionary Computation*, v. 25, n. 3, p. 407–437, 2017. Citado na pág. 31, 32.

NAGATA, Y.; ONO, I. A Guided Local Search with Iterative Ejections of Bottleneck Operations for the Job Shop Scheduling Problem. *Computers & Operations Research*, Elsevier Ltd, 2017. ISSN 03050548. Citado na pág. 20.

NASIRI, M. M.; KIANFAR, F. A guided tabu search/path relinking algorithm for the job shop problem. *International Journal of Advanced Manufacturing Technology*, v. 58, n. 9-12, p. 1105–1113, 2012. ISSN 02683768. Citado na pág. 74.

NERI, F.; COTTA, C. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, Elsevier B.V., v. 2, p. 1–14, 2012. ISSN 22106502. Citado na pág. 45.

NOOR, S.; LALI, M. I.; NAWAZ, M. S.; CAMPUS, S. Solving Job Shop Scheduling Problem with Genetic Algorithm. *Sci.Int.(Lahore)*, v. 27, n. 4, p. 3367–3371, 2015. ISSN 1013-5316. Citado na pág. 17.

NOWICKI, E.; SMUTNICKI, C. A fast taboo search algorithm for the job shop problem. *Management Science*, v. 42, n. 6, p. 797–813, 1996. Citado na pág. 33, 36.

NOWICKI, E.; SMUTNICKI, C. An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling*, v. 8, n. 2, p. 145–159, 2005. ISSN 10946136. Citado na pág. 19, 35, 36, 56, 74.

NOWICKI, E.; SMUTNICKI, C. Some New Ideas in TS for Job Shop Scheduling. In: SHARDA, R.; VOSS, S.; REGO, C.; ALIDAEE, B. (Ed.). *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*. Boston, MA: Springer US, 2005. p. 165–190. Citado na pág. 33.

OLIVEIRA, R. M. de; CHAVES, A. A.; LORENA, L. A. N. A comparison of two hybrid methods for constrained clustering problems. *Applied Soft Computing Journal*, Elsevier B.V., v. 54, p. 256–266, 2017. ISSN 15684946. Citado na pág. 64.

PARDALOS, P. M.; SHYLO, O. V.; VAZACOPOULOS, A. Solving job shop scheduling problems utilizing the properties of backbone and "big valley". *Computational Optimization and Applications*, v. 47, n. 1, p. 61–76, sep 2010. ISSN 0926-6003. Citado na pág. 19, 56.

PENG, B.; LÜ, Z.; CHENG, T. A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Computers & Operations Research*, Elsevier, v. 53, p. 154–164, jan 2015. ISSN 03050548. Citado na pág. 47, 51, 74.

PÉREZ, E.; HERRERA, F.; HERNÁNDEZ, C. Finding multiple solutions in job shop scheduling by niching genetic algorithms. *Journal of Intelligent Manufacturing*, v. 14, n. 3-4, p. 323–339, 2003. ISSN 09565515. Citado na pág. 82.

PÉREZ, E.; POSADA, M.; HERRERA, F. Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling. *Journal of Intelligent Manufacturing*, v. 23, n. 3, p. 341–356, 2012. ISSN 09565515. Citado na pág. 61, 82, 86.

PÉREZ, M. P.; RODRÍGUEZ, F. A.; VEGA, J. M. M. On the use of path relinking for the $\rho$ – hub median problem. In: GOTTLIEB, J.; RAIDL, G. R. (Ed.). *Evolutionary Computation in Combinatorial Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 155–164. ISBN 978-3-540-24652-7. Citado na pág. 48.

PINEDO, M. L. *Scheduling: Theory, Algorithms, and Systems*. 4th.. ed. Boston, MA: Springer US, 2012. Citado na pág. 17, 18, 23, 28.

REEVES, C. R. Landscapes, operators and heuristic search. *Annals of Operations Research*, Springer Nature, v. 86, p. 473–490, 1999. Citado na pág. 35.

REEVES, C. R.; EREMEEV, A. V. Statistical analysis of local search landscapes. *Journal of the Operational Research Society*, Informa UK Limited, v. 55, n. 7, p. 687–693, 2004. Citado na pág. 54.

REIDYS, C. M.; STADLER, P. F. Combinatorial landscapes. *SIAM Review*, Society for Industrial & Applied Mathematics (SIAM), v. 44, n. 1, p. 3–54, jan 2002. Citado na pág. 32, 54.

RESENDE, M. G.; RIBEIRO, C. C. GRASP with Path-Relinking: Recent Advances and Applications. In: IBARAKI, T.; NONOBE, K.; YAGIURA, M. (Ed.). *Metaheuristics: Progress as Real Problem Solvers*. Boston, MA: Springer-Verlag, 2005. cap. 2, p. 29–63. ISBN 978-0-387-25383-1. Citado na pág. 48.

RESENDE, M. G. C. Biased random-key genetic algorithms with applications in telecommunications. *TOP*, v. 20, n. 1, p. 130–153, apr 2012. ISSN 1134-5764. Citado na pág. 64.

RESENDE, M. G. C.; RIBEIRO, C. C. Introdution. In: _____. *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*. New York, NY: Springer New York, 2016. cap. 1, p. 1–11. ISBN 9781493965304. Citado na pág. 22.

RESENDE, M. G. C.; RIBEIRO, C. C. Local search. In: _____. *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*. New York, NY: Springer New York, 2016. cap. 4, p. 63–93. ISBN 9781493965304. Citado na pág. 35.

RESENDE, M. G. C.; RIBEIRO, C. C. Path-relinking. In: *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*. New York, NY: Springer New York, 2016. cap. 8, p. 167–188. ISBN 9781493965304. Citado na pág. 74.

RIBEIRO, C. C.; RESENDE, M. G. C. Path-relinking intensification methods for stochastic local search algorithms. *Journal of Heuristics*, v. 18, n. 2, p. 193–214, 2012. ISSN 1381-1231. Citado na pág. 47.

SADOWSKI, Ł.; NIKOO, M.; NIKOO, M. Principal Component Analysis combined with a Self Organization Feature Map to determine the pull-off adhesion between concrete layers. *Construction and Building Materials*, v. 78, p. 386–396, 2015. ISSN 09500618. Citado na pág. 46, 66.

SANTO, R. D. E. Utilização da Análise de Componentes Principais na compressão de imagens digitais. *Einstein*, v. 10, n. 11, p. 135–139, 2012. Citado na pág. 46.

SANTO, R. D. E.; PEREIRA, F. H.; JÚNIOR, E. A. Image Compression Based on Generalized Principal Components Analysis and Simulated Annealing. *International Journal of Cognitive Informatics and Natural Intelligence*, v. 6, n. 2, p. 41–67, 2012. ISSN 1557-3958. Citado na pág. 46.

SCHIMIT, P. H.; PEREIRA, F. H. Disease spreading in complex networks: A numerical study with Principal Component Analysis. *Expert Systems with Applications*, Elsevier Ltd, v. 97, p. 41–50, 2018. ISSN 09574174. Citado na pág. 56, 66, 67.

SHA, D. Y.; LIN, H. H. A multi-objective PSO for job-shop scheduling problems. *Expert Systems with Applications*, Elsevier Ltd, v. 37, n. 2, p. 1065–1070, 2010. ISSN 09574174. Citado na pág. 35, 51.

SILVA, R. M. A.; RESENDE, M. G. C. Algoritmo genético de chaves aleatórias viciadas para problemas de otimização global com restrições de caixa sujeitas a restrições não-lineares. In: *Simpósio Brasileiro de Pesquisa Operacional - XLV SBPO*. [S.l.: s.n.], 2013. p. 3670–3679. Citado na pág. 42, 43.

SILVA, R. M. A.; RESENDE, M. G. C.; PARDALOS, P. M.; FACO, J. L. Biased random-key genetic algorithm for nonlinearly-constrained global optimization. In: *2013 IEEE Congress on Evolutionary Computation*. [S.l.]: IEEE, 2013. v. 1, n. 6, p. 2201–2206. ISBN 978-1-4799-0454-9. Citado na pág. 43, 52.

SMITH-MILES, K.; LOPES, L. Measuring instance difficulty for combinatorial optimization problems. *Computers and Operations Research*, Elsevier, v. 39, n. 5, p. 875–889, 2012. ISSN 03050548. Citado na pág. 30.

SMUTNICKI, C.; BOŻEJKO, W. Tabu search and solution space analyses. The job shop case. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 10671 LNCS, p. 383–391, 2018. ISSN 16113349. Citado na pág. 35, 56.

STADLER, P. F.; SCHNABL, W. The landscape of the traveling salesman problem. *Physics Letters A*, Elsevier, v. 161, n. 4, p. 337–344, 1992. Citado na pág. 54.

STREETER, M. J.; SMITH, S. F. Charecterizing the distribution of low-makespan schedules in the job shop scheduling problem. In: *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005), June 5-10 2005, Monterey, California, USA*. [S.l.]: AAAI, 2005. p. 61–70. Citado na pág. 55.

STREETER, M. J.; SMITH, S. F. How the landscape of random job shop scheduling instances depends on the ratio of jobs to machines. *Journal of Artificial Intelligence Research*, v. 26, p. 247–287, sep 2006. ISSN 10769757. Citado na pág. 55, 82.

TAILLARD, É. D. Benchmarks for basic sheduling problems. *European Journal of Operational Research*, v. 64, p. 278–285, 1993. Citado na pág. 56.

TAMSSAOUET, K.; DAUZÈRE-PÉRÈS, S.; YUGMA, C. Metaheuristics for the job-shop scheduling problem with machine availability constraints. *Computers and Industrial Engineering*, v. 125, p. 1–8, 2018. ISSN 03608352. Citado na pág. 50.

TAYARANI-NAJARAN, M.-H.; PRÜGEL-BENNETT, A. Anatomy of the fitness landscape for dense graph-colouring problem. *Swarm and Evolutionary Computation*, Elsevier, v. 22, p. 47–65, jun 2015. ISSN 22106502. Citado na pág. 54.

TAYARANI-NAJARAN, M.-H.; PRÜGEL-BENNETT, A. An Analysis of the Fitness Landscape of Travelling Salesman Problem. *Evolutionary Computation*, v. 24, n. 2, p. 347–384, jun 2016. ISSN 1063-6560. Citado na pág. 54, 57.

TOSO, R. F.; RESENDE, M. G. C. A C++application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, v. 30, n. 1, p. 81–93, jan 2015. ISSN 1055-6788. Citado na pág. 19, 41, 59, 64.

VANHATALO, E.; KULAHCI, M.; BERGQUIST, B. On the structure of dynamic principal component analysis used in statistical process monitoring. *Chemometrics and Intelligent Laboratory Systems*, Elsevier Ltd, v. 167, n. May, p. 1–11, 2017. ISSN 18733239. Citado na pág. 46.

WALL, M. *GAlib: A C++ library of genetic algorithm components*. [S.l.]: Mechanical Engineering Department, Massachussetts Institute of Technology, 2007. Citado na pág. 59.

WANG, B.; WANG, X.; LAN, F.; PAN, Q. A hybrid local-search algorithm for robust job-shop scheduling under scenarios. *Applied Soft Computing*, Elsevier B.V., v. 62, p. 259–271, jan 2018. ISSN 15684946. Citado na pág. 75.

WANG, W.; LI, T. Improved cultural algorithms for job shop scheduling problem. *International Journal of Industrial Engineering : Theory Applications and Practice*, v. 18, n. 4, p. 162–168, 2011. ISSN 10724761. Citado na pág. 78.

WANG, X.; DUAN, H. A hybrid biogeography-based optimization algorithm for job shop scheduling problem. *Computers & Industrial Engineering*, Elsevier Ltd, v. 73, n. April, p. 96–114, 2014. Citado na pág. 119.

WATSON, J.-P. An Introduction to Fitness Landscape Analysis and Cost Models for Local Search. In: *Handbook of Metaheuristics*. [S.l.: s.n.], 2010. p. 599–623. ISBN 978-1-4419-1663-1. Citado na pág. 19, 35.

WATSON, J.-P.; BECK, J.; HOWE, A. E.; WHITLEY, L. Problem difficulty for tabu search in job-shop scheduling. *Artificial Intelligence*, v. 143, n. 2, p. 189–217, feb 2003. ISSN 00043702. Citado na pág. 33.

WATSON, J.-P.; HOWE, A. E.; WHITLEY, L. Deconstructing Nowicki and Smutnicki's i-TSAB tabu search algorithm for the job-shop scheduling problem. *Computers & Operations Research*, v. 33, n. 9, p. 2623–2644, sep 2006. ISSN 03050548. Citado na pág. 19, 50, 51.

WONG, L.-P.; PUAN, C. Y.; LOW, M. Y. H.; CHONG, C. S. Bee Colony Optimization algorithm with Big Valley landscape exploitation for Job Shop Scheduling problems. *2008 Winter Simulation Conference*, p. 2050–2058, 2008. Citado na pág. 19, 33, 56.

WONG, L.-P.; PUAN, C. Y.; LOW, M. Y. H.; WONG, Y. W.; CHONG, C. S. Bee colony optimisation algorithm with big valley landscape exploitation for job shop scheduling problems. *International Journal of Bio-Inspired Computation*, v. 2, n. 2, p. 85–99, 2010. ISSN 1758-0366. Citado na pág. 19, 35, 56.

WRIGHT, S. *The roles of mutation, inbreeding, crossbreeding and selection in evolution.* 1932. 356–366 p. Citado na pág. 31.

YAMADA, T. *Studies on metaheuristics for jobshop and flowshop scheduling problems.* 133 p. Tese (Doutorado) — Kyto Universit, Kyoto, Japan, november 2003. Disponível em: <http://www.kecl.ntt.co.jp/as/members/yamada/YamadaThesis.pdf>. Acesso em: 27-08-2015. Citado na pág. 23, 24, 25, 26, 27, 30.

YAMADA, T.; NAKANO, R. Chapter 7: Job shop scheduling. In: ZALZALA, A. M. S.; FLEMING, P. J. (Ed.). *Genetic algorithms in engineering systems.* London: Institution of Electrical Engineers, 1997. cap. Chapter 7, p. 134–160. ISBN 0852969023. Citado na pág. 23, 34.

YAN, Y.; SOHN, H. suk; REYES, G. A modified ant system to achieve better balance between intensification and diversification for the traveling salesman problem. *Applied Soft Computing Journal*, Elsevier B.V., v. 60, p. 256–267, 2017. ISSN 15684946. Disponível em: <http://dx.doi.org/10.1016/j.asoc.2017.06.049>. Citado na pág. 64.

YAZDANI, M.; ALETI, A.; KHALILI, S. M.; JOLAI, F. Optimizing the sum of maximum earliness and tardiness of the job shop scheduling problem. *Computers and Industrial Engineering*, v. 107, p. 12–24, 2017. ISSN 03608352. Citado na pág. 17.

ZAHEDI, J.; ROUNAGHI, M. M. Application of artificial neural network models and principal component analysis method in predicting stock prices on Tehran Stock Exchange. *Physica A: Statistical Mechanics and its Applications*, Elsevier B.V., v. 438, p. 178–187, 2015. ISSN 03784371. Citado na pág. 46.

ZHANG, C. Y.; LI, P.; GUAN, Z.; RAO, Y. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research*, v. 34, n. 11, p. 3229–3242, 2007. ISSN 03050548. Citado na pág. 34, 50.

ZHANG, C. Y.; LI, P. G.; RAO, Y. Q.; GUAN, Z. L. A very fast TS/SA algorithm for the job shop scheduling problem. *Computers and Operations Research*, v. 35, n. 1, p. 282–294, 2008. ISSN 03050548. Citado na pág. 36.

ZHANG, H.; LIU, S.; MORACA, S.; OJSTERSEK, R. An Effective Use of Hybrid Metaheuristics Algorithm for Job Shop Scheduling Problem. *International Journal of Simulation Modelling*, v. 16, n. 4, p. 644–657, 2016. ISSN 17264529. Citado na pág. 48.

ZHANG, R.; CHIONG, R. Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *Journal of Cleaner Production*, Elsevier Ltd, v. 112, p. 3361–3375, 2016. ISSN 09596526. Citado na pág. 45.

ZHAO, F.; LIU, Y.; ZHANG, Y.; MA, W.; ZHANG, C. A hybrid harmony search algorithm with efficient job sequence scheme and variable neighborhood search for the permutation flow shop scheduling problems. *Engineering Applications of Artificial Intelligence*, Elsevier Ltd, v. 65, n. October 2016, p. 178–199, 2017. ISSN 09521976. Citado na pág. 18.

ZHAO, F.; ZHANG, J.; ZHANG, C.; WANG, J. An improved shuffled complex evolution algorithm with sequence mapping mechanism for job shop scheduling problems. *Expert Systems with Applications*, Elsevier Ltd, v. 42, n. 8, p. 3953–3966, 2015. ISSN 09574174. Citado na pág. 51.

ZUDIO, A.; da Silva Costa, D. H.; MASQUIO, B. P.; COELHO, I. M.; PINTO, P. E. D. BRKGA/VND Hybrid Algorithm for the Classic Three-dimensional Bin Packing Problem. *Electronic Notes in Discrete Mathematics*, v. 66, p. 175–182, 2018. ISSN 15710653. Citado na pág. 64.