



# Adaptação de Parâmetros em Algoritmos Genéticos para o problema de Escalonamento *Job-Shop*

**Marcus Vinicius da Silva**

JUIZ DE FORA

JULHO, 2019

# Adaptação de Parâmetros em Algoritmos Genéticos para o problema de Escalonamento *Job-Shop*

Marcus Vinicius da Silva

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Heder Soares Bernardino

JUIZ DE FORA

JULHO, 2019

# Adaptação de Parâmetros em Algoritmos Genéticos para o problema de Escalonamento *Job-Shop*

Marcus Vinicius da Silva

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Heder Soares Bernardino
Doutor em Modelagem Computacional (LNCC)

Lorenza Leão Oliveira Moreno
Doutora em Informática (PUC-Rio)

Luciana Brugiolo Gonçalves
Doutora em Computação (UFF)

JUIZ DE FORA
05 DE JULHO, 2019

*Aos meus amigos que se tornaram irmãos ao longo do tempo.*
*A minha mãe pelo apoio na área que escolhi.*
*Às políticas públicas de acesso e de permanência na Universidade, que favoreceram pessoas sem condição financeira a chegar onde cheguei.*

# Resumo

Problemas de escalonamento são comumente encontrados em várias áreas de pesquisa e no dia a dia das empresas. Montadoras e empresas de reparação de veículos são exemplos de como estes tipos de problemas estão relacionados com tarefas cotidianas. Direta ou indiretamente, uma boa alocação de pessoas ou máquinas para a execução de um serviço influencia características qualitativas de uma empresa ou um profissional, tais como: rapidez na execução de um serviço e economia na geração de um produto. O problema de Escalonamento *Job Shop* consiste em alocar tarefas à um conjunto de máquinas, respeitando a sequência de execução independente de cada tarefa em cada máquina. Dada a complexidade de resolução exata deste problema, são necessárias ferramentas computacionais para auxiliar a tomada de decisão neste contexto. Os Algoritmos Genéticos são alternativas para encontrar soluções de qualidade, mas achar os melhores parâmetros para a sua execução se torna outra questão a ser resolvida. Surge então a necessidade de utilização de técnicas de adaptação de parâmetros e operadores para definir, em tempo de execução, quais valores serão utilizados no algoritmo. Este trabalho tem como objetivo analisar o desempenho de um Algoritmo Genético (AG) que tem seus parâmetros e operadores definidos pelo usuário com Algoritmos Genéticos Adaptativos (AGAs), que utilizam esquemas de recompensa e seleção de operadores e parâmetros durantes o processo de busca. Dois conjuntos de experimentos foram realizados, o primeiro a fim de encontrar qual o algoritmo tem o melhor desempenho para um problema específico, e o segundo experimento, o algoritmo que melhor generaliza o comportamento para problemas de diferentes tamanhos. Para a comparação das propostas, foram utilizados os Perfis de Desempenho, análises de frequências de operadores utilizados nos AGAs e análises estatísticas.

**Palavras-chave:** Algoritmos Genéticos, Adaptação de Parâmetros, Job Shop

# Abstract

Scheduling problems are commonly found in various areas of research and day to day of companies. Car makers and vehicle repair companies are examples of how these types of problems are related to everyday tasks. Directly or indirectly, a good allocation of people or machines for the execution of a service influences the qualitative characteristics of a company or a professional, such as: speed in the execution of a service and economy in the generation of a product. The Problem of Job Shop Scheduling consists in allocating tasks to a set of machines, respecting the sequence of independent execution of each task in each machine. Given the exact resolution complexity of this problem, computational tools are needed to aid decision making in this context. Genetic Algorithms are alternatives to find quality solutions, but finding the best parameters for their execution becomes another issue to be solved. The need to use parameter adaptation techniques and operators to define, at run time, which values will be used in the algorithm. This work aims to analyze the performance of a Genetic Algorithm (AG) with its user-defined parameters and operators with Adaptive Genetic Algorithms (AGAs), which use reward schemes and operator selection and parameters during the search process. Two sets of experiments were performed, the first in order to find which algorithm has the best performance for a specific problem, and the second experiment, the algorithm that best generalizes the behavior for problems of different sizes. For the comparison of the proposals, the Performance Profiles, operator frequency analyzes used in the AGAs and statistical analyzes were used.

**Keywords:** Genetic Algorithms, Parameters Adjustment, Job Shop.

# Agradecimentos

A minha mãe pela educação dada ao longo dos anos e pelo apoio às decisões que eu tomei.

Ao meu irmão por ser sempre uma bússola moral e intelectual.

Aos meus amigos pelos constantes incentivos para que eu não desistisse no meio do caminho.

As políticas de acesso e permanência na Universidade, que mesmo em tempos de ameaça, tornam sonhos de pessoas necessitadas em realidade.

*"Existem questões a cuja resposta eu daria um valor infinitamente maior do que às matemáticas, por exemplo questões sobre ética, sobre nosso relacionamento com Deus, sobre nosso destino e nosso futuro. Para a alma existe uma satisfação de espécie superior, para a qual dispenso o que é material".*

*Karl Friedrich Gauss*

# Conteúdo

**Lista de Figuras** **8**

**Lista de Tabelas** **11**

**Lista de Abreviações** **13**

**1 Introdução** **14**

**2 O problema de Escalonamento Job-Shop** **17**

**3 Revisão Bibliográfica** **19**
3.1 Abordagens em problemas *Job-Shop* . . . . . 19
3.2 Algoritmos Genéticos em Problemas *Job-Shop* . . . . . 22
3.3 Adaptação de Parâmetros . . . . . 23

**4 Algoritmos Genéticos em Problemas Job-Shop** **25**

**5 Algoritmos Genéticos Adaptativos** **31**
5.1 *Adaptive Pursuit* com Valor Extremo (APEX) . . . . . 32
5.2 *Rank-based Multi-Armed Bandit* com *Sum of Ranks* (RMAB) . . . . . 34

**6 Experimentos Computacionais** **36**
6.1 Perfis de Desempenho . . . . . 38
6.2 Experimento 1 . . . . . 39
6.2.1 Fase de configuração . . . . . 39
6.2.2 15 máquinas e 15 *jobs* . . . . . 40
6.2.3 20 máquinas e 15 *jobs* . . . . . 43
6.2.4 20 máquinas e 20 *jobs* . . . . . 46
6.2.5 30 máquinas e 15 *jobs* . . . . . 49
6.2.6 30 máquinas e 20 *jobs* . . . . . 52
6.2.7 50 máquinas e 15 *jobs* . . . . . 55
6.2.8 Todos os problemas . . . . . 58
6.2.9 Análise do Experimento 1 . . . . . 61
6.3 Experimento 2 . . . . . 62
6.3.1 Fase de configuração . . . . . 62
6.3.2 15 máquinas e 15 *jobs* . . . . . 63
6.3.3 20 máquinas e 15 *jobs* . . . . . 66
6.3.4 20 máquinas e 20 *jobs* . . . . . 69
6.3.5 30 máquinas e 15 *jobs* . . . . . 72
6.3.6 30 máquinas e 20 *jobs* . . . . . 75
6.3.7 50 máquinas e 15 *jobs* . . . . . 78
6.3.8 Todos os problemas . . . . . 81
6.3.9 Análise do Experimento 2 . . . . . 81

**7 Conclusões e Trabalhos Futuros** **85**

# Lista de Figuras


2.1 Exemplo de uma configuração de problema com 3 *Jobs* e 3 Máquinas . . . 18
2.2 Uma solução para o exemplo da Figura 2.1 com *Makespan* = 12. . . . . 18

3.1 Exemplo de Representação da Regra de Despacho RD + (DD - SP). Retirado de (ABEDNEGO; HENDRATMO, 2011). . . . . 21

4.1 Exemplo de codificação de um indivíduo com 10 *jobs* . . . . . 25
4.2 Exemplo OPX. . . . . 27
4.3 Exemplo LOX. . . . . 27
4.4 Exemplo PMX. . . . . 28
4.5 Exemplo OBX. . . . . 28
4.6 Exemplo SM. . . . . 28
4.7 Exemplo PM. . . . . 29
4.8 Exemplo IM. . . . . 29

6.1 Exemplo PD (FERREIRA, 2018). . . . . 39
6.2 Perfis de Desempenho para o problema 15_15. . . . . 41
6.3 Área Normalizada dos PDs para o problema 15_15. . . . . 41
6.4 Frequência Relativa dos Operadores utilizados nos AGAs para o problema 15_15. . . . . 42
6.5 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 15_15. . . . . 42
6.6 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 15_15. . . . . 43
6.7 Perfis de Desempenho para o problema 20_15. . . . . 44
6.8 Área Normalizada dos PDs para o problema 20_15. . . . . 44
6.9 Frequência Relativa dos Operadores utilizados nos AGAs para o problema 20_15. . . . . 45
6.10 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 20_15. . . . . 45
6.11 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 20_15. . . . . 46
6.12 Perfis de Desempenho para o problema 20_20. . . . . 47
6.13 Área Normalizada dos PDs para o problema 20_20. . . . . 47
6.14 Frequência Relativa dos Operadores utilizados nos AGAs para o problema 20_20. . . . . 48
6.15 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 20_20. . . . . 48
6.16 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 20_20. . . . . 49
6.17 Perfis de Desempenho para o problema 30_15. . . . . 50
6.18 Área Normalizada dos PDs para o problema 30_15. . . . . 50
6.19 Frequência Relativa dos Operadores utilizados nos AGAs para o problema 30_15. . . . . 51

6.20 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 30_15. . . . . 51
6.21 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 30_15. . . . . 52
6.22 Perfis de Desempenho para o problema 30_20. . . . . 53
6.23 Área Normalizada dos PDs para o problema 30_20. . . . . 53
6.24 Frequência Relativa dos Operadores utilizados nos AGAs para o problema 30_20. . . . . 54
6.25 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 30_20. . . . . 54
6.26 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 30_20. . . . . 55
6.27 Perfis de Desempenho para o problema 50_15. . . . . 56
6.28 Área Normalizada dos PDs para o problema 50_15. . . . . 56
6.29 Frequência Relativa dos Operadores utilizados nos AGAs para o problema 50_15. . . . . 57
6.30 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 50_15. . . . . 57
6.31 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 50_15. . . . . 58
6.32 Perfis de Desempenho para todos os problemas. . . . . 59
6.33 Área Normalizada dos PDs para todos os problemas. . . . . 59
6.34 Frequência Relativa dos Operadores utilizados nos AGAs para todos os problemas. . . . . 60
6.35 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para todos os problemas. . . . . 60
6.36 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para todos os problemas. . . . . 61
6.37 Perfis de Desempenho para o problema 15_15. . . . . 63
6.38 Área Normalizada dos PDs para o problema 15_15. . . . . 64
6.39 Frequência Relativa dos Operadores utilizados nos AGAs para o problema 15_15. . . . . 64
6.40 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 15_15. . . . . 65
6.41 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 15_15. . . . . 65
6.42 Perfis de Desempenho para o problema 20_15. . . . . 66
6.43 Área Normalizada dos PDs para o problema 20_15. . . . . 67
6.44 Frequência Relativa dos Operadores utilizados nos AGAs para o problema 20_15. . . . . 67
6.45 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 20_15. . . . . 68
6.46 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 20_15. . . . . 68
6.47 Perfis de Desempenho para o problema 20_20. . . . . 69
6.48 Área Normalizada dos PDs para o problema 20_20. . . . . 70
6.49 Frequência Relativa dos Operadores utilizados nos AGAs para o problema 20_20. . . . . 70
6.50 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 20_20. . . . . 71

6.51 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 20_20. . . . . 71
6.52 Perfis de Desempenho para o problema 30_15. . . . . 72
6.53 Área Normalizada dos PDs para o problema 30_15. . . . . 73
6.54 Frequência Relativa dos Operadores utilizados nos AGAs para o problema 30_15. . . . . 73
6.55 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 30_15. . . . . 74
6.56 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 30_15. . . . . 74
6.57 Perfis de Desempenho para o problema 30_20. . . . . 75
6.58 Área Normalizada dos PDs para o problema 30_20. . . . . 76
6.59 Frequência Relativa dos Operadores utilizados nos AGAs para o problema 30_20. . . . . 76
6.60 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 30_20. . . . . 77
6.61 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 30_20. . . . . 77
6.62 Perfis de Desempenho para o problema 50_15. . . . . 78
6.63 Área Normalizada dos PDs para o problema 50_15. . . . . 79
6.64 Frequência Relativa dos Operadores utilizados nos AGAs para o problema 50_15. . . . . 79
6.65 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 50_15. . . . . 80
6.66 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 50_15. . . . . 80
6.67 Perfis de Desempenho para todos os problemas . . . . . 81
6.68 Área Normalizada dos PDs para todos os problemas . . . . . 82
6.69 Frequência Relativa dos Operadores utilizados nos AGAs para todos os problemas. . . . . 82
6.70 Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para todos os problemas. . . . . 83
6.71 Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para todos os problemas. . . . . 83

# Lista de Tabelas

6.1 Parâmetros utilizados na fase de Configuração dos AGs. . . . . 37
6.2 Hiper-Parâmetros utilizados na fase de Configuração dos AGAs. . . . . 38
6.3 Parâmetros dos AGs escolhidos para cada configuração de problema no Experimento 1. . . . . 40
6.4 Hiper-Parâmetros dos AGAs escolhidos para cada configuração de problema no Experimento 1. . . . . 40
6.5 Parâmetros dos AGs escolhidos a partir do problema 15x15 no Experimento 2. . . . . 62
6.6 Parâmetros dos AGAs escolhidos a partir do problema 15x15 no Experimento 2. . . . . 63

# Lista de Abreviações

| | |
|---|---|
| DCC | Departamento de Ciência da Computução |
| UFJF | Universidade Federal de Juiz de Fora |
| JSS | *Job-Shop Scheduling* |
| DJSS | *Dynamic Job-Shop Scheduling* |
| AG | Algoritmo Genético |
| AGA | Algoritmo Genético Adaptativo |
| SA | *Simulated Annealing* |
| BS | *Beam Search* |
| PSO | *Particle Swarm Optimization* |
| ABC | *Artificial Bee Colony* |
| TS | *Tabu Search* |
| PG | Programação Genética |
| ILS | *Iterated Local Search* |
| ACO | *Ant Colony Optimization* |
| GOX | *Generalizated Order-Crossover* |
| TSP | *Traveling Salesman Problem* |
| OPX | *Order-Preserving One-Point Crossover* |
| LOX | *Linear-Order Crossover* |
| PMX | *Partially Mapped Crossover* |
| OBX | *Order-Based Crossover* |
| SM | *Shift Mutation* |
| PM | *Pairwise Interchange Neighborhood Mutation* |
| IM | *Inverse Mutation (Neighborhood)* |
| AP | *Adaptive Pursuit* |
| RMAB | *Ranked Multi-Armed Bandit* |
| APEX | *Adaptive Pursuit with Extreme Value* |
| RMABSR | *Ranked Multi-Armed Bandit with Sum of Ranks* |
| PPC | *Predictive Parameter Control* |
| MAB | *Multi-Armed Bandit* |

# 1 Introdução

Problemas de escalonamento são comumente encontrados em várias áreas de pesquisa e dia a dia de empresas. Montadoras e empresas de reparação de veículos são exemplos de como estes tipos de problemas estão relacionados com tarefas cotidianas. Direta ou indiretamente, uma boa alocação de pessoas ou máquinas para a execução de um serviço influencia características qualitativas de uma empresa ou um profissional, tais como: rapidez na execução de um serviço e economia na geração de um produto.

Considerando um ambiente industrial, um dos fatores que torna uma empresa competitiva no mercado é um bom planejamento de sua produção (ZHANG; SONG; WU, 2013). A organização de tarefas e processos influi diretamente em fatores como: atraso e/ou adiantamento da produção, tempo de ociosidade das máquinas, tempo de preparação das máquinas, custo com estocagem de produtos, etc (BILKAY; ANLAGAN; KILIC, 2004). Portanto, solucionar um complexo planejamento de produção consiste em achar uma organização ideal dos recursos utilizados no processo (máquinas e produto), de forma a atender um conjunto de exigências do problema em questão (ÇALIş; BULKAN, 2015).

São vários os tipos de problemas de escalonamento conhecidos na área de Pesquisa Operacional. Dentre eles, o escalonamento *Job-Shop* foi escolhido como tema deste trabalho dada a alta complexidade de encontrar boas soluções com orçamentos computacionais limitados (ASADZADEH; ZAMANIFAR, 2010).

O problema de escalonamento *Job-Shop* envolve um produto e processos que devem ser aplicados à ele (Conjunto de Máquinas). Em um contexto prático, pode-se imaginar uma ferramenta que necessita de processos de corte, pintura e polimento. Este tipo de problema permite diferentes fluxos das ordens de produção entre as máquinas e os diferentes números de operações por ordem, que são processadas apenas uma vez em cada máquina. Portanto, definir uma ordem de processamento das tarefas em cada máquina é uma solução para esse tipo de problema (CHENG; GEN; TSUJIMURA, 1996).

À medida que o problema cresce, cresce também a necessidade do uso de ferra-

mentas computacionais no auxílio à tomada de decisão quanto à organização do processo de produção. Técnicas que utilizam formulações matemáticas (*Branch-and-Bound*) e algoritmos evolucionários de busca como Algoritmos Genéticos (AGs) e Colônia de Formigas (ACO) têm sido opções utilizadas para solucionar problemas deste tipo ao longo dos anos (CHAUDHRY; KHAN, 2016).

A utilização de Algoritmos de Busca para resolver um problema particular necessita da especificação, por parte do usuário, da representação e uma forma de avaliação de uma solução candidata. Mais especificamente nos Algoritmos Evolucionários (AE) outras decisões de projeto devem ser tomadas, tais como: operadores de movimento, mecanismos de seleção, etc. Cada componente utilizado pode conter um parâmetro associado, e a escolha destes valores influencia diretamente no desempenho do algoritmo (EIBEN; HINTERDING; MICHALEWICZ, 1999). Portanto, ao configurar os parâmetros de um AE, as características do problema devem ser levadas em consideração (KREMPSER; FIALHO; BARBOSA, 2012).

Configurar um AG, por exemplo, é definir valores de parâmetros como: operadores de movimento e suas taxas de aplicação, tamanho da população de indivíduos, taxa de aplicação de busca local, etc. Definir manualmente estes parâmetros se torna outro problema à ser resolvido e como o processo de busca é dinâmico, valores determinados no início da execução do algoritmo podem fazer com que este processo fique estagnado em algum ponto. Para contornar este tipo de problema, técnicas de adaptação de parâmetros ao longo do processo de busca surgem como alternativa à questão do desempenho dos AE (ALETI; MOSER, 2016).

Fialho (2010) divide o processo de escolha de um operador (parâmetro) em duas fases: o *feedback* de sua aplicação, denominada Atribuição de Crédito (recompensa); e a Técnica de Seleção de Operadores, que utiliza a informação de recompensa para definir como serão escolhidos os operadores naquela fase. Algumas técnicas de atribuição de crédito podem ser destacadas: a Recompensa Instantânea (RI), onde o valor da diferença entre a aptidão do indivíduo antes e depois da aplicação de um operador é considerada; a Recompensa baseada em Valor Extremo, que utiliza um histórico de recompensas instantâneas de um operador e escolhe sempre a de maior valor. As técnicas *Dynamic*

*Multi-Armed Bandit* e *Sliding Multi-Armed Bandit* adaptadas do problema Multi-Armed Bandit (GITTINS et al., 1989) são referência no contexto da Seleção de Operadores.

A proposta deste trabalho visa comparar o desempenho para o problema de Escalonamento *Job-Shop* de um conjunto de Algoritmos Genéticos Adaptativos (AGAs) com AGs que utilizam parâmetros escolhidos pelo usuário e analisar se o custo computacional gasto com a seleção de parâmetros para o AG é compensado pela utilização de AGAs. O *Adaptive Pursuit* com valor extremo (APEX) e o *Ranked Multi-Armed Bandit* com *Sum of Ranks* (RMABSR) foram os AGAs utilizados neste trabalho. Dois AGs que utilizam as distribuições de frequência dos operadores utilizados nos AGAs (PROB_APEX, PROB_RMAB) também serão considerados na fase de experimentos.

Para avaliar o desempenho dos algoritmos em questão foram definidos dois tipos de experimentos: o primeiro avalia qual algoritmo tem o melhor desempenho para um problema específico e o segundo experimento tem como objetivo encontrar qual algoritmo tem melhor capacidade de resolução de problemas distintos. Os perfis de desempenho (PDs) foram utilizados como ferramenta gráfica na visualização dos resultados dos 5 algoritmos utilizados.

# 2 O problema de Escalonamento Job-Shop

O problema de escalonamento *Job-Shop* pode ser descrito como um conjunto de $n$ *Jobs* $J_i$ onde $i = 1, 2, ..., n$ que devem ser processados em um conjunto de $m$ máquinas $M_k$ onde $k = 1, 2, ..., m$. Cada *Job* consiste em uma cadeia de operações que são processadas sem interrupção em um determinado subconjunto de máquinas por um determinado período de tempo (TAMSSAOUET; DAUZèRE-PéRèS; YUGMA, 2018). Para a resolução de um problema deste tipo são necessárias as informações: número de *jobs*, número de máquinas, ordem de execução das máquinas para cada *job*, tempo de execução em cada máquina para cada *job*. Exemplos de aplicações práticas deste problema podem sem vistos em sistemas de manufatura e na indústria química e farmacêutica (CHEN; IHLOW; LEHMANN, 1999). As restrições desse problema são definidas abaixo:

- Um *job* não visita uma máquina mais de uma vez;
- Não existem restrições de precedência entre as operações de *jobs* diferentes;
- Cada máquina só pode processar um *job* por vez;
- Cada *job* só pode ser processado por apenas uma máquina de cada vez;
- Não são especificadas o tempo de preparo nem de entrega dos *jobs*.

Os objetivos ao resolver problemas de escalonamento envolvem: reduzir tempo de completude do sistema (*makespan*), reduzir tempo de ociosidade das máquinas, reduzir atraso médio das tarefas, etc. O objetivo adotado neste trabalho é minimizar o *makespan*, que representa a maior parte dos estudos na área (ÇALIş; BULKAN, 2015).

Um exemplo de configuração pode ser visto abaixo, onde a sequência de operação de cada *job* é definida seguindo os fluxos das máquinas. O *job J1*, por exemplo, deve ser executado na seguinte ordem: máquina $M1$, máquina $M2$ e, por fim, máquina $M3$. Os valores entre parênteses significam o tempo de processamento de cada *job* em cada máquina.

$$
\begin{array}{llll}
\text{J1} & \rightarrow \text{M1}(3) & \rightarrow \text{M2}(3) & \rightarrow \text{M3}(2) \\
\text{J2} & \rightarrow \text{M1}(1) & \rightarrow \text{M3}(5) & \rightarrow \text{M2}(3) \\
\text{J3} & \rightarrow \text{M2}(3) & \rightarrow \text{M1}(2) & \rightarrow \text{M3}(4)
\end{array}
$$

Figura 2.1: Exemplo de uma configuração de problema com 3 *Jobs* e 3 Máquinas

Uma solução utilizando o *makespan* como o objetivo a ser minimizado nesse exemplo é retornada pelo software LEKIN[1] como o gráfico de *Gantt* visto na figura a seguir.

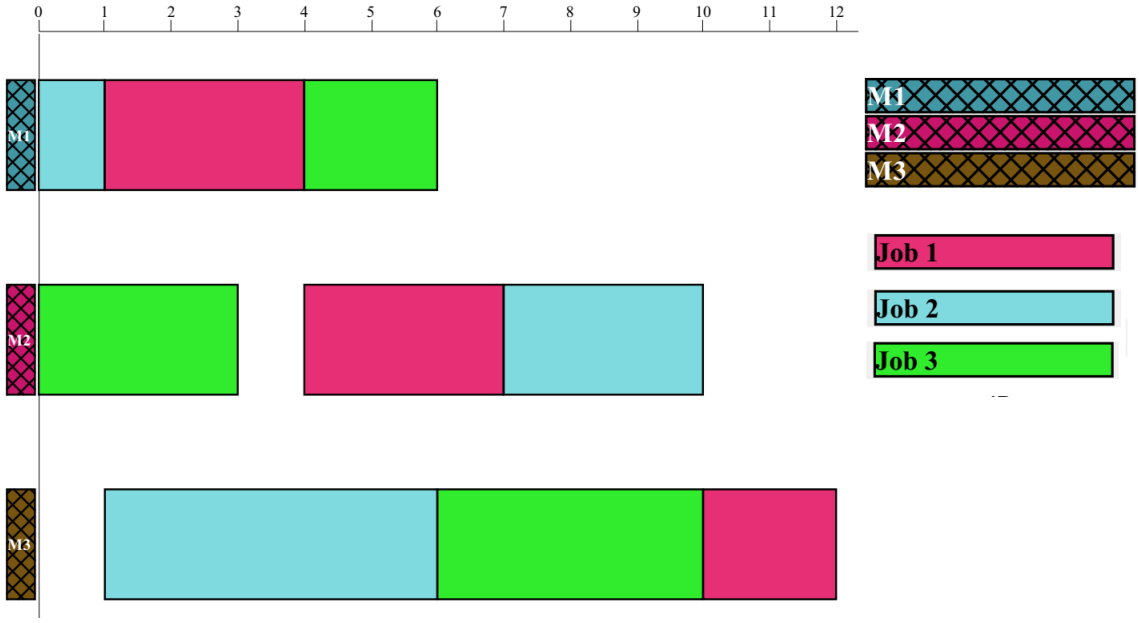


Figura 2.2: Uma solução para o exemplo da Figura 2.1 com *Makespan* = 12.

[1] http://web-static.stern.nyu.edu/om/software/lekin/

# 3 Revisão Bibliográfica

Este capítulo apresenta um compilado de trabalhos onde os temas são relacionados ao conteúdo desta monografia. São apresentados primeiramente métodos aplicados nos problemas de escalonamento *Job-Shop* (JSS), em seguida trabalhos que utilizam métodos baseadas em AG para JSS e posteriormente referências na área de Adaptação de Parâmetros.

## 3.1 Abordagens em problemas *Job-Shop*

Heurísticas e Meta-Heurísticas vem sendo utilizadas na resolução de problemas que exigem alto custo computacional para se achar uma solução ótima (SABUNCUOGLU; BAYIZ, 1999).

No ramo das Meta-Heurísticas, são destacadas as que se caracterizam pelo fator de sua inspiração ser baseada na natureza como o algoritmo Colônia de Formigas (ACO). Heinonen e Pettersson (2007) propõem uma hibridização do ACO com pós-processamento que utiliza uma Busca Local baseada no trabalho de Nowicki e Smutnicki (1996). Após a matriz de feromônios ser definida pelas formigas, um escalonamento viável é construído a partir dessa informação e a Busca Local identifica o caminho crítico do escalonamento e executa perturbações na representação da possível solução a fim de diminuir o *makespan*. Na comparação de performance com uma instância com 10 *jobs* e 10 máquinas, um AG foi utilizado e seu desempenho foi superior ao ACO quando executada o pós-processamento, mas ao verificar a matriz de feromônios do ACO antes e depois do pós-processamento não foram identificadas diferenças significantes, o que significa que o ACO não necessitou de um pós-processamento para ter um bom resultado comparado ao AG com pós-processamento.

Uma combinação entre Algoritmo Genético (AG) e *Simulated Annealing* (SA) pode ser vista em Wang e Zheng (2001). O processo consiste em gerar indivíduos através dos operadores do AG e utilizar o SA em cada indivíduo de maneira paralela. Na comparação dos resultados, foram utilizadas instâncias conhecidas da literatura (KOLISCH;

HARTMANN, 2006) e os algoritmos de forma separada. Em todos os casos de teste, a combinação das metaheurísticas (GASA) tiveram comportamento superior ao GA e o SA utilizando o mesmo orçamento computacional. O autor justifica o desempenho inferior das metaheurísticas ao fato da necessidade de explorar individualmente um maior espaço de busca, e para isso dependem de um ajuste fino de parâmetros.

Em Zhang et al. (2011) é apresentada uma abordagem híbrida que consiste em uma combinação das Meta-Heurísticas Busca Tabu (TS), SA e *Particle Swarm Optimization* (PSO). O framework desenhado neste trabalho consiste em gerar soluções iniciais pelo PSO e utilizar o SA e TS como formas exploração do espaço de busca. Foram utilizadas um conjunto de 43 instâncias de teste a performance do algoritmo proposto foi comparada com abordagens vistas nos trabalhos de Gonçalves, Mendes e Resende (2005), Xia e Wu (2006) e Lin et al. (2010) obtendo o melhor desempenho em 40 das 43 instâncias avaliadas.

Baseando-se nas heurísticas construtivas como objeto de otimização no processo evolucionário, em Ho e Tay (2005) foi utilizada uma abordagem via Programação Genética (PG) na evolução de regras compostas para o problema JSS que utiliza máquinas paralelas em cada processo. A representação da PG utiliza funções pré-definidas e características de cada *Job* com a representação do indivíduo na forma de árvore. Posteriormente, em 2013 foi apresentado em Nguyen et al. (2013) um estudo de representações em PG para o problema de escalonamento *Job-Shop* (JSS). Três representações foram propostas neste trabalho: representação por Árvore de decisão (R1), representação aritmética para geração de Regras de Despacho (R2) e uma terceira representação (R3) que combina R1 com R2. A Representação por Árvore de Decisão utiliza um indivíduo que é a derivação de uma Gramática Formal que incorpora dados como: regras de Despacho, estruturas condicionais, atributos dos *Jobs* e das máquinas. A representação R2 é semelhante à vista em Abednego e Hendratmo (2011) e um exemplo de representação pode ser visto na figura abaixo, onde RD, DD e SP significam: data de lançamento, data de entrega e a soma de todos os tempos de processamento de todos os *Jobs*, respectivamente. As melhores regras encontradas neste trabalho obtiveram performances competitivas quando comparadas aos AGs híbridos de Zhou, Cheung e Leung (2009).

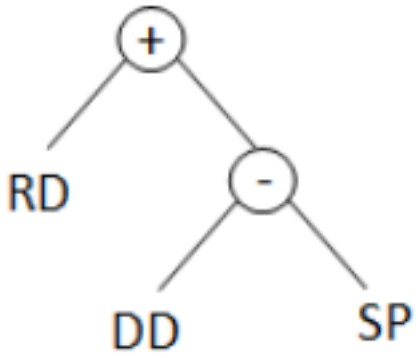


Figura 3.1: Exemplo de Representação da Regra de Despacho RD + (DD - SP). Retirado de (ABEDNEGO; HENDRATMO, 2011).

Uma abordagem via ILS (*Iterated Local Search*) para o DJSS é apresentada em (NGUYEN; ZHANG; TAN, 2015) (APRILS). A Regra de Despacho candidata é inicializada de forma semelhante a PG proposta em (HO; TAY, 2005): a partir do não-terminal inicial da Gramática são sorteados os nós da árvore de derivação respeitando as regras definidas na Gramática. A avaliação de uma Regra de Despacho é feita sobre todos os casos de teste, seguindo a função de otimalidade do sistema. O procedimento de mutação, denominado Mutação de Sub-Árvore Restrita (RSM) consiste em selecionar um nó da solução candidata e derivar novamente na Gramática, com a restrição da nova sub-árvore não exceder duas unidades de altura. Dois procedimentos de perturbação são utilizados nessa abordagem: a mutação RSM e a Extração de Sub-Árvore (SE). O procedimento SE consiste em selecionar um nó da árvore e remover a sub-árvore correspondente. A característica deste procedimento é a não-inclusão de nova informação na solução candidata. Em comparação com as abordagens vistas em Geiger e Uzsoy (2008) e Hildebrandt, Heger e Scholz-Reiter (2010) o algoritmo proposto neste trabalho se mostrou mais eficiente na descoberta de regras de despacho para os problemas considerados.

Os estudos de Çaliş e Bulkan (2015) evidenciam que para os problemas JSS, os trabalhos da literatura que utilizam o *makespan* como função objetivo representam mais de 50% dos avaliados e como estratégia de resolução dos problemas JSS os AGs, com uma representatividade por volta de 26%. A estratégia utilizada neste trabalho será semelhante, visto a importância da representatividade de trabalhos publicados na área, o que facilita comparações com técnicas que mostraram bons resultados anteriormente.

Lvshan, Dongzhi e Weiyu (2017) propõem uma combinação da metaheursística Colônia de Abelhas Artificiais (ABC) com um AG. O ABC é utilizado para gerar soluções iniciais com bastante diversidade e o operador de mutação do AG é utilizado para fazer buscas locais nas soluções geradas pelo ABC. O operador de mutação do AG seleciona duas sequências da solução e inverte sua ordem. Para a comparação de desempenho desta proposta, foram criadas instâncias artificiais e as versões separadas do ABC e do AG foram consideradas nos testes. Os experimentos mostraram que a abordagem híbrida teve desempenho superior aos algoritmos isolados, comprovando a tese do autor que as duas abordagens se complementam.

## 3.2 Algoritmos Genéticos em Problemas *Job-Shop*

Em Bierwirth (1995) é proposto um tipo de operador de recombinação denominado GOX (*Generalizated Order-Crossver*), que é baseado no operador OX (*Order-Crossover*). O indivíduo é representado como uma fila de operações à serem executadas. Com a utilização de um escalonador para decodificar o indivíduo, essa representação é vantajosa pois a cada operação de mutação ou recombinação aplicada, não é gerada nenhuma solução inviável. Utilizando os problemas teste de Applegate e Cook (1991), o operador obteve resultados satisfatórios quando aplicado em instâncias de tamanho grande, levando em consideração que a proposta não utilizou ajuste de parâmetros.

Com uma combinação de um AG e um procedimento gerador de agendamentos, Gonçalves, Mendes e Resende (2005) utiliza uma representação baseada em chaves aleatórias (*Random Keys*) (BEAN, 1994). O procedimento gerador de agendamentos é dividido em 3 partes: decodificação das prioridades e dos tempos de atraso, geração do agendamento e Busca Local. As buscas locais foram baseadas nos trabalhos de Roy e Sussmann (1964) e Nowicki e Smutnicki (1996). O algoritmo obteve soluções ótimas e próximas do ótimo nas 43 instâncias de teste (FISHER, 1963; LAWRENCE, 1984) quando comparado com algoritmos propostos em outros trabalhos utilizando o mesmo conjunto de problemas teste.

Um modelo de ilhas aplicado a um AG pode ser visto em Kurdi (2015). A divisão da população de indivíduos em ilhas faz com que cada ilha explore diferentes espaços de

busca utilizando seu próprio processo evolutivo. Os melhores indivíduos de cada ilha são selecionados para a aplicação de uma busca Tabu e os piores sofrem a aplicação combinada de três operadores de mutação para explorar outras vizinhanças. Para a análise de desempenho desta abordagem foram utilizados 76 problemas bem conhecidos da literatura e um conjunto de 15 algoritmos de trabalhos anteriores e a proposta deste trabalho é capaz de encontrar a melhor solução conhecida em 71% dos casos.

Utilizando uma versão adaptativa de um AG para o controle de parâmetros, Ferreira e Bernardino (2017) utilizam um esquema de *feedback* que retorna a qualidade de um operador ou parâmetro utilizado para inferir qual será a próxima escolha do algoritmo. A versão adaptativa proposta necessita da configuração somente de dois hiper-parâmetros, enquanto o AG utilizado para comparação de resultados tem dois operadores (recombinação e mutação) e três parâmetros (taxas de aplicação de mutação, recombinação e busca local). Os problemas de teste foram gerados segundo Taillard (1993) e adaptados para a versão flexível do JSS e a versão adaptativa do AG obteve desempenho superior considerando todos os problemas avaliados.

## 3.3 Adaptação de Parâmetros

Com o objetivo de variar as taxas de mutação e recombinação de um AG, Srinivas e Patnaik (1994) propõe um esquema adaptativo que considera a aptidão do indivíduo e a aptidão média da população para definir quais valores serão usados. A diferença entre esses dois valores mostra se um sistema convergiu, quando for pequena, ou se ele está com soluções dispersas no espaço de busca. Quando comparado ao desempenho de um AG simples em um conjunto de problemas de Caixeiro Viajante (TSP), a versão adaptativa conseguiu encontrar soluções ótimas enquanto a versão simples do AG não obteve nenhum desempenho ótimo considerando todos os problemas utilizados.

Para uma seleção adequada dos operadores envolvidos no processo de busca de um AGA é necessário um método de atribuição de crédito que leve em consideração a eficiência de um certo operador quando aplicado (COSTA et al., 2008). Fialho, Schoenauer e Sebag (2010) propõe um método de atribuição de crédito que utiliza uma janela de tempo de tamanho $W$ onde cada posição representa o melhor operador aplicado naquele instante. O

crédito dado a um operador $i$ em um tempo $t$ é dado pela equação abaixo, que define *Sum of Ranks* (SR), onde $D \in [0, 1]$ é o fator de distorção da distribuição ($D = 1$ representa uma linearidade no decaimento) e $r$ é a posição na janela $W$.

$$SR_{i,t} = \frac{\sum_{opr=i} D^r (W - r)}{\sum_{r=1}^{W} (W - r)} \tag{3.1}$$

Uma metodologia baseada na medida de qualidade de aplicações anteriores de um operador pode ser vista em Aleti e Moser (2011). O *Predictive Parameter Control* (PPC) gera uma série temporal de acordo com a equação que segue, onde $u_i$ é a frequência do parâmetro $i$ e $u_i^s$ é a frequência de sucesso do parâmetro $i$, que é contabilizada quando a aptidão do indivíduo gerado é maior que a aptidão média da população. A partir dessa informação, as taxas de sucesso das duas últimas gerações são utilizadas para extrapolar uma previsão para a próxima geração. Seu desempenho foi superior dos métodos de controle de parâmetros considerados mais bem sucedidos para instâncias do problema *Royal Road* (JONES, 1994).

$$Taxa\ de\ sucesso = \frac{u_i^s}{u_i} \tag{3.2}$$

Ferreira (2018) analisa o desempenho da técnica de seleção de operadores *Adaptive Pursuit* (AP) (THIERENS, 2005) combinada com a técnica de atribuição de crédito por valor Extremo (EX) (FIALHO et al., 2008) para o JSS flexível, onde podem existir mais máquinas por estágio. Os parâmetros considerados na versão adaptativa do AG considerado são os operadores de movimento e suas taxas de aplicação e a taxa de aplicação da Busca Local (RUIZ; STüTZLE, 2007). Os problemas de teste foram adaptados de Taillard (1993) e o critério de parada utilizado para os algoritmos foi o número de avaliações de função objetivo. Considerando todos os problemas, o AP com EX aliado ao AG obteve desempenho superior ao melhor AG simples, que varia os parâmetros escolhidos pelo usuário.

# 4 Algoritmos Genéticos em Problemas Job-Shop

Dentre as Meta-Heurísticas utilizadas para a resolução de problemas de escalonamento destacam-se os Algoritmos Evolucionários (CHENG; GEN; TSUJIMURA, 1996). Baseando-se na evolução natural com o intuito de aprimorar um conjunto de indivíduos segundo uma característica (aptidão), os AGs são comumente utilizados para solucionar problemas complexos na área de otimização (HOLLAND, 1975).

Baseado em uma população de indivíduos, os AGs utilizam operadores para modificar características da solução candidata a fim de melhorar suas aptidões ao longo do processo evolucionário. Seu processo se inicia ao criar uma população inicial, avaliar os indivíduos dessa população segundo sua aptidão e aplicar um processo iterativo que consiste em: selecionar os indivíduos para a aplicação dos operadores de movimento (mutação e recombinação), aplicar uma busca local, avaliar os novos indivíduos e decidir se eles entram na população ou são descartados (ASADZADEH, 2015). O algoritmo abaixo exemplifica o fluxo do AG com busca local utilizado neste trabalho e todos os processos serão explicados nos próximos parágrafos.

A representação que será utilizada neste trabalho é a baseada em Tarefas (HOLSAPPLE et al., 1993). O indivíduo será uma permutação de tarefas com seu tamanho sendo igual à quantidade de *jobs* do problema. A figura abaixo exemplifica a codificação de um indivíduo.

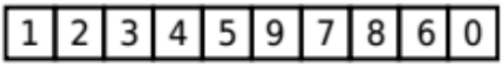


Figura 4.1: Exemplo de codificação de um indivíduo com 10 *jobs*

Para avaliar um indivíduo com esse tipo de codificação é necessário um algoritmo que gera o escalonamento e calcula o seu *makespan*. As tarefas sempre serão alocadas quando não estiverem sendo executadas em outras máquinas e sempre em máquinas oci-

**Algoritmo 1:** Pseudo-código AG.

```
1  Criar aleatoriamente a população inicial P;
2  Avaliar todos os indivíduos de P;
3  enquanto critério de parada não for satisfeito faça
4  |  Seleção;
5  |  se probabilidade de cruzamento então
6  |  |  Cruzamento;
7  |  fim se
8  |  se probabilidade de mutação então
9  |  |  Mutação;
10 |  fim se
11 |  se probabilidade de busca local então
12 |  |  Busca Local;
13 |  fim se
14 |  Avaliação;
15 |  Substituição da População Antiga;
16 fim enquanto
17 retorna melhor indivíduo encontrado;
```

osas. A vantagem de utilizar este tipo de representação é que o processo de decodificação da solução gera somente soluções viáveis. O processo de avaliação de um indivíduo é descrito abaixo.

**Algoritmo 2:** Pseudo-código Avaliação indivíduo.

```
1  enquanto Todas as tarefas não forem executadas em todas as máquinas
   faça
2  |  Selecionar a tarefa t_i a ser executada na lista;
3  |  Selecionar próxima máquina m_i na ordem de execução da tarefa t_i;
4  |  se a tarefa estiver sendo executada em outra máquina então
5  |  |  Adicionar ao tempo de início da tarefa t_i o instante que a outra
   |  |  máquina m_{i-1} terminar de executá-la;
6  |  fim se
7  |  Adicionar ao tempo de início da tarefa t_i o instante que a máquina m_i
   |  estiver livre;
8  |  Alocar a tarefa t_i à máquina m_i;
9  fim enquanto
10 retorna tempo de completude do sistema;
```

Selecionar indivíduos para a aplicação dos operadores de mutação e recombinação requer um critério bem definido. Neste trabalho, o critério utilizado será o Torneio Binário: dois indivíduos são selecionados aleatoriamente e o de melhor aptidão passa para a fase de aplicação dos operadores. No caso dos operadores de recombinação, onde são necessários dois indivíduos para a operação, outro torneio será realizado para selecionar a quantidade

de indivíduos necessários (EIBEN; SMITH et al., 2003).

Para a fase de recombinação foram escolhidos quatro operadores baseados em permutações de elementos: *Order-Preserving one-point crossover*, *Linear-Order crossover*, *Partially mapped crossover* e *Order-based Crossover* (WERNER, 2011). A descrição de seus comportamentos seguem na sequência:

- *Order-Preserving one-point crossover* (OPX): um ponto de corte aleatoriamente é selecionado de um pai e é copiado para o filho, a informação restante é copiada do outro pai preservando a ordem e sem repetir elementos.

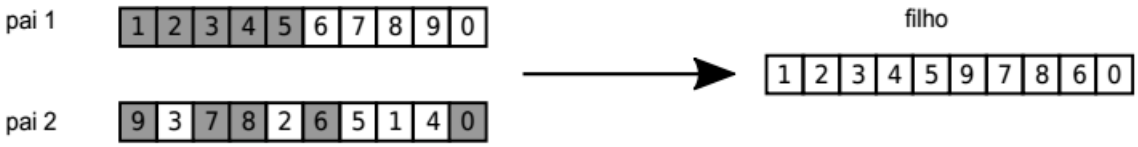


Figura 4.2: Exemplo OPX.

- *Linear-Order crossover* (LOX): semelhante ao OPX, o LOX utiliza dois pontos de corte e preserva a ordem da informação dos dois pais.

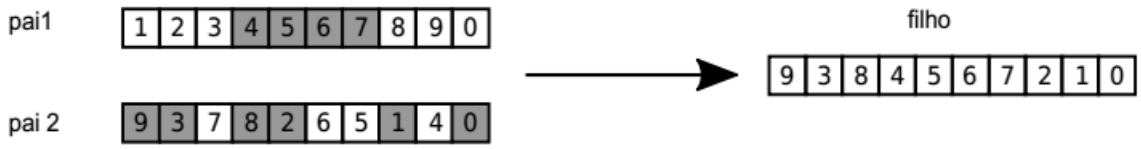


Figura 4.3: Exemplo LOX.

- *Partially mapped crossover* (PMX): dois pontos de corte são selecionados em um pai e copiados para o filho, o restante eh copiado do outro pai nas posições que não estão no corte. As duplicidades são resolvidas fazendo mapeamentos de valor e posição entre os pais até que seja encontrado um valor que não tenha sido copiado ainda para o filho.

- *Order-based Crossover* (OBX): seleciona-se um subconjunto de *jobs* de um pai e eles são copiados no filho, respeitando as posições do segundo pai e a ordem do primeiro

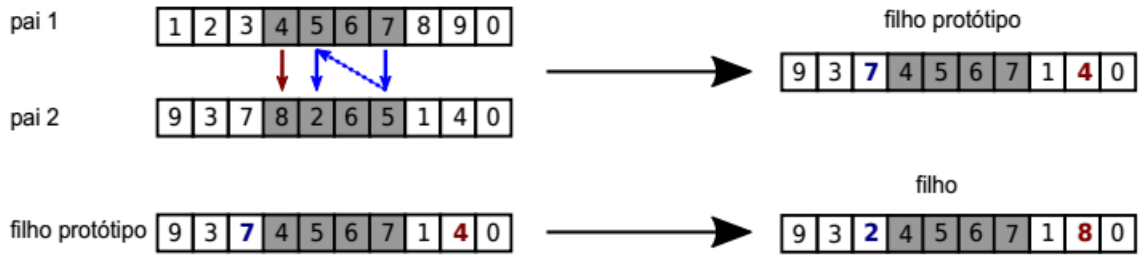


Figura 4.4: Exemplo PMX.

pai. O restante dos *jobs* são copiados do segundo pai para o filho respeitando sua ordem relativa.

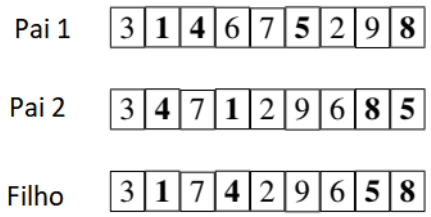


Figura 4.5: Exemplo OBX.

Foram três operadores de mutação escolhidos para serem utilizados neste trabalho (*Shift Mutation*, *Pairwise Interchange Neighborhood Mutation* e *Inverse Mutation*) e seus funcionamentos são descritos abaixo (BÄCK; FOGEL; MICHALEWICZ, 2018):

- *Shift Mutation* (SM): um elemento e um ponto de inserção são selecionado aleatoriamente e esse elemento é inserido na posição fazendo um deslocamento nos demais elementos.

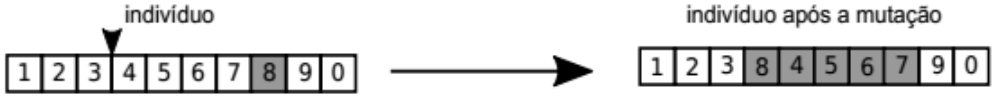


Figura 4.6: Exemplo SM.

- *Pairwise Interchange Neighborhood Mutation* (PM): dois elementos são selecionados e são trocados de posição.


Figura 4.7: Exemplo PM.

- *Inverse Mutation* (IM): dois pontos de corte são selecionados e a ordem dos elementos contidos nesses pontos é trocada.

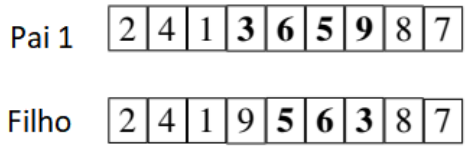


Figura 4.8: Exemplo IM.

Após a aplicação dos operadores de movimento novos indivíduos serão gerados e o esquema de substituição da população utilizado neste trabalho é o *Steady-state*: os $n$ indivíduos gerados a partir dos operadores de movimento serão adicionados à nova população. A população novamente é ordenada segundo a aptidão e os $n$ de pior aptidão serão descartados, mantendo a população sempre com tamanho constante (SYSWERDA, 1991).

Com o objetivo de iniciar a população com indivíduos de boa qualidade sem fazer uso de grande esforço computacional, o uso de heurísticas construtivas surgem como opção neste contexto. A heurística NEH (NAWAZ; ENSCORE; HAM, 1983) foi inicialmente proposta para problemas *Flow-Shop* com a representação baseada em tarefas. Sendo a mesma representação utilizada neste trabalho, a heurística NEH pode ser utilizada para o problema *Job-Shop*. Seu funcionamento é explicado no algoritmo que segue.

A solução começa com as duas tarefas de maior duração. A permutação que gerar menor *makespan* será adotada para a inserção das próximas tarefas. Cada tarefa que ainda não foi inserida na solução será avaliada em todas as permutações possíveis respeitando a ordem relativa das tarefas já inseridas. A posição de inserção que gerar menor *makespan* será a fixada para a tarefa corrente. Este processo termina quando todas as tarefas forem inseridas na solução.

**Algoritmo 3:** Pseudo-código NEH.

1 **Entrada**: Lista de Tarefas ;
2 Calcular a duração total das operações para cada tarefa;
3 Ordenar de forma decrescente pela duração total;
4 Selecionar as duas primeiras tarefas e escolher a permutação que gera menor *makespan*;
5 **para** *cada tarefa ainda não selecionada* **faça**
6 | Inserir esta tarefa na sequência em todas as posições, mantendo a ordem relativa das tarefas que já foram inseridas;
7 | Escolher a ordem que gera menor makespan;
8 **fim para**

A Busca Local (RUIZ; STüTZLE, 2007) é semelhante à Heurística Construtiva: inicialmente, são selecionadas de forma aleatória um conjunto de tarefas para serem retiradas da solução. Sua reinserção segue o passo 5 do Algoritmo 3 até que não existam tarefas a serem reinseridas. O número de tarefas removidas da solução segue o parâmetro taxa de destruição (txDes), que é proporcional ao tamanho do indivíduo.

# 5 Algoritmos Genéticos Adaptativos

Os estudos de Karafotias, Hoogendoorn e Eiben (2015) salientam a importância de se definir bons parâmetros em Algoritmos Evolucionários para que um bom desempenho seja observado. É evidenciado também que os parâmetros que se aproximam do ótimo podem mudar durante o processo evolutivo. Com o intuito de evitar um esforço computacional na escolha dos parâmetros antes de execução dos algoritmos, foram propostos abordagens de ajuste de parâmetros durante a execução dos algoritmos.

Exitem basicamente três esquemas de controle de parâmetros durante a execução de um algoritmo: controle determinístico, controle por adaptação e controle auto-adaptação (EIBEN; HINTERDING; MICHALEWICZ, 1999). O esquema determinístico altera os parâmetros seguindo uma regra que é definida antes da execução do algoritmo. Já o controle por adaptação faz uso do *feedback* da aplicação de um operador para determinar a magnitude da alteração nos valores dos parâmetros do algoritmo. Por fim, no controle por auto-adaptação o indivíduo carrega, em sua codificação, os parâmetros das aplicações dos operadores.

A técnica de ajuste adotada neste trabalho é a de controle por adaptação. Os ajustes considerados são baseados nas técnicas de seleção de operadores e na atribuição de crédito após a sua aplicação no processo de busca. Os parâmetros que farão uso desta técnica são: operadores de movimento (recombinação e mutação), taxas de aplicação dos operadores de movimento e taxas de aplicação da busca local. Logo, a estrutura dos Algoritmos Genéticos Adaptativos pode ser descrita abaixo.

Os passos que dizem respeito ao AG simples serão os mesmos definidos no capítulo anterior. Dois métodos de seleção de operadores foram escolhidos para este trabalho: o *Adaptive Pursuit* (AP)(THIERENS, 2005) e o *Ranked Multi-Armed Bandit* (RMAB)(FIALHO; SCHOENAUER; SEBAG, 2010). O método de recompensa por valor extremo (EX) aliado ao AP, como visto em Ferreira e Bernardino (2017), demonstrou resultados superiores ao AG comum e será adotado neste trabalho. O método *Sum of Ranks* (SR) aliado ao RMAB demonstrou, dentre as técnicas adaptativas, o melhor desempenho na resolução

**Algoritmo 4:** Pseudo-código AGA

```
Criar aleatoriamente a população inicial P;
Avaliar todos os indivíduos de P;
enquanto critério de parada não for satisfeito faça
    Seleção dos Indivíduos;
    Escolha dos Operadores;
    Escolha das taxas de aplicação dos Operadores;
    Escolha da taxa de aplicação da Busca Local;
    se probabilidade de operador então
        Aplicação Operador Movimento;
    fim se
    se probabilidade de busca local então
        Aplicação Busca Local;
    fim se
    Avaliação dos Indivíduos;
    Recompensar Operadores e Taxas de Aplicação;
    Recompensar Taxa de aplicação da Busca Local;
    Substituição da População Antiga;
fim enquanto
retorna melhor indivíduo encontrado;
```

dos problemas JSS Flexíveis em Ferreira (2018).

## 5.1 *Adaptive Pursuit* com Valor Extremo (APEX)

Proposta por Thierens (2005), a técnica de seleção de operadores *Adaptive Persiut* sorteia de maneira uniforme na distribuição de probabilidades um operador para ser aplicado. No início da execução desta técnica, todas as probabilidades são definidas com o mesmo valor e ao longo do processo evolutivo elas vão sofrendo alterações de acordo com a qualidade dos operadores aplicados. Os operadores que demonstrarem melhor qualidade terão mais chances de ser escolhidos. A qualidade dos operadores é relacionada à recompensa atribuída a ele após sua aplicação. A inicialização e o funcionamento da técnica são descritos abaixo.

**Algoritmo 5:** Inicialização Adaptive Pursuit.

```
Entrada: K, P_min, α, β;
P_max ← 1 − (K − 1)P_min;
para i ← até K faça
    P_i(T_0) ← 1/K;
    Q_i(T_0) ← 1, 0;
fim para
```

**Algoritmo 6:** Pseudo-código Adaptive Pursuit.

```
1 i ← SeleçãoUniforme(P);
2 R_i ← Recompensa(i);
3 Q(t+1) = Q_i + α[R_i(t) − Q_i(t)];
4 i* ← argmax(Q_i(t+1));
5 P_i*(t+1) ← P_i*(t) + β[P_max − P_i*(t)] ;
6 para i ← até K, com i ≠ i* faça
7 |   P_i(t+1) ← P_i(t) + β[P_min − P_i(t)] ;
8 fim para
```

A qualidade ($Q_i$) de um operador $i$ é atualizada, sempre que ele for sorteado, a uma taxa $\alpha$ da diferença da recompensa atribuída à esse operador ($R_i$) pela qualidade da última aplicação do mesmo. O operador de melhor qualidade ($i^*$) terá sua probabilidade ($P_{i^*}$) alterada em um fator $\beta$ da diferença de $P_{max}$ definida no início do algoritmo pela probabilidade da última aplicação deste operador. Os demais operadores terão a atualização de suas probabilidades realizadas da mesma maneira, somente trocando $P_{max}$ por $P_{min}$. $K$ é a quantidade de operadores disponíveis e $P_{min}$ é definido pelo usuário.

A técnica de recompensa ($R_i$) que será utilizada para este algoritmo é chamada de Valor Extremo (EX)

$$R\left(i\right) = argmax\left\{RI\left(t_i\right), i = 1 \cdots W\right\} \tag{5.1}$$

e foi proposta em Fialho et al. (2008). É utilizada uma janela temporal que armazena as $W$ recompensas instantâneas ($RI$) mais recentes das aplicações dos operadores para definir qual será o valor de recompensa no tempo atual de acordo com a equação que segue. A $RI$ utilizada neste trabalho é o maior valor da diferença da aptidão do indivíduo pai pela aptidão do indivíduo filho, gerado pela aplicação de um operador, considerando todos os filhos gerados.

$$RI = max(0, \textbf{Aptidão}(pai) - \textbf{Aptidão}(filho)) \tag{5.2}$$

## 5.2 *Rank-based Multi-Armed Bandit* com *Sum of Ranks* (RMAB)

Baseado no problema clássico da literatura *Multi-Armed Bandit* (MAB)(GITTINS; GLAZEBROOK; WEBER, 2011), uma adaptação para o problema de seleção de operadores e taxas de ocorrências pode ser vista em Fialho, Schoenauer e Sebag (2010). A seleção dos operadores é feita de maneira uniforme nos não-operados até que todos os operadores tenham sido escolhidos. A partir disso, os operadores são escolhidos com base no seu valor de recompensa $\hat{q}_i$ associada a um fator ($C$) da quantidade de vezes ($n_i$) que este operador aparece em um intervalo de tempo fixo ($W$). Diferente do AP, a janela temporal do RMAB considera as aplicações de todos os operadores.

**Algoritmo 7:** Inicialização Rank-based Multi-Armed Bandit.

```
1 Entrada: K, C, D, W;
2 para i ← até K faça
3 |   n_i ← 0;
4 |   q̂_i ← 0,0;
5 fim para
```

**Algoritmo 8:** Pseudo-código Rank-based Multi-Armed Bandit.

```
1  se existe algum operador não aplicado então
2  |   op ← Seleção Uniforme(Não Operados);
3  fim se
4  senão
5  |   op ← argmax_{i=1...K} ( q̂_{i,t} + C sqrt( 2log Σ_K n_{k,t} / n_{i,t} ) );
6  fim se
7  para i ← até K faça
8  |   q̂_i ← Recompensa(i) ;
9  |   n_i ← Quantidade(JanelaTemporal, i) ;
10 fim para
```

No trabalho de Fialho, Schoenauer e Sebag (2010) é proposto um método de atribuição de crédito que utiliza a Janela Temporal de tamanho $W$ onde cada posição representa o melhor operador aplicado naquele instante. O crédito dado a um operador $i$ em um tempo $t$ é dado pela equação abaixo, denominada *Sum of Ranks* (SR),

$$SR_{i,t} = \frac{\sum_{opr=i} D^r (W - r)}{\sum_{r=1}^{W} (W - r)} \tag{5.3}$$

onde $D \in [0, 1]$ define o fator de distorção da distribuição ($D = 1$ representa uma linearidade no decaimento) e $r$ é a posição na janela $W$. O SR permite que operadores que tenham histórico antigo de aplicação mas que não foram selecionados recentemente ainda tenham chance de ser escolhidos.

# 6 Experimentos Computacionais

Para comparar o desempenho das abordagens utilizadas neste trabalho, foram propostos dois tipos de experimentos com duas fases cada um: fase de configuração e fase de teste. A fase de configuração utiliza um subconjunto de instâncias para configurar os parâmetros dos AGs e os hiper-parâmetros dos AGAs manualmente. A fase de testes executa os algoritmos utilizando as instâncias que não foram avaliadas com os melhores parâmetros encontrados na fase de configuração. Os resultados encontrados nas fases de teste serão os considerados ao comparar o desempenho dos algoritmos.

A fase de configuração do primeiro experimento utiliza 20% das instâncias de cada problema enquanto a do segundo experimento utiliza todas as instâncias do problema de menor tamanho. O objetivo do primeiro experimento é encontrar o melhor algoritmo que resolve um problema específico. O segundo experimento serve para detectar qual algoritmo tem melhor capacidade de generalização ao resolver problemas de vários tipos.

Os problemas de teste foram gerados segundo Taillard (1993) e podem ser encontrados no site do autor[2]. Para compor o conjunto de instâncias de teste foram escolhidas 6 configurações de problemas ($N$ *jobs*, $M$ máquinas) com 10 instâncias cada, totalizando 60 instâncias-teste com as características abaixo:

- 15 *jobs* e 15 máquinas
- 20 *jobs* e 15 máquinas
- 20 *jobs* e 20 máquinas
- 30 *jobs* e 15 máquinas
- 30 *jobs* e 20 máquinas
- 50 *jobs* e 15 máquinas

Os algoritmos avaliados nestes experimentos serão: um AG comum, que utilizará um operador de recombinação e um de mutação; dois AGAs, RMAB e APEX e dois

[2]http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html

AGs baseados nas distribuição de frequências dos AGAs que sorteiam uniformemente um operador (recombinação ou mutação), uma taxa de aplicação do operador e uma taxa de ocorrência da Busca Local (PROB_APEX E PROB_RMAB). As frequências de utilização das taxas e dos operadores nos AGAs indicam quais operadores são mais utilizados após processo de busca. Se os AGs que utilizam essas frequências obtiverem bons resultados, significa que as probabilidades de seleção dos operadores não são afetadas pelo instante da busca.

O critério de parada da execução dos algoritmos é o número de avaliações de função objetivo, dado pela fórmula $(800+ 10N)N^2$ (ARROYO; PEREIRA, 2011). Cada algoritmo é executado 5 vezes para cada instância e o *makespan* médio das execuções é considerado. A Heurística NEH será utilizada para gerar 20% dos indivíduos na fase de criação da população inicial dos algoritmos.

Na fase de configuração de cada experimento são escolhidos valores de parâmetros nas tabelas abaixo para os AGs e os hiper-parâmetros dos AGAs. O tamanho da população dos AGAs e dos PROBs seguirão o definido pela configuração do AG no problema correspondente.

Tabela 6.1: Parâmetros utilizados na fase de Configuração dos AGs.

| **Parâmetros** | **Valores** |
|---|---|
| Tamanho População | 50 - 76 - 100 |
| Operador recombinação | LOX- OPX - OBX - PMX |
| Operador Mutação | SM - IM - IN |
| Taxa recombinação | 0,7 - 0,8 - 0,9 |
| Taxa Mutação | 0,3 - 0,4 - 0,5 |
| Taxa de Busca Local | 0,015 - 0,030 - 0,045 |
| Parâmetro Busca Local (txDes) | 0,10 |

Para visualizar e interpretar os resultados serão utilizados os Perfis de Desempenho (PD), que serão explicados na Seção 6.1. A área normalizada abaixo das curvas de PD servirá como medida de robustez dos algoritmos avaliados. As distribuições de frequência de utilização dos operadores, das taxas de utilização dos operadores e das taxas de utilização da busca local serão também avaliadas e expostas de forma gráfica. Para a verificação estatística dos resultados será utilizado o método de Kruskal e Wallis (1952), que detecta se há diferença estatística entre os resultados retornados pelos algoritmos analisando se o p-valor retornado pelo método for menor que $0,05$.

Tabela 6.2: Hiper-Parâmetros utilizados na fase de Configuração dos AGAs.

| **Parâmetros** | **Escolha** | **Valores** |
|---|---|---|
| Operadores de Movimento | AGAs | LOX- OPX - OBX - PMX SM - IM - IN |
| Taxas de Aplicação | AGAs | 0,3 - 0,4 - 0,5 - 0,7 - 0,8 - 0,9 |
| $\alpha$ | Usuário (APEX) | 0,2 - 0,5 - 0,8 |
| $\beta$ | Usuário (APEX) | 0,2 - 0,5 - 0,8 |
| $P_{min}$ | Usuário (APEX) | 0,05 |
| $D$ | Usuário (RMABSR) | 0,3 - 0,6 - 1,0 |
| $C$ | Usuário (RMABSR) | 0,1 - 1 - 10 |
| Taxa Busca Local | AGAs | 0,015 - 0,030 - 0,045 |
| Parâmetro Busca Local (txDes) | Usuário | 0,1 |

## 6.1 Perfis de Desempenho

Proposto por Dolan e Moré (2002), os PDs representam de forma gráfica a quantidade de problemas que um algoritmo resolve com relação a melhor solução encontrada para o problema. Seja $P$ um conjunto de problemas, com $I_p$ sendo o conjunto de instâncias para cada problema e um conjunto $Al$ de algoritmos $al$. A razão de desempenho é definida por:

$$Rd(al, I_p) = \frac{C_{max}(al, I_p)}{min\left\{C_{max}(al, I_p) : al \in Al\right\}} \tag{6.1}$$

onde $C_{max}$ é o *makespan* médio das execuções independentes de cada algoritmo.

Logo, os Perfis de Desempenho de um algoritmo $al$ são definidos na equação abaixo, onde $\rho_{al}$ é a fração de problemas resolvidos dentro do fator $\tau$ da melhor solução encontrada.

$$\rho_{al}\left(\tau\right) = \frac{1}{|I_p|}\left|\left\{i \in I_p : Rd(al, I_p) \leq \tau\right\}\right| \tag{6.2}$$

Um exemplo de Perfil de Desempenho retirado de (FERREIRA, 2018) mostra a comparação entre os algoritmos utilizados no trabalho.

Os valores de $\tau$ indicam o quanto as soluções se distanciam do melhor valor conhecido. O $\rho(\tau)$ representa os valores acumulados de problemas que são resolvidos até um fator $\tau$ da melhor solução conhecida. A medida de comparação considerada é a área abaixo da curva gerada pelo PD: o algoritmo que tem a maior área, resolve mais problemas se distanciando menos do melhor valor conhecido para aquele problema.

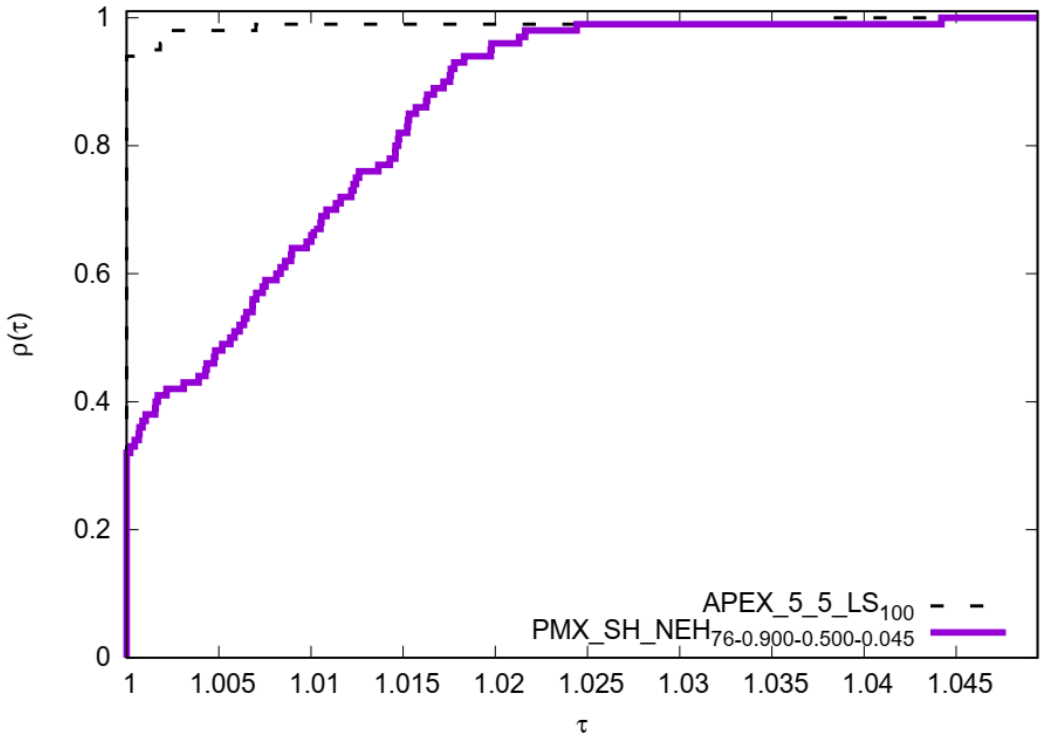


Figura 6.1: Exemplo PD (FERREIRA, 2018).

## 6.2 Experimento 1

Para avaliar o desempenho dos algoritmos em cada classe de problema buscando o que melhor se comporta em cada classe, este experimento em sua fase de configuração utiliza 20% das instâncias de cada problema (2 instâncias) para definir os parâmetros (AG) e os hiper-parâmetros (AGA) que serão utilizados na fase de testes, onde serão executados e comparados os desempenhos para as instâncias restantes (8 instâncias).

### 6.2.1 Fase de configuração

Após a execução dos algoritmos propostos nos subconjuntos de instâncias de cada configuração, os melhores parâmetros encontrados estão dispostos nas tabelas abaixo.

A partir da definição destes parâmetros, a fase de testes, onde serão comparados os desempenhos de todos os algoritmos propostos, será detalhada em cada classe de problema, e no fim, todos os problemas serão avaliados.

Tabela 6.3: Parâmetros dos AGs escolhidos para cada configuração de problema no Experimento 1.

| Problema | Op Cross | Op Mut | Tx Cross | Tx Mut | TPop | TsLS |
|---|---|---|---|---|---|---|
| **15x15** | OPX | SM | 0,7 | 0,3 | 50 | 0,045 |
| **20x15** | OPX | IM | 0,7 | 0,3 | 50 | 0,045 |
| **20x20** | OPX | IN | 0,7 | 0,3 | 50 | 0,045 |
| **30x15** | OPX | IM | 0,7 | 0,3 | 50 | 0,045 |
| **30x20** | OPX | SM | 0,7 | 0,3 | 50 | 0,045 |
| **50x15** | OPX | SM | 0,7 | 0,3 | 50 | 0,045 |

Tabela 6.4: Hiper-Parâmetros dos AGAs escolhidos para cada configuração de problema no Experimento 1.

| | **APEX** | | **RMAB** | |
|---|---|---|---|---|
| | $\alpha$ | $\beta$ | $C$ | $D$ |
| **15x15** | 0,5 | 0,8 | 0,1 | 1,0 |
| **20x15** | 0,5 | 0,5 | 0,1 | 1,0 |
| **20x20** | 0,5 | 0,5 | 0,1 | 1,0 |
| **30x15** | 0,2 | 0,5 | 0,1 | 1,0 |
| **30x20** | 0,8 | 0,8 | 0,1 | 1,0 |
| **50x15** | 0,2 | 0,5 | 0,1 | 1,0 |

### 6.2.2 15 máquinas e 15 *jobs*

Nesta classe de problema, o PROB_RMAB apresenta o segundo melhor desempenho relativo e fica abaixo somente do melhor AG (OPX-SM) configurado. As áreas normalizadas dos algoritmos APEX, PROB_APEX e RMAB tiveram valores sem diferença significativa.

Ao verificar as frequências relativas da utilização dos operadores, o RMAB prioriza os operadores de mutação, sendo o SM o mais utilizado, enquanto o APEX, em seu processo de busca, utiliza todos os operadores com frequências parecidas, sendo o IM o que mais acumula utilizações neste problema.

Para o caso das taxas de utilização dos operadores, o RMAB, em sua maioria, utiliza taxas maiores (acima de 0,5), enquanto o APEX distribui melhor o comportamento em relação a escolha das taxas.

No caso da taxa de utilização da Busca Local, os dois AGAs propostos ajustaram o seu processo de busca para, em mais de 70% dos casos, utilizar a maior taxa considerada neste trabalho (0,045).

O p-valor do teste de Kruskal-Wallis é superior a 0,95, concluindo-se que não

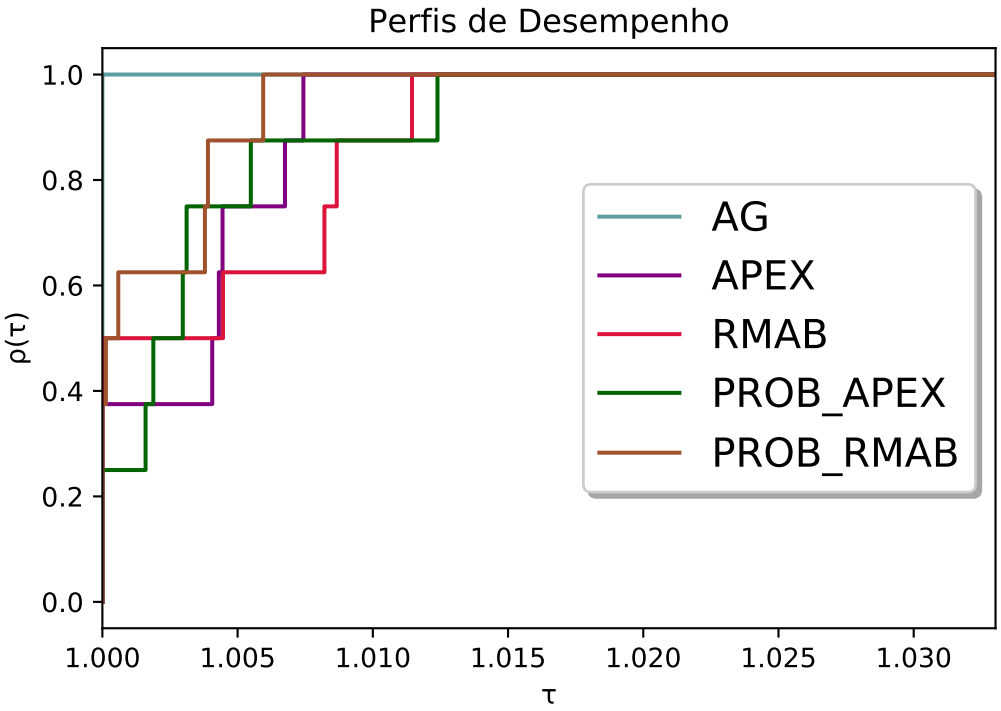


Figura 6.2: Perfis de Desempenho para o problema 15_15.

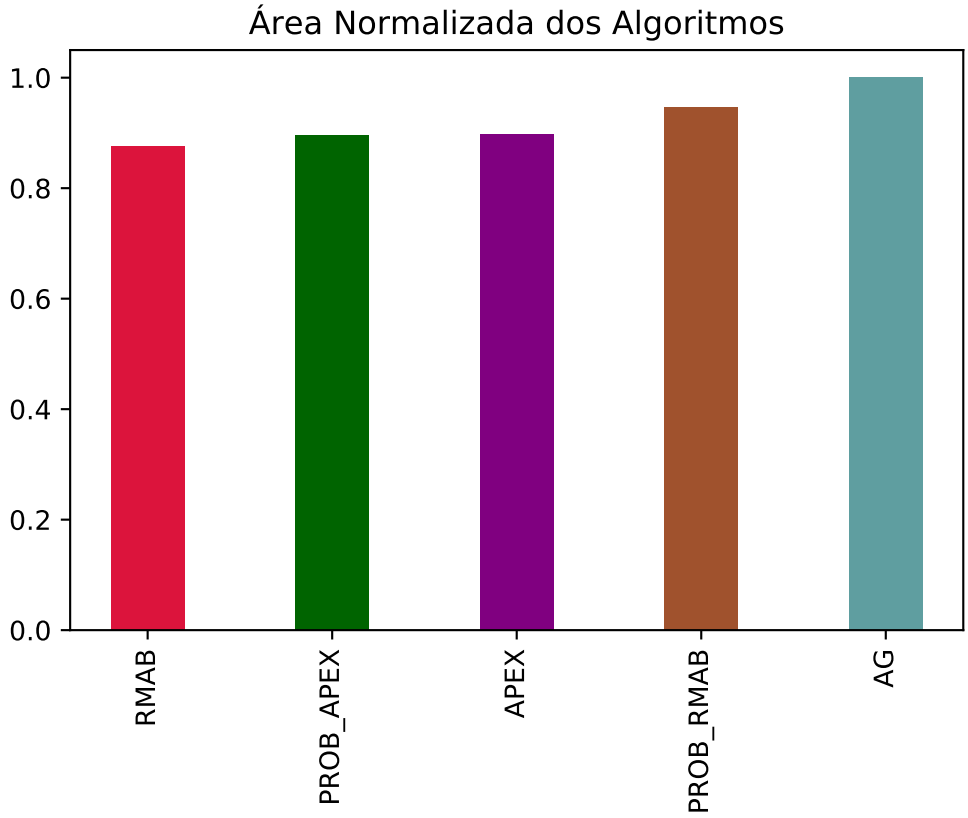


Figura 6.3: Área Normalizada dos PDs para o problema 15_15.

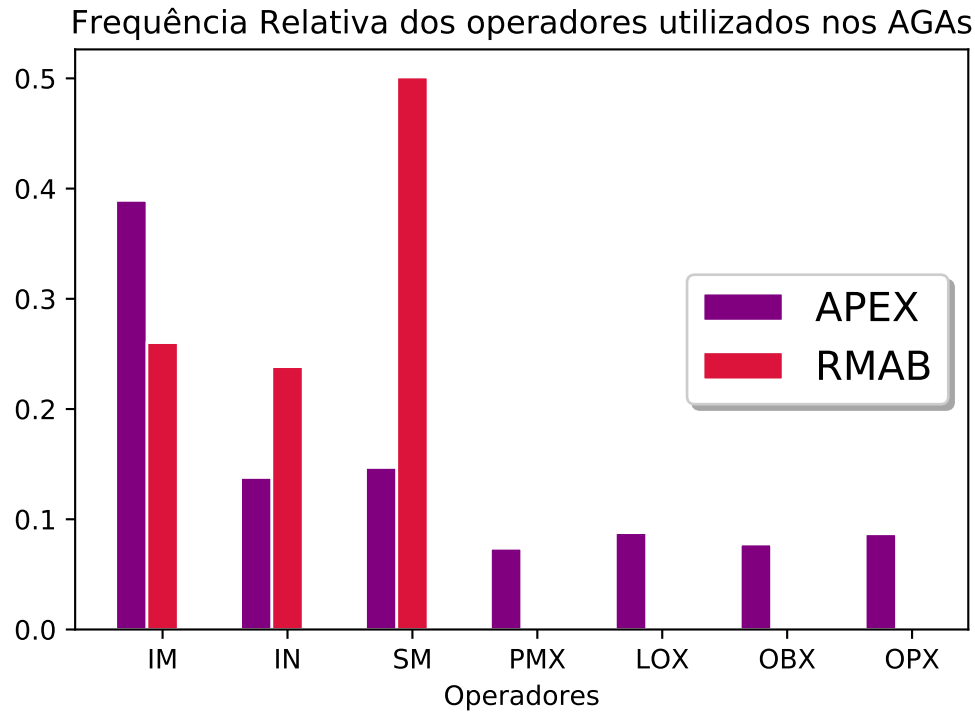


Figura 6.4: Frequência Relativa dos Operadores utilizados nos AGAs para o problema 15_15.

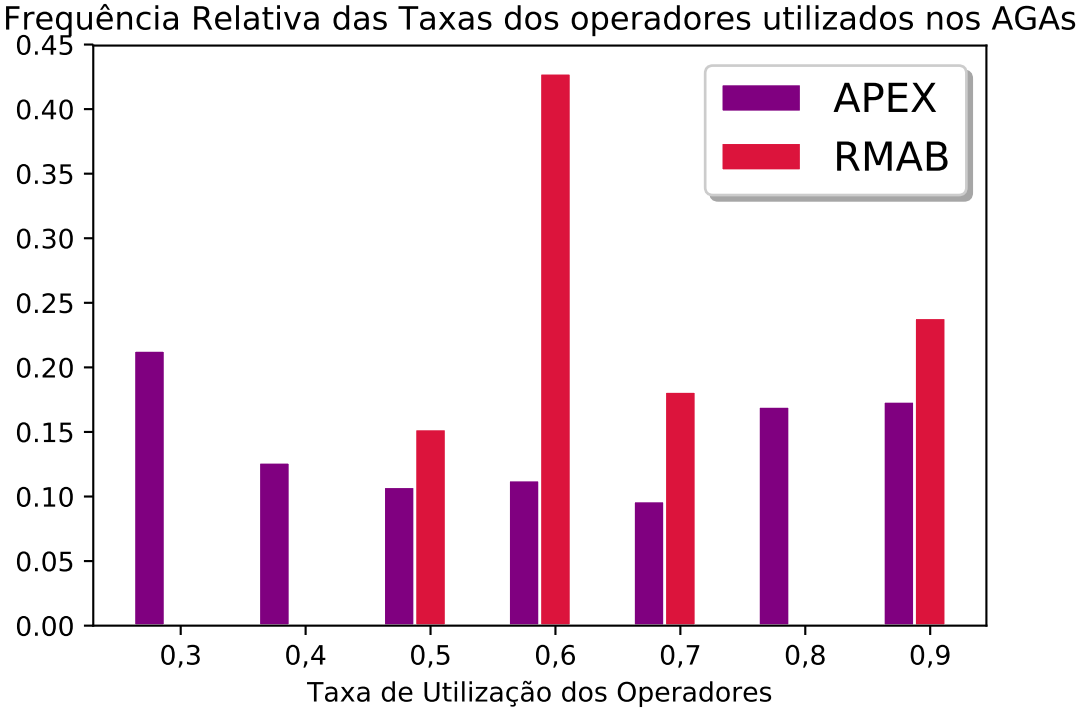


Figura 6.5: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 15_15.

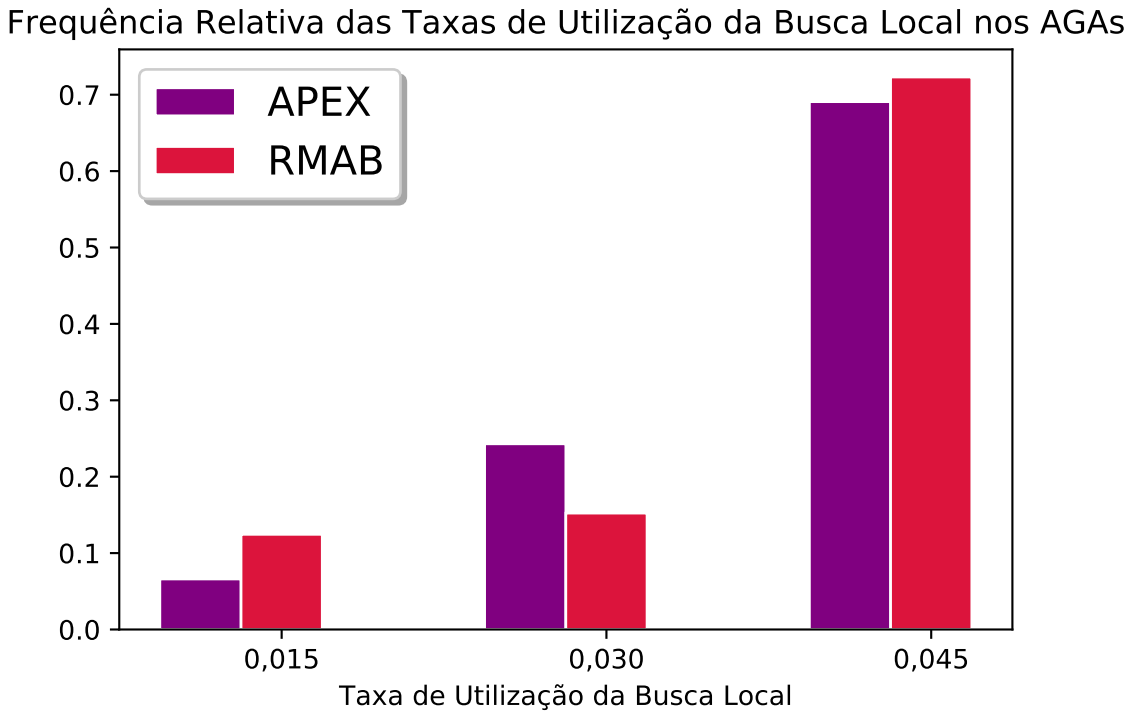


Figura 6.6: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 15_15.

existem diferenças estatísticas entre o comportamento dos algoritmos.

### 6.2.3 20 máquinas e 15 *jobs*

Nesta classe de problemas, o APEX demonstrou melhor desempenho relativo, ficando somente atrás do melhor AG configurado (OPX-IM). A versão probabilística que utiliza as frequências do APEX (PROB_APEX) teve o pior comportamento relativo neste caso.

Considerando a utilização dos operadores e suas taxas de aplicação, o comportamento do APEX é mais uniforme, enquanto o RMAB, semelhante ao caso anterior (15_15), diferenciando apenas no uso significativo do operador PMX, prioriza mais os operadores de mutação e taxas mais elevadas de aplicação.

O comportamento na utilização das taxas de Busca Local é semelhante ao caso anterior, onde majoritariamente os dois algoritmos utilizam a maior taxa definida (0,045).

O p-valor encontrado para este caso é de 0,91. Logo, não existe diferença estatística para o resultados obtidos dos algoritmos.

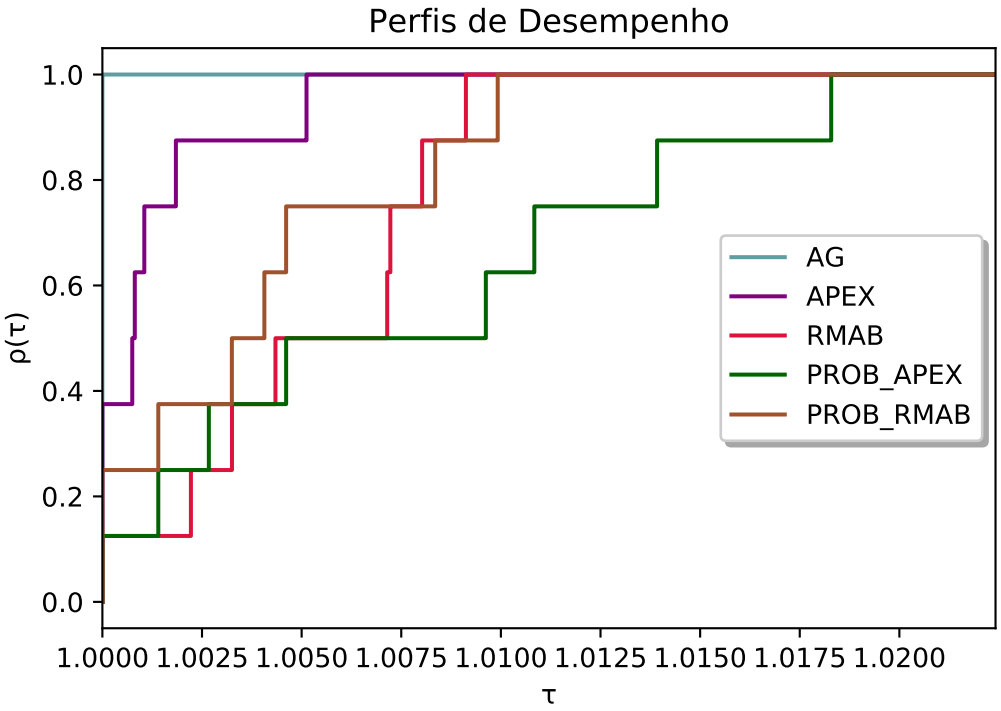


Figura 6.7: Perfis de Desempenho para o problema 20_15.

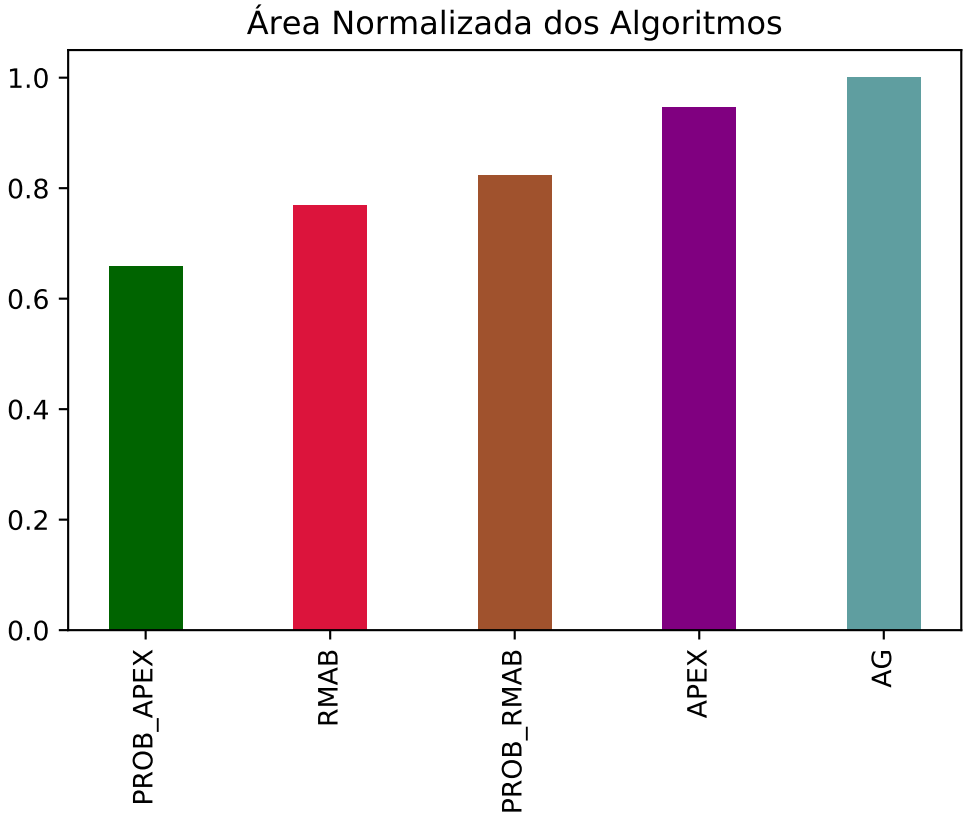


Figura 6.8: Área Normalizada dos PDs para o problema 20_15.

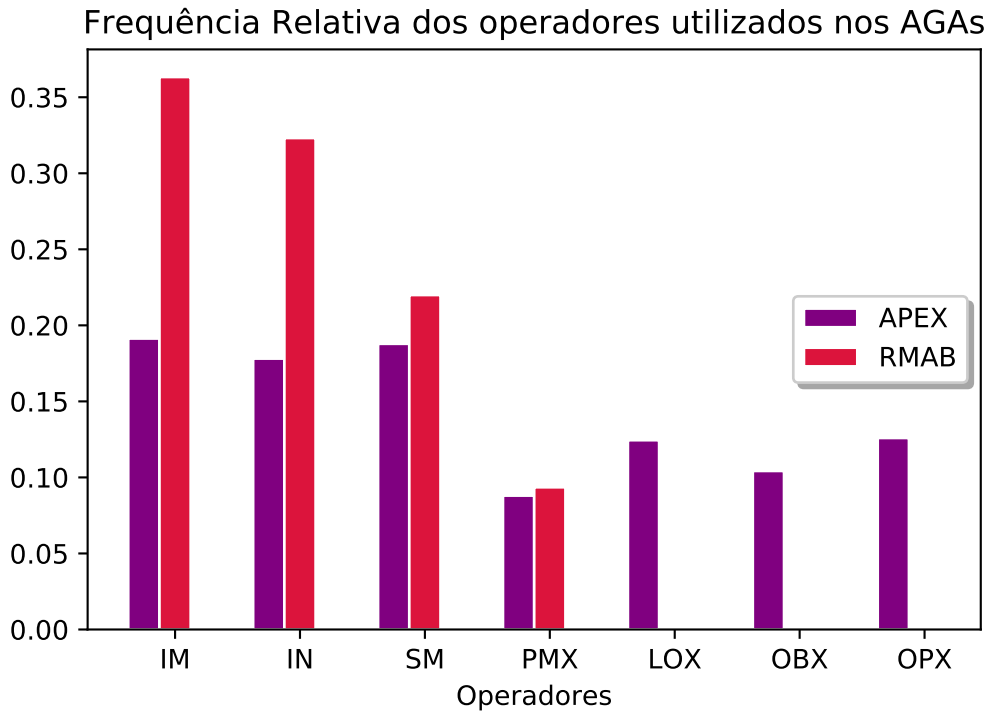


Figura 6.9: Frequência Relativa dos Operadores utilizados nos AGAs para o problema 20_15.

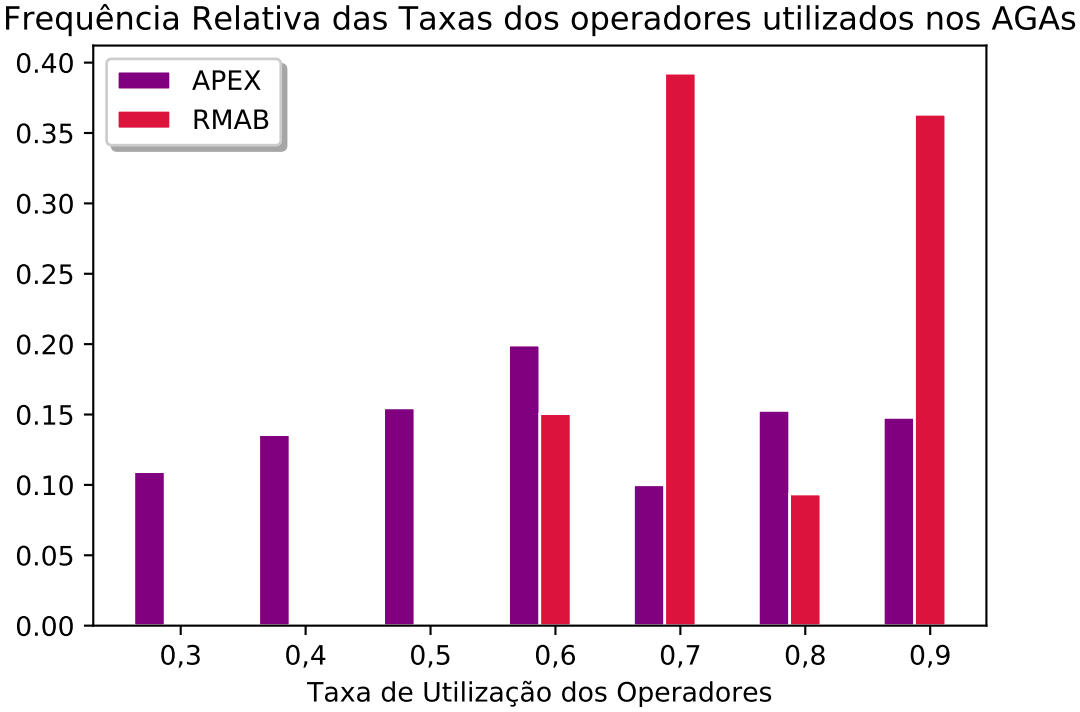


Figura 6.10: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 20_15.

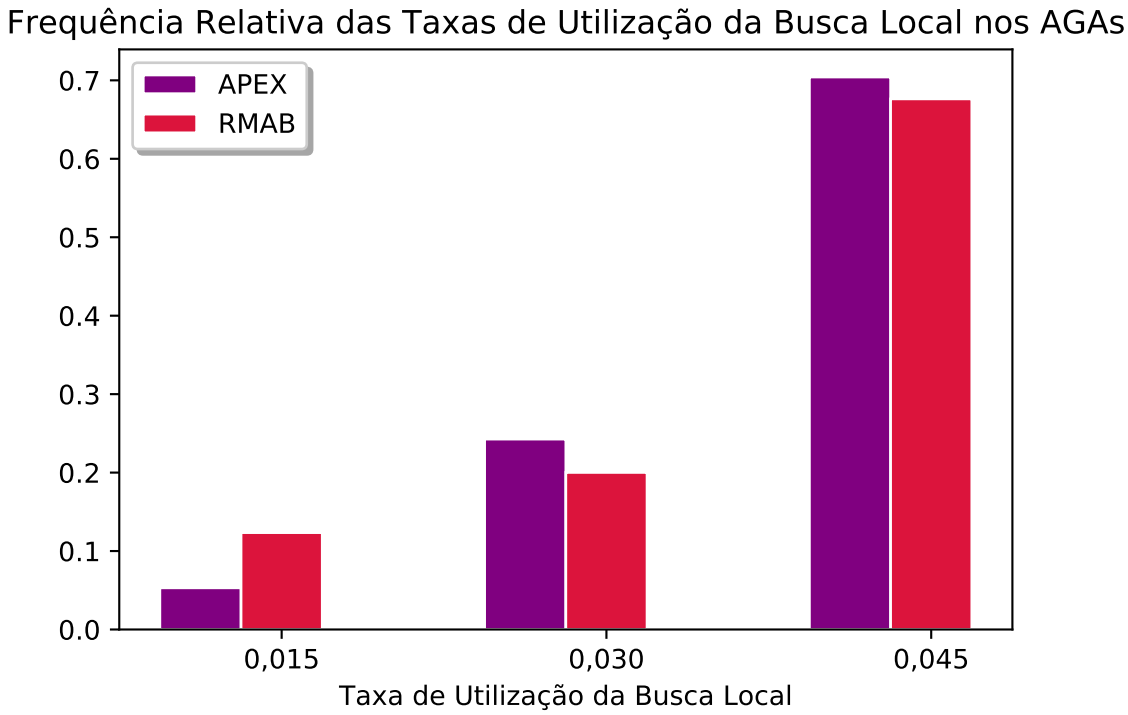


Figura 6.11: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 20_15.

### 6.2.4 20 máquinas e 20 *jobs*

Como no caso anterior, APEX obteve o melhor desempenho relativo, ficando somente atrás do AG (OPX-IN). O pior desempenho foi do PROB_APEX, que consegue resolver todos os problemas a um fator de aproximadamente 1,013 da melhor solução conhecida.

Para este caso, o RMAB, considerando os operadores e as taxas de aplicação, prioriza em sua maioria as maiores taxas (maiores que 0,6) e os operadores de mutação. O operador de recombinação PMX aparece com valor significativo dentre os outros operadores (LOX, OBX e OPX)

Como nos casos anteriores, a taxa de busca local mais utilizadas pelo APEX e RMAB foi a de 0,045.

O p-valor deste caso é 0,96, concluindo-se que não existe diferença estatística entre os resultados de todos os algoritmos.

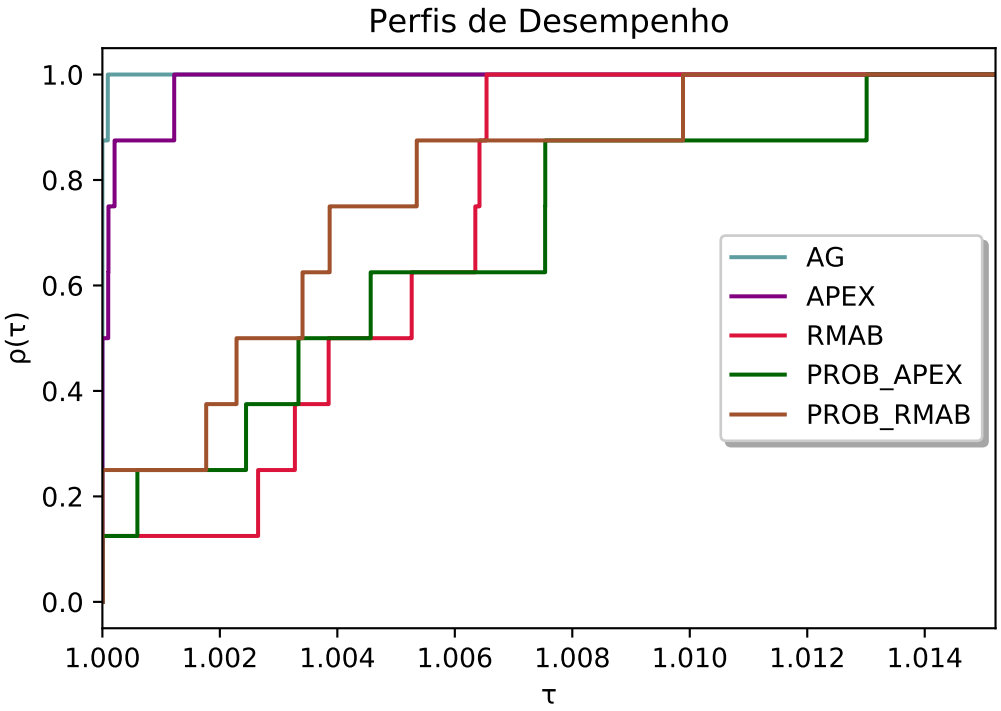


Figura 6.12: Perfis de Desempenho para o problema 20_20.

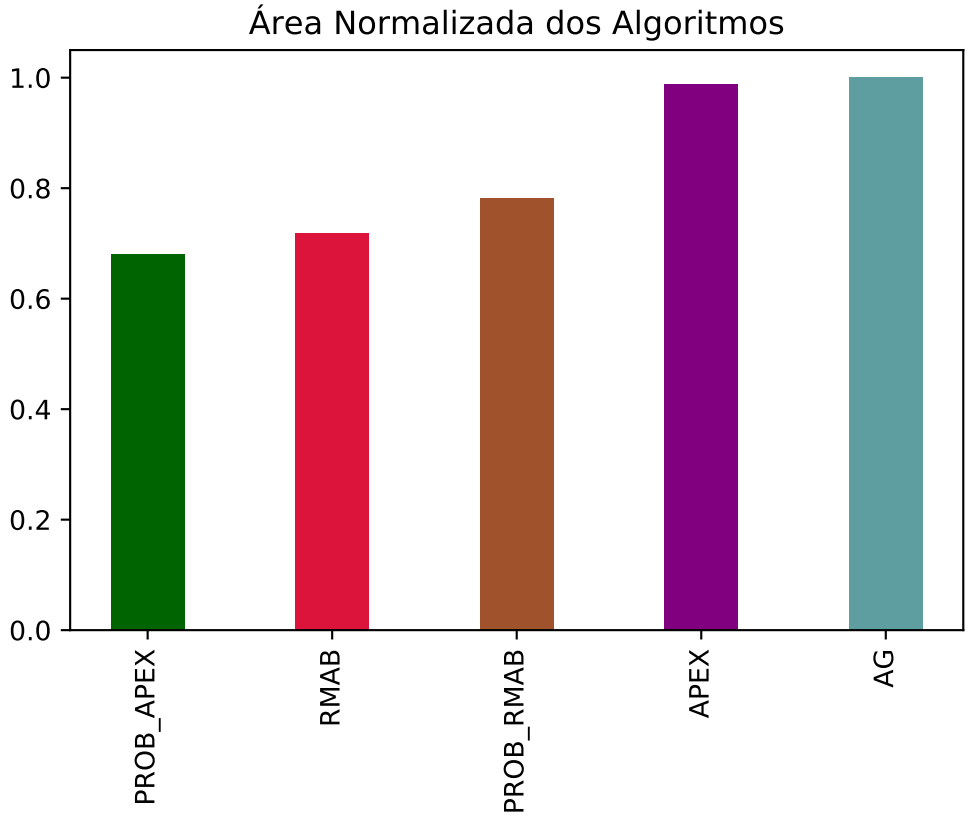


Figura 6.13: Área Normalizada dos PDs para o problema 20_20.

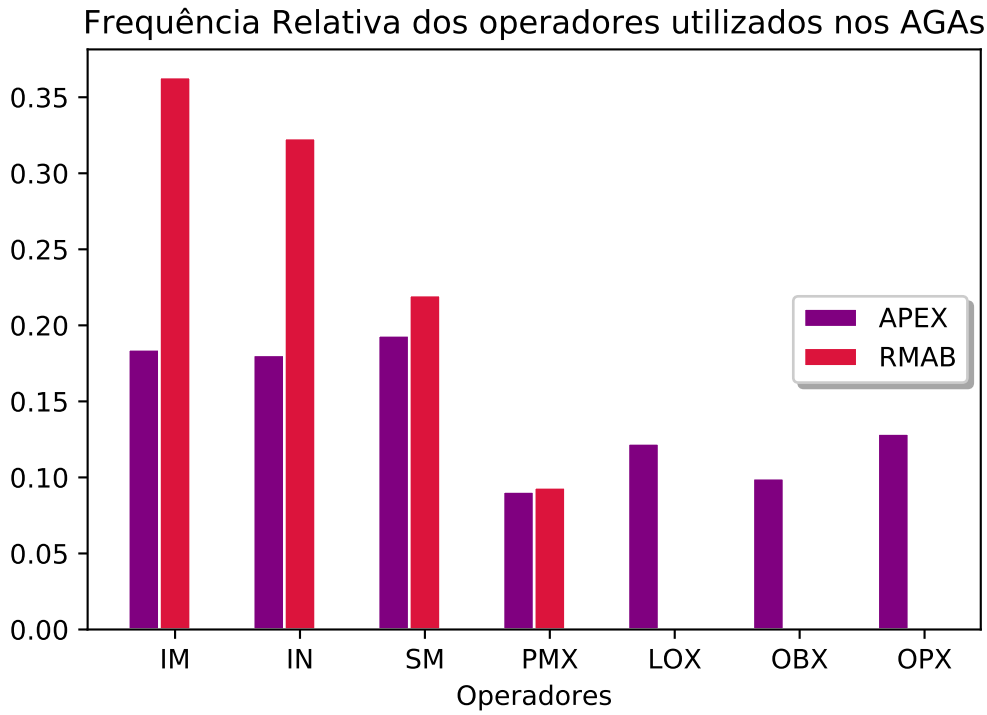


Figura 6.14: Frequência Relativa dos Operadores utilizados nos AGAs para o problema 20_20.

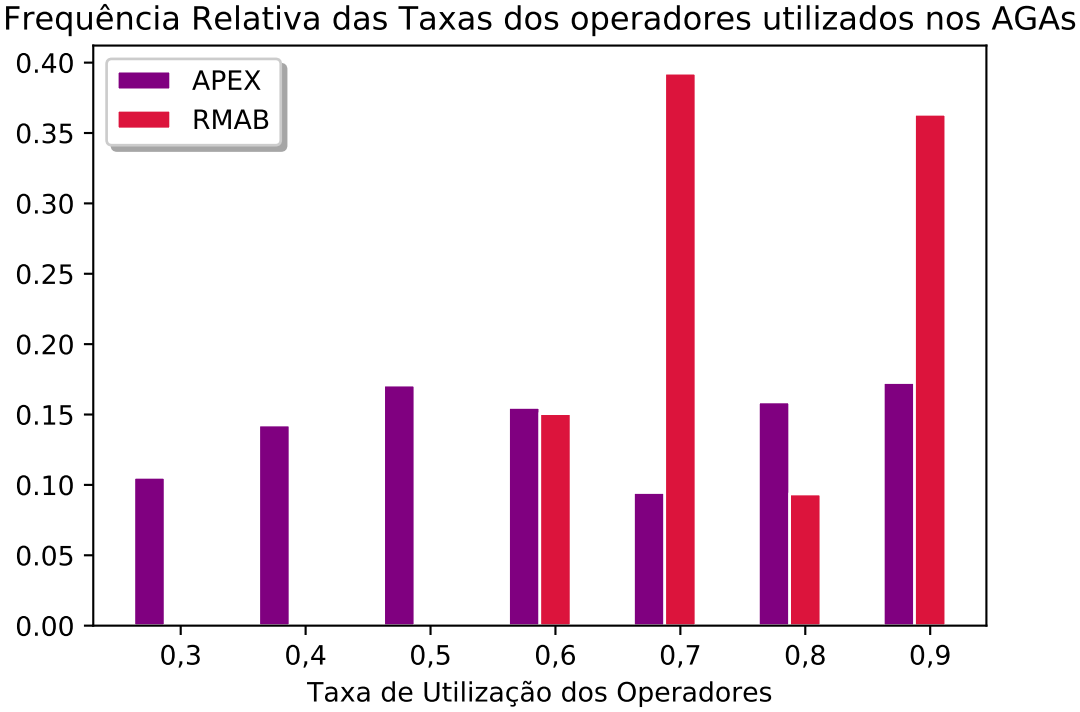


Figura 6.15: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 20_20.

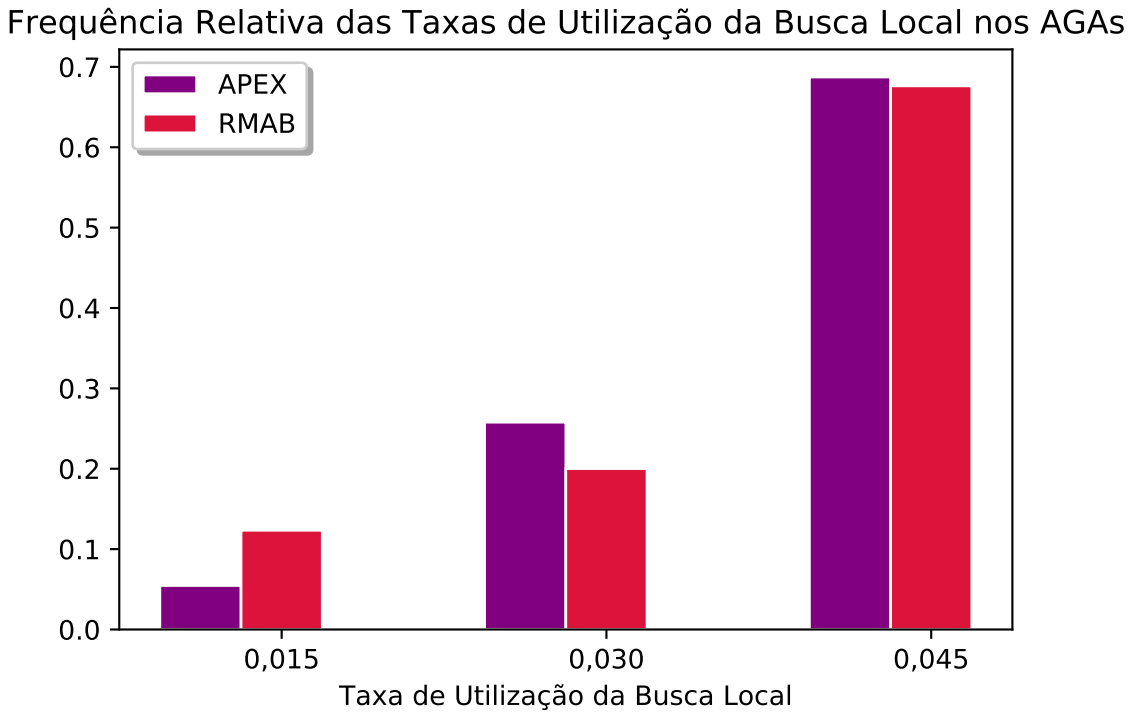


Figura 6.16: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 20_20.

### 6.2.5 30 máquinas e 15 *jobs*

Apesar da proximidade dos valores de área normalizada da maioria dos algoritmos analisados, o AG permanece com o melhor desempenho relativo, seguido do APEX e o PROB_RMAB. O pior desempenho foi registrado pelo PROB_APEX, que consegue resolver todos os problemas com um fator de aproximadamente 1,02 da melhores soluções conhecidas.

Nesta classe de problema, a houve uma diferença significativa na utilização do operador IN pelo RMAB em relação à todos os outros operadores. O operador OPX também aparece com um desempenho significativo comparado aos outros problemas. O APEX mantém a característica de utilização de todos os operadores com frequência similar. Para as taxas de utilização dos operadores, um comportamento diferente dos problemas menores pode ser observado: enquanto o APEX manteve o mesmo comportamento, o RMAB utilizou em sua maioria as taxas 0,3, 0,6 e 0,9.

O RMAB, ao contrário dos problemas já analisados, utilizou com mais frequência a taxa de 0,030 para este problema e não obteve frequência significativa para a taxa de

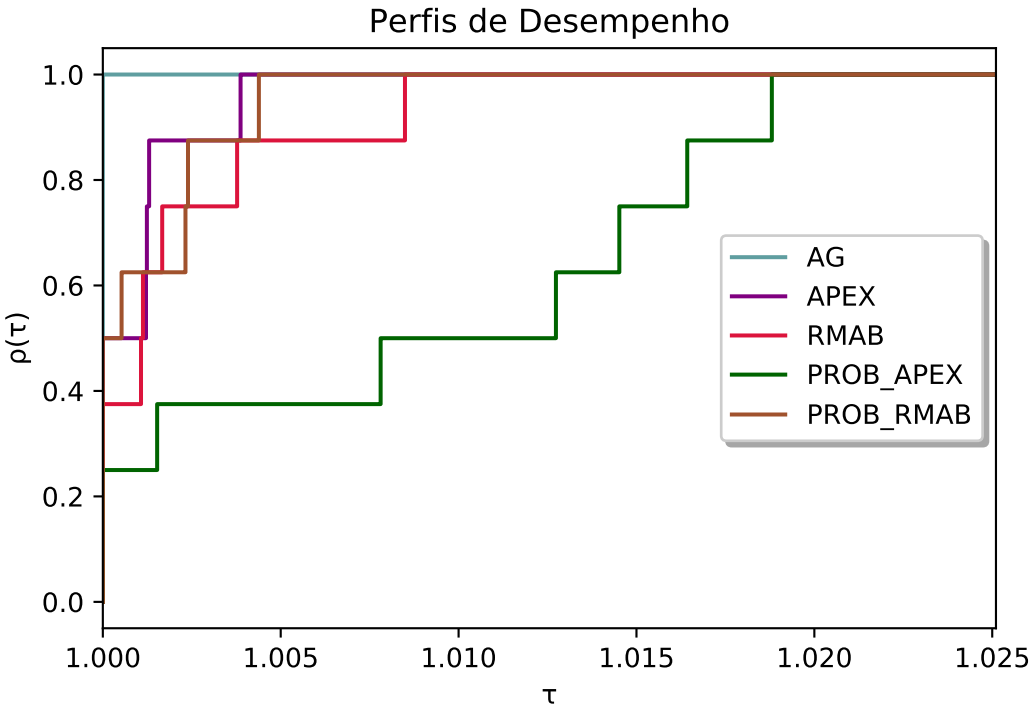


Figura 6.17: Perfis de Desempenho para o problema 30_15.

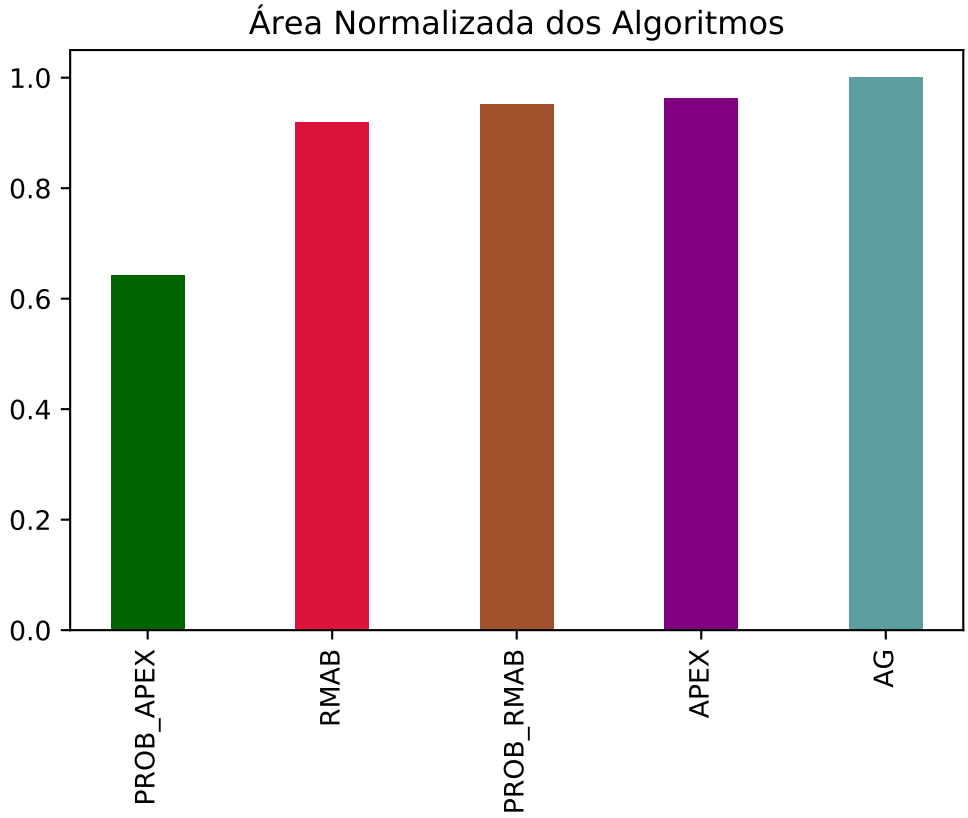


Figura 6.18: Área Normalizada dos PDs para o problema 30_15.

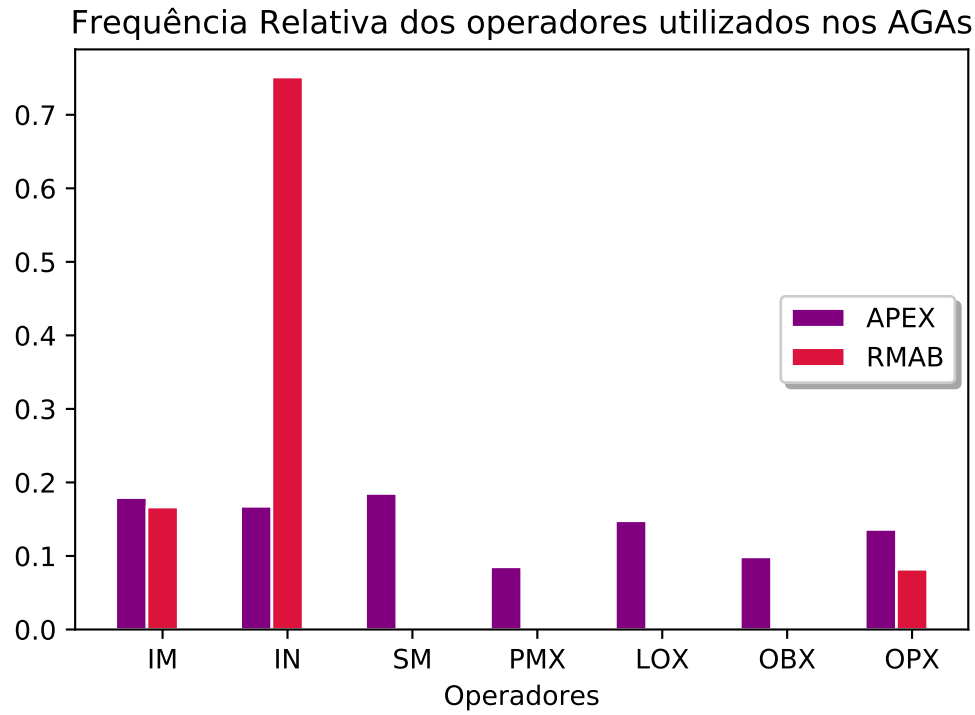


Figura 6.19: Frequência Relativa dos Operadores utilizados nos AGAs para o problema 30_15.

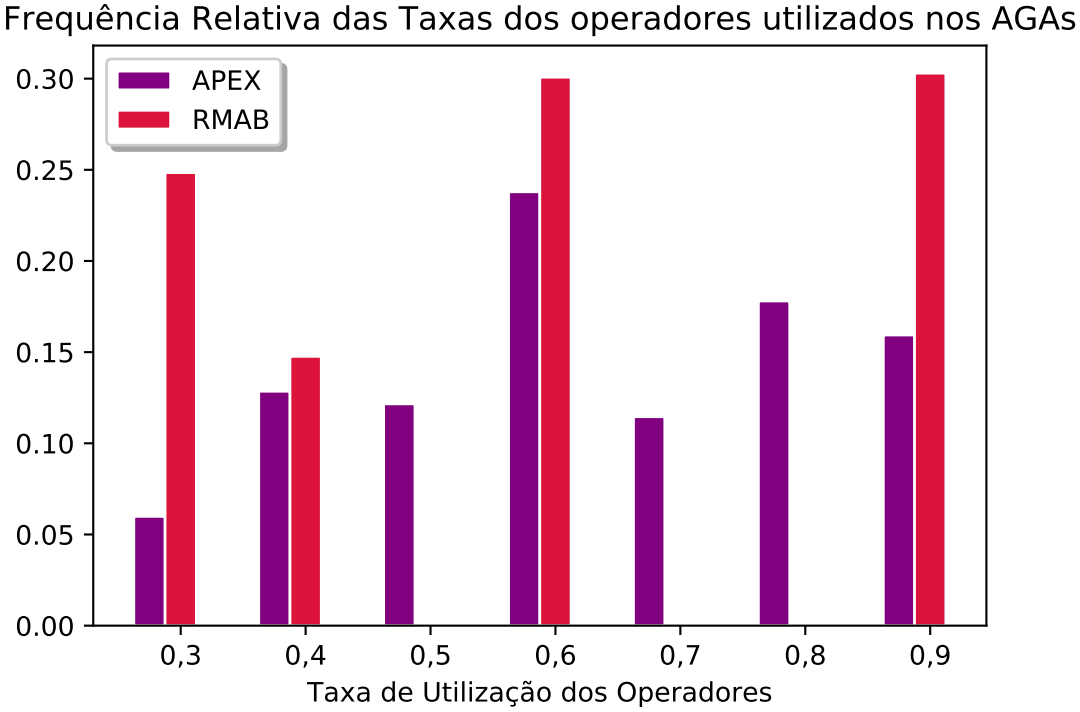


Figura 6.20: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 30_15.

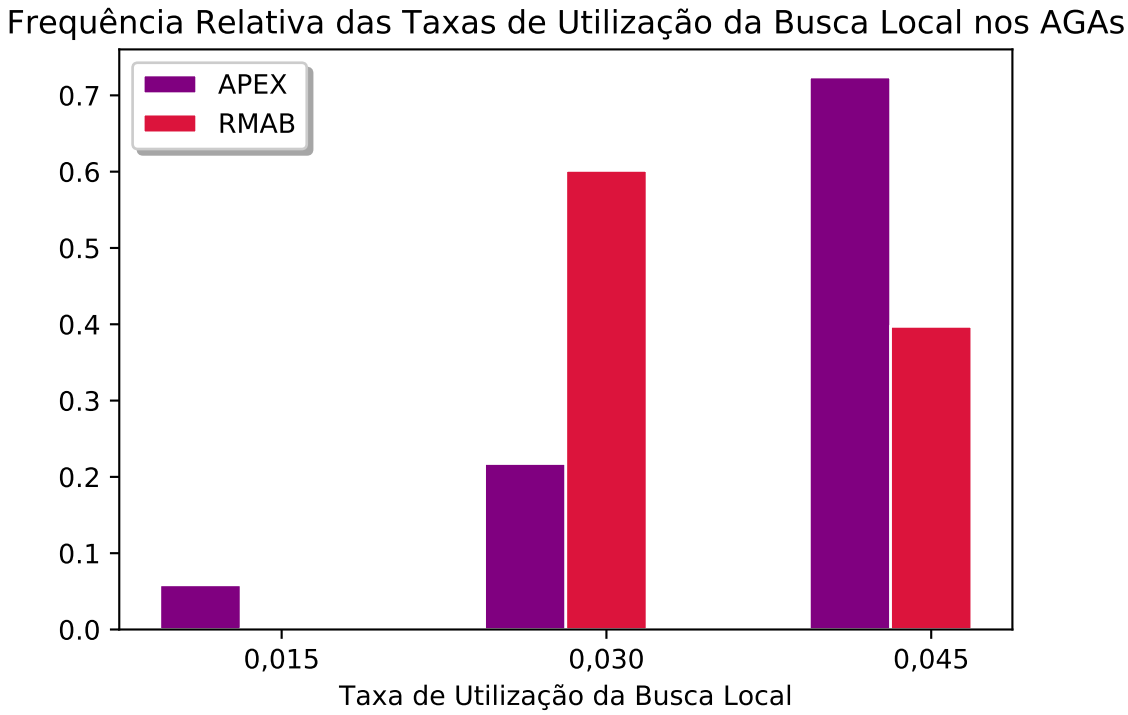


Figura 6.21: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 30_15.

menor valor (0,015). O APEX, como visto anteriormente, manteve a utilização das taxas proporcional aos valores.

O p-valor encontrado para esta classe de problemas é 0.97, o que evidencia a não diferença estatística entre os resultados dos algoritmos.

### 6.2.6 30 máquinas e 20 *jobs*

Para este tipo de problema o melhor comportamento observado após o AG (OPX-SM) foi do PROB_RMAB. O RMAB e o APEX obtiveram comportamento muito semelhante ao PROB_RMAB e o PROB_APEX teve o pior desempenho relativo, com sua área normalizada próxima a metade da área do AG.

Considerando a escolha dos operadores, o RMAB, em sua maioria, utilizou o operador IN e o APEX o IM. Como no caso anterior, o RMAB aparece com frequências significantes para o operador OPX. As taxas de utilização dos operadores foram bem distribuídas para o APEX, enquanto o RMAB não obteve valores significativos para as taxas 0,5, 0,7 e 0,8. Para as taxas de utilização da busca local, o comportamento é

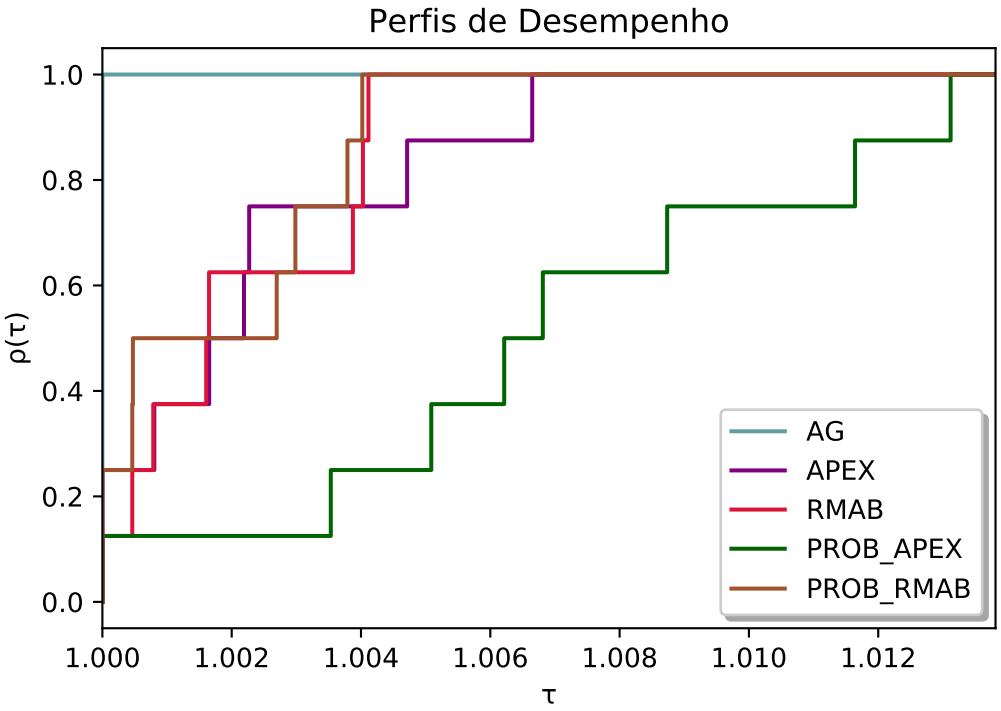


Figura 6.22: Perfis de Desempenho para o problema 30_20.

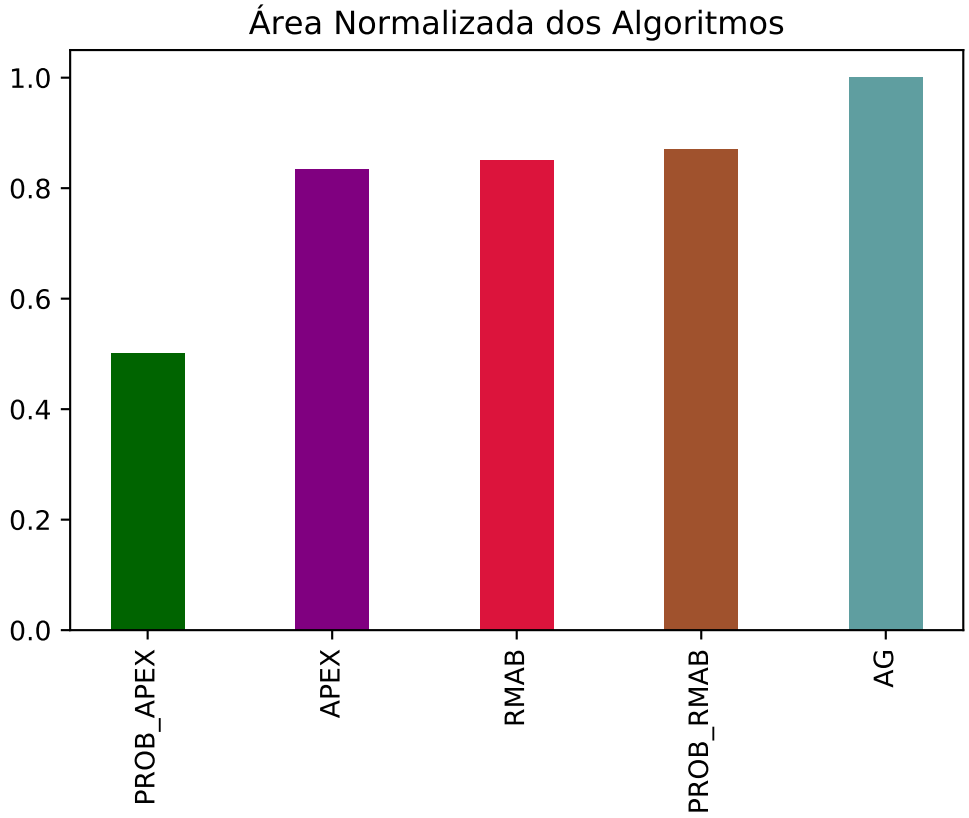


Figura 6.23: Área Normalizada dos PDs para o problema 30_20.

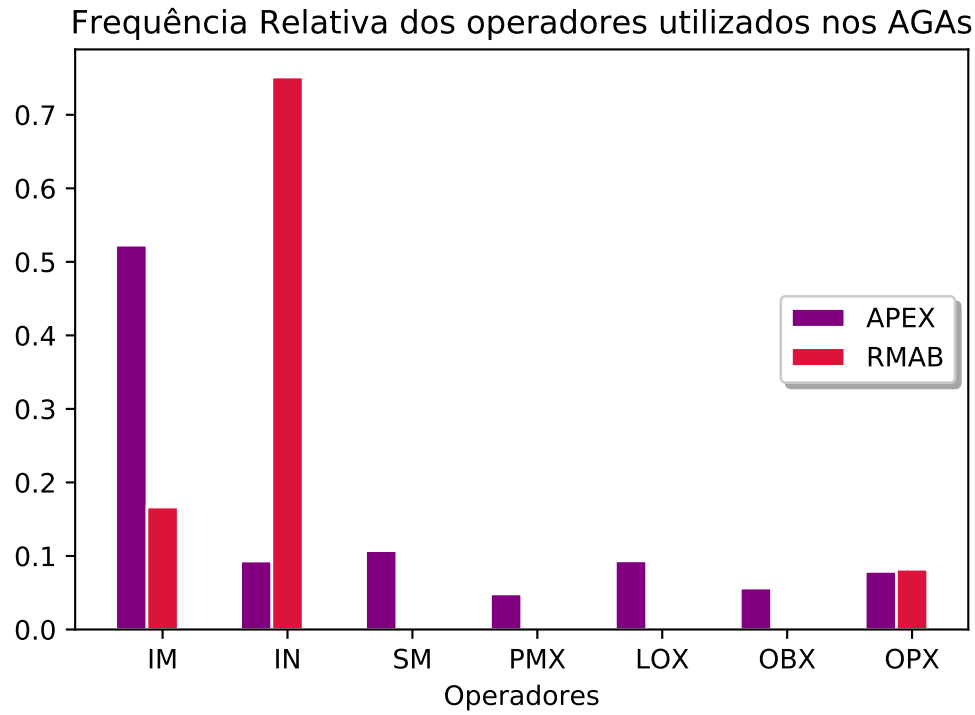


Figura 6.24: Frequência Relativa dos Operadores utilizados nos AGAs para o problema 30_20.

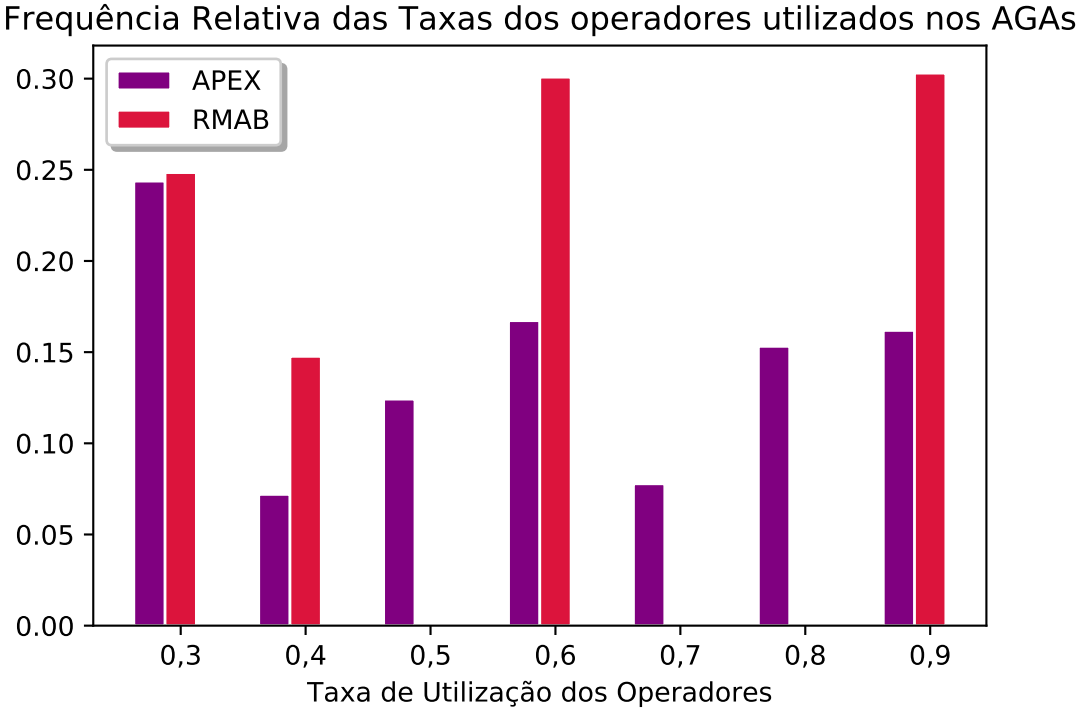


Figura 6.25: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 30_20.

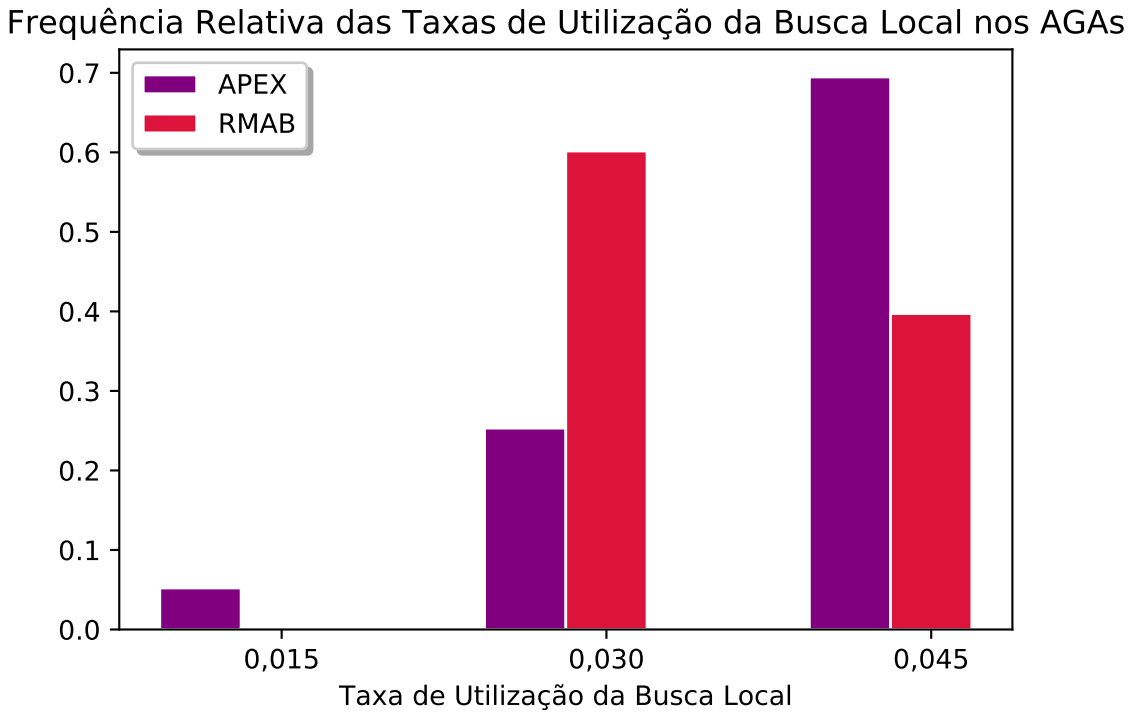


Figura 6.26: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 30_20.

semelhante ao visto para os problemas de 30 *jobs* e 15 máquinas: O APEX utiliza mais a taxa de 0,45 comparado ao RMAB. A situação se inverte para a taxa de 0,030, onde o RMAB é superior ao APEX na utilização. Para a menor taxa, o RMAB não tem valor relativo significativo.

O p-valor para este problema é de 0,93. Logo, não existe diferença estatística no desempenho dos algoritmos analisados.

### 6.2.7 50 máquinas e 15 *jobs*

Neste problema, o PROB_RMAB obteve o segundo melhor desempenho, ficando somente atrás do AG (OPX-SM). O APEX não manteve o mesmo desempenho considerando os problemas de menor tamanho, ficando atrás do PROB_RMAB e do RMAB.

Para os operadores e suas taxas de utilização, o RMAB utilizou em sua maioria os operadores IM e IN e as taxas a partir de 0,5. O APEX manteve a utilização semelhante dos operadores. Para as taxas de utilização, o APEX, em sua maioria, utilizou as maiores taxas, mas ainda mantendo frequências significativas para as de menor valor.

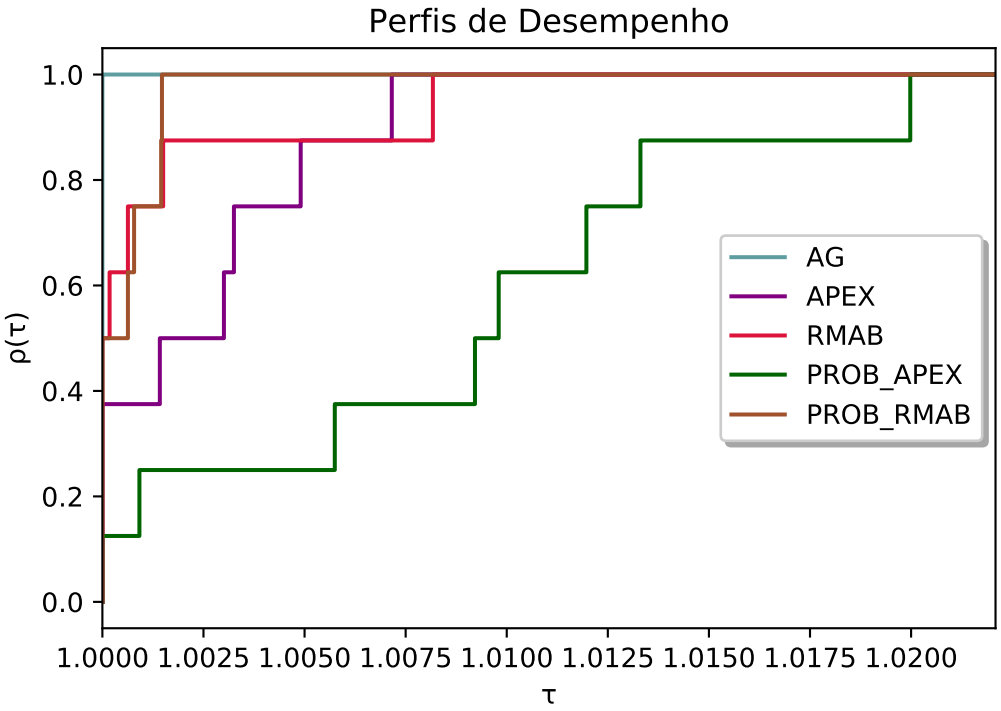


Figura 6.27: Perfis de Desempenho para o problema 50_15.

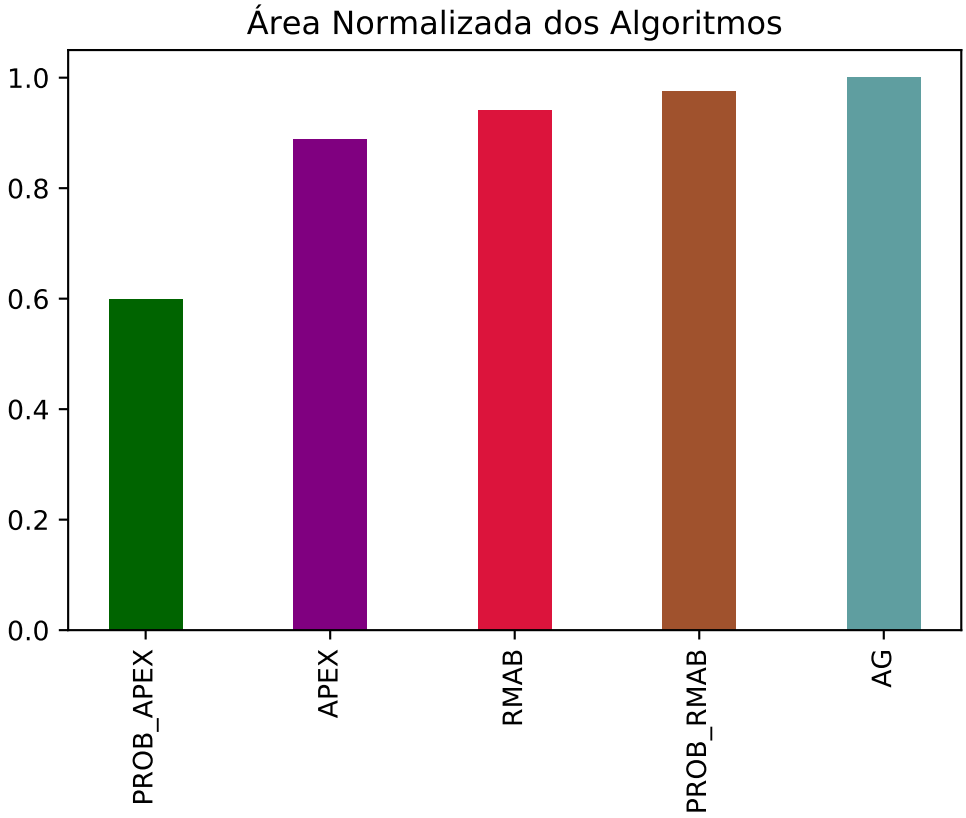


Figura 6.28: Área Normalizada dos PDs para o problema 50_15.

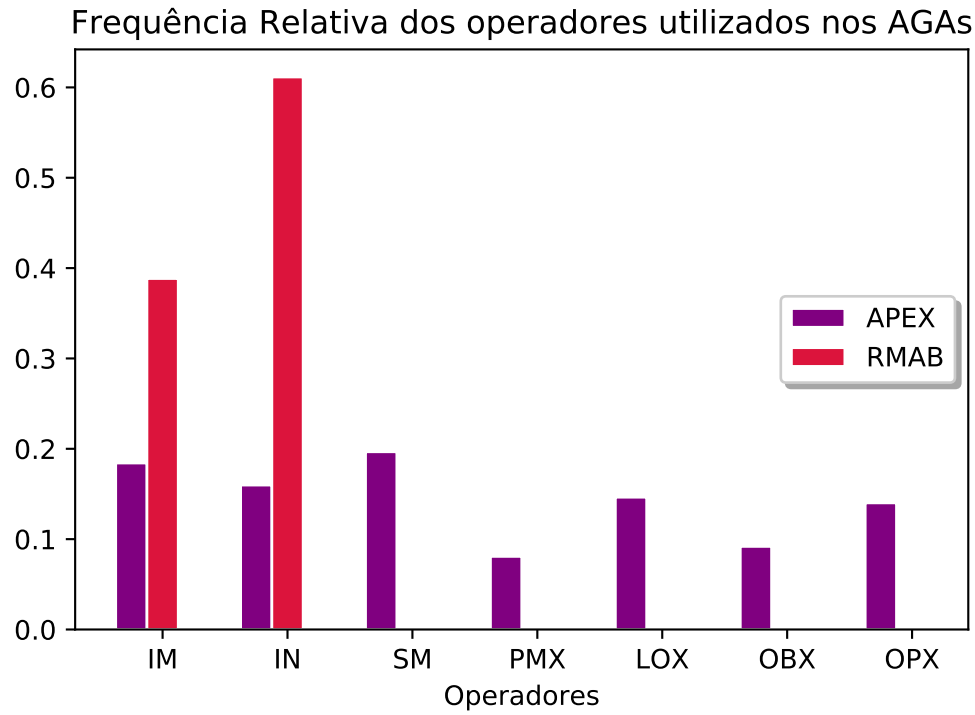


Figura 6.29: Frequência Relativa dos Operadores utilizados nos AGAs para o problema 50_15.

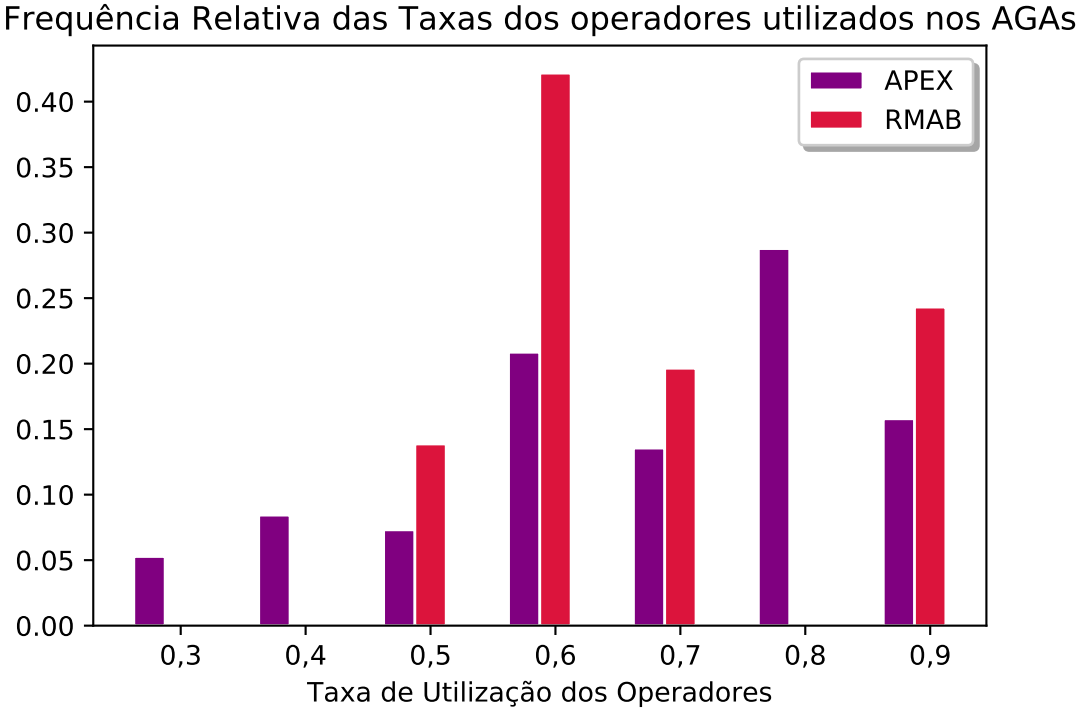


Figura 6.30: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 50_15.

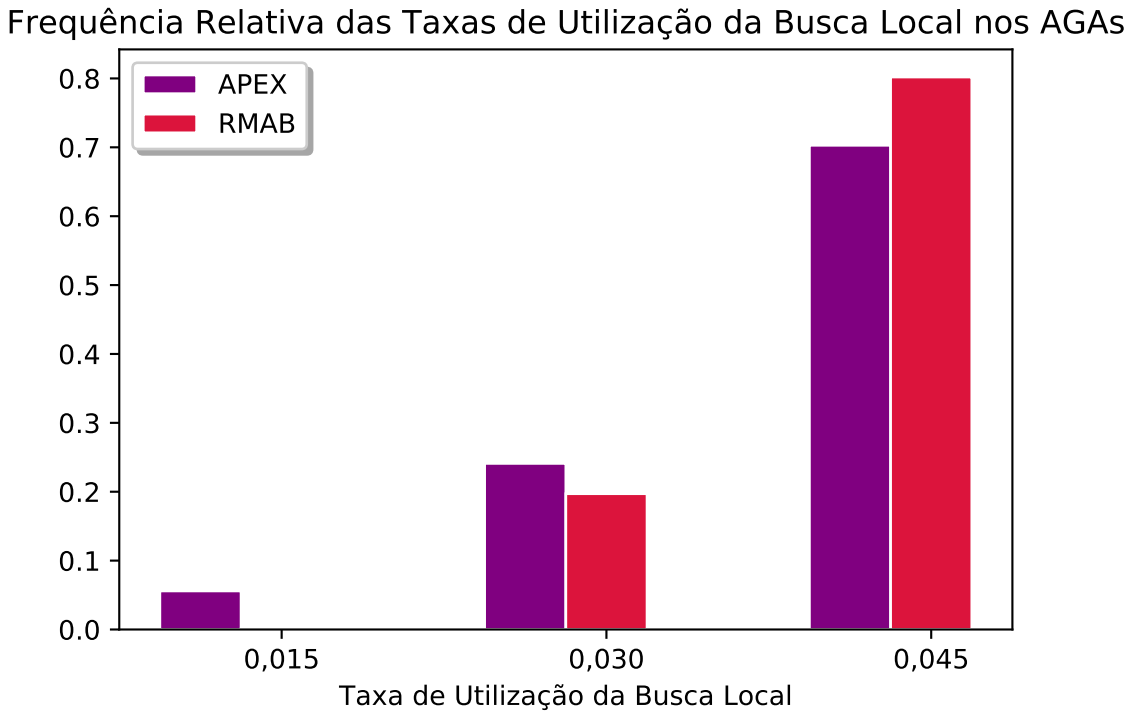


Figura 6.31: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 50_15.

Considerando as taxas de utilização da busca local, o APEX e o RMAB mantiveram o comportamento visto nos problemas de tamanho menores.

Os algoritmos não tiveram desempenho estatístico distintos, pois o p-valor para este problema foi de 0,92.

### 6.2.8 Todos os problemas

Considerando todos os problemas, o algoritmo que tem comportamento mais próximo do AG é o APEX, seguido do PROB_RMAB e do RMAB.

As frequências dos operadores e as taxas de utilização dos operadores tiveram o seu comportamento semelhante ao visto anteriormente. O RMAB não acumulou valores significativos para os operadores LOX e OBX e obteve mais de 50% de utilização do operadore IN. O APEX utilizou todos os operadores, sendo o IM o com maior frequência (30%). Quando considerado todos os problemas, o RMAB utiliza 30% das vezes as taxas 0,6 e 0,9 e o APEX não passa de 20% em nenhuma taxa, tendo valores semelhantes em todas elas.

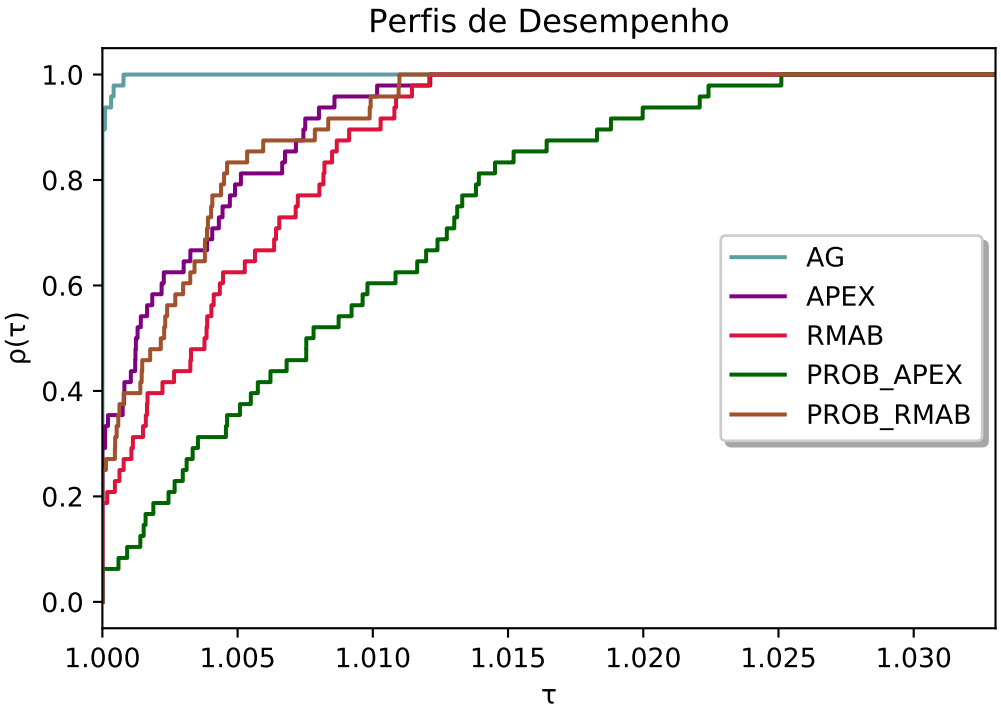


Figura 6.32: Perfis de Desempenho para todos os problemas.

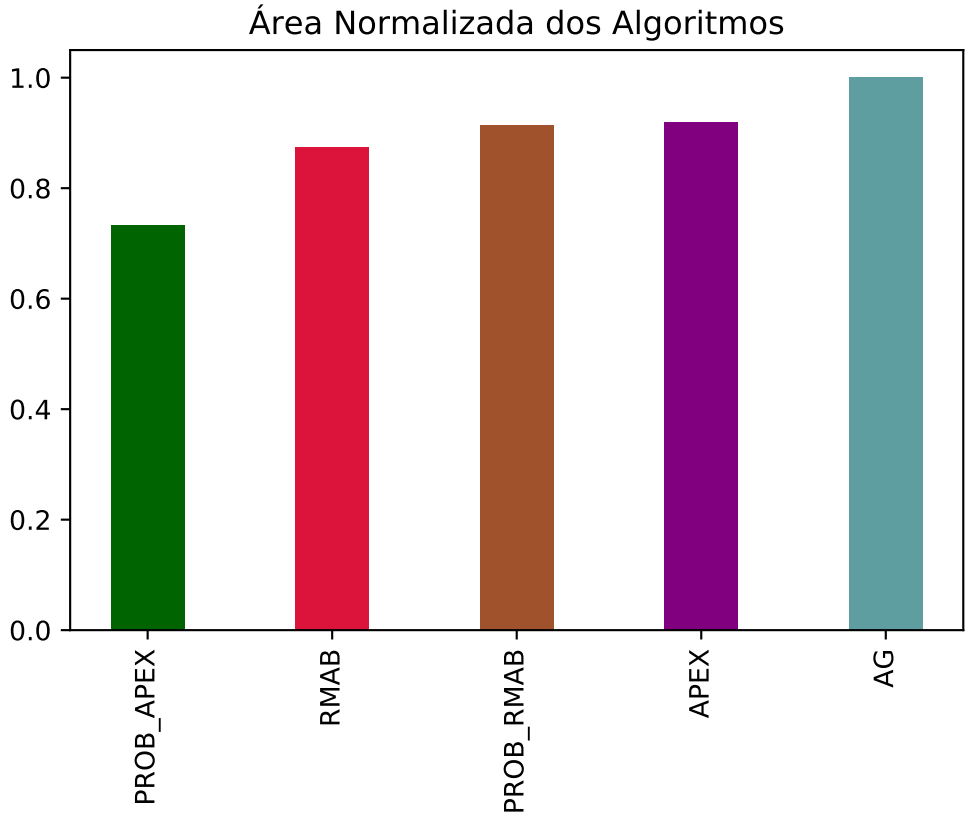


Figura 6.33: Área Normalizada dos PDs para todos os problemas.

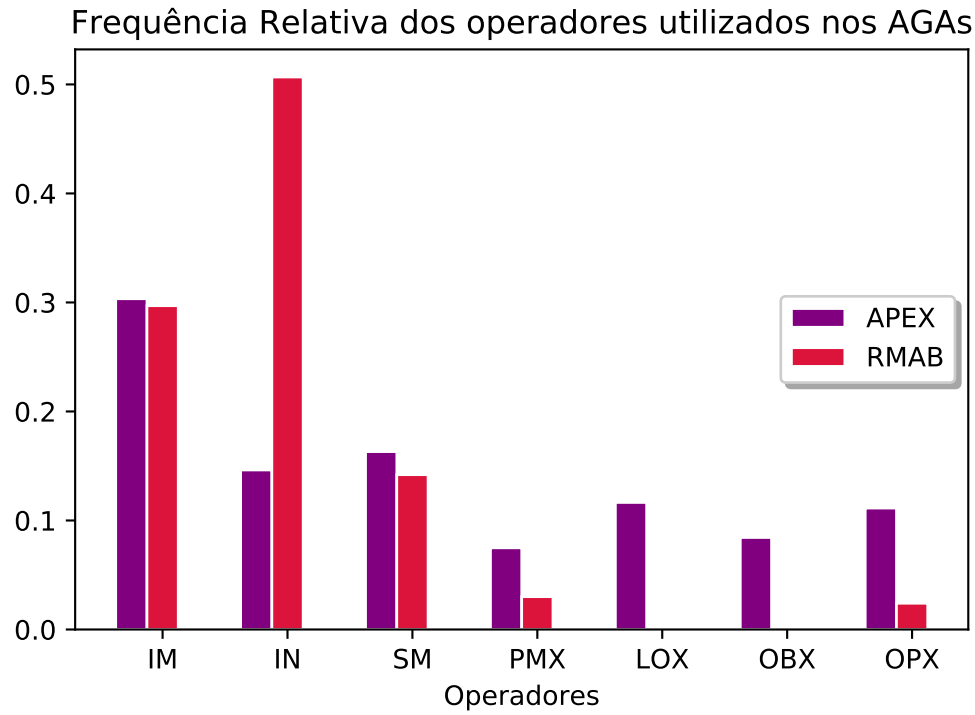


Figura 6.34: Frequência Relativa dos Operadores utilizados nos AGAs para todos os problemas.

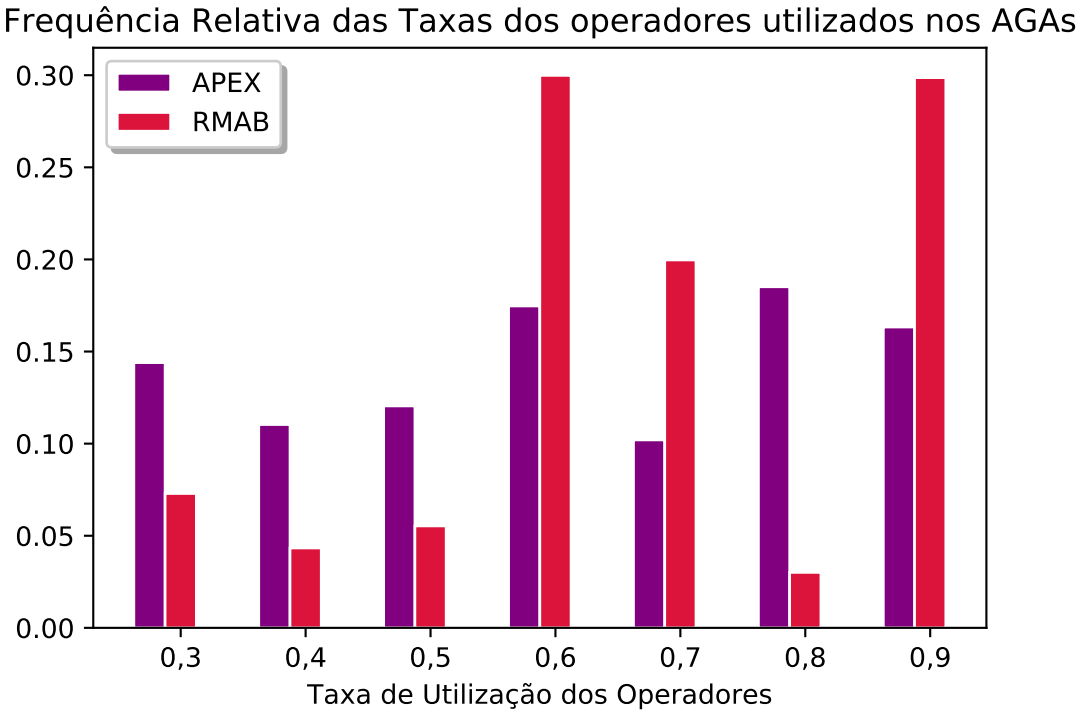


Figura 6.35: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para todos os problemas.

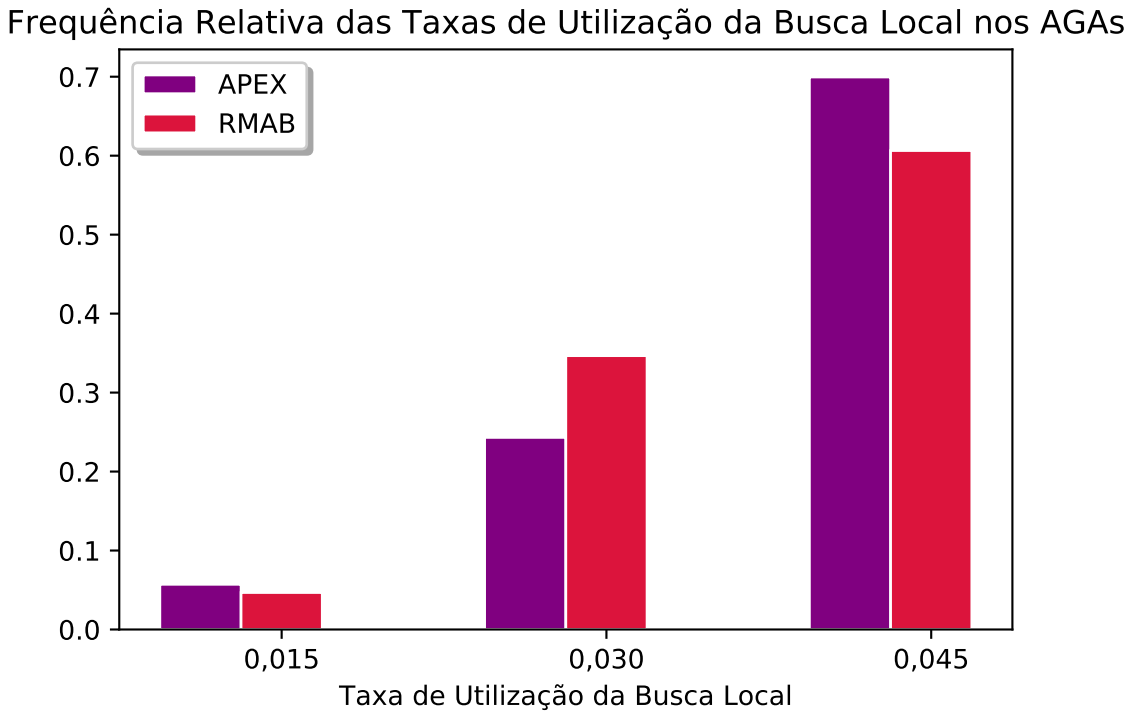


Figura 6.36: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para todos os problemas.

As taxas de utilização mais frequentes foram 0,045 e 0,030. O APEX acumulou aproximadamente 70% e 35% para as taxas 0,045 e 0,030 respectivamente. Já o RMAB 60% e 35% para as taxas 0,045 e 0,030 respetivamente.

O p-valor considerando todos os problemas foi de 0,99, o que nos permite afirmar que não existem diferenças estatísticas neste experimento.

### 6.2.9 Análise do Experimento 1

Em todos os casos observados, a análise estatística não detectou diferenças entre os algoritmos analisados, mesmo com diferenças significativas de desempenho relativo, quando observado a área abaixo da curva dos PDs de todos os algoritmos. Em todos os casos, os fatores $\tau$ não ultrapassaram 1,03, o que é possível afirmar que mesmo existindo diferença numérica entre a qualidade das soluções, ela é muito baixa.

O APEX obteve o melhor desempenho dentre os AGAs utilizados, mas a sua geração de frequências e taxas não fizeram com que o PROB_APEX tivesse desempenho próximo ao AG considerado. Já O RMAB, apesar de não ficar na frente do APEX na

maioria dos problemas, conseguiu gerar distribuições de frequências que fizeram com que o comportamento do PROB_RMAB fosse satisfatório.

Na maioria dos casos, os operadores de mutação foram os mais utilizados, sendo que o APEX ainda manteve frequências significativas para os operadores de recombinação, enquanto o RMAB não obteve frequências relevantes. Em relação às taxas de utilização dos operadores, o comportamento tende à utilização de taxas maiores por parte dos dois algoritmos, sendo o RMAB o que menos utiliza as taxas de menores valores. Para compensar o comportamento dos operadores de mutação, ambos os algoritmos utilizaram frequências maiores de busca local, e em alguns casos o RMAB não chega a ter valores significativos para a de menor valor (0,015).

## 6.3 Experimento 2

Neste experimento os problemas de menor tamanho (15x15) são utilizados para configurar os parâmetros definidos pelo usuário nos algoritmos. Todas as instâncias desta configuração são utilizadas para definir quais parâmetros serão utilizados para o restante dos problemas teste. A intenção deste tipo de teste é verificar se o comportamento dos algoritmos em problemas pequenos pode ser extrapolado para problemas maiores.

### 6.3.1 Fase de configuração

Após as execuções dos algoritmos variando os parâmetros e operadores para todas as instâncias de menor tamanho (972 possibilidades para o AG e 9 possibilidades para cada AGA), foram definidos os melhores desempenhos dos AGs e AGAs seguindo as informações das tabelas abaixo.

Tabela 6.5: Parâmetros dos AGs escolhidos a partir do problema 15x15 no Experimento 2.

| **Op Cross** | **Op Mut** | **Tx Cross** | **Tx Mut** | **TPop** | **TxLS** |
|---|---|---|---|---|---|
| OPX | SM | 0,7 | 0,3 | 50 | 0,045 |

Tabela 6.6: Parâmetros dos AGAs escolhidos a partir do problema 15x15 no Experimento 2.

| **Algoritmo** | **Parâmetros** | **Valor** |
|---|---|---|
| APEX | $\alpha$ | 0,2 |
| APEX | $\beta$ | 0,2 |
| RMAB | $D$ | 1,00 |
| RMAB | $C$ | 0,1 |

### 6.3.2 15 máquinas e 15 *jobs*

Para este caso, as versões probabilísticas PROB_RMAB e PROB_APEX obtiveram melhores desempenhos, ficando atrás somente para o AG (OPX-SM), quando observado a área abaixo da curva dos PDs.

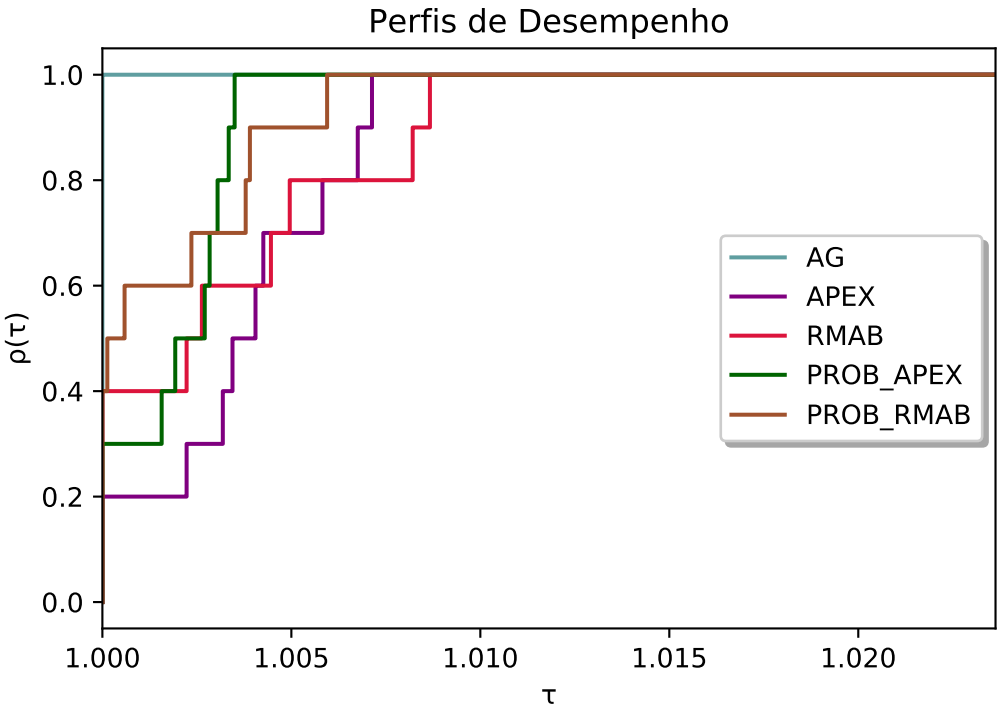


Figura 6.37: Perfis de Desempenho para o problema 15_15.

O APEX manteve baixas frequências relativas de utilização para a maioria dos operadores, diferenciando apenas no operador IM, contabilizando mais de 70% de utilização. O RMAB não contabilizou frequências significativas para os operadores de recombinação e o operador SM foi o mais utilizado. Com relação às taxas de utilização dos operadores, o APEX utilizou, em sua maioria, a taxa de 0,3 e o RMAB, a taxa de 0,6.

As taxas de utilização da busca local mantiveram o mesmo comportamento visto

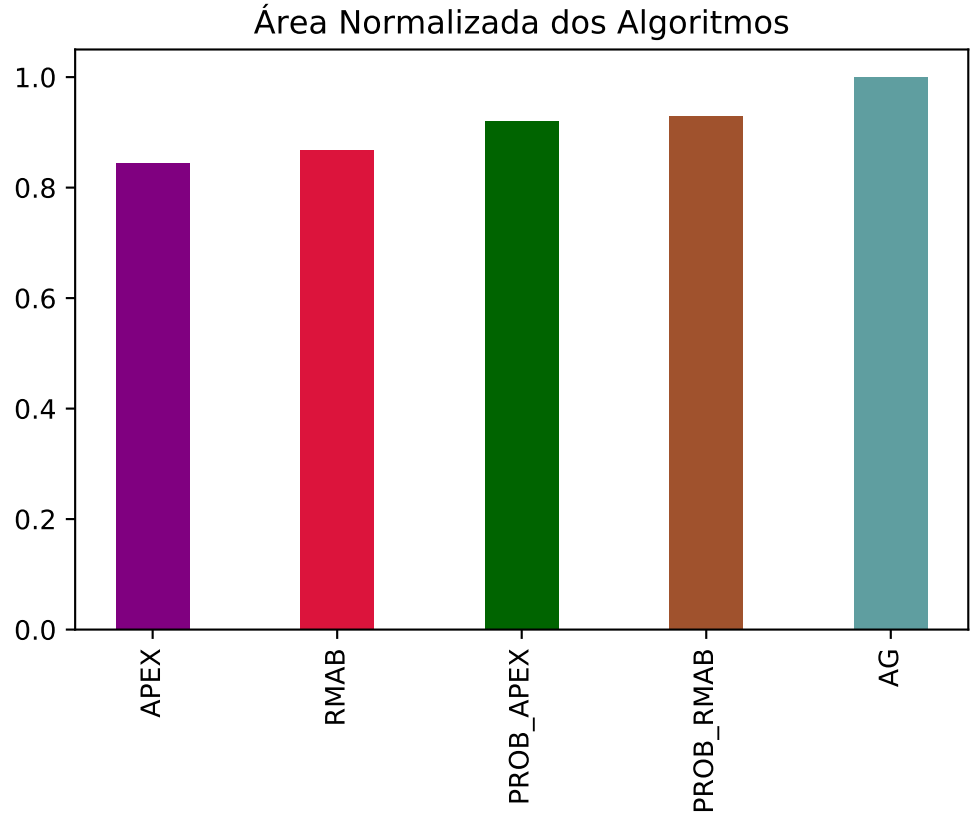


Figura 6.38: Área Normalizada dos PDs para o problema 15_15.

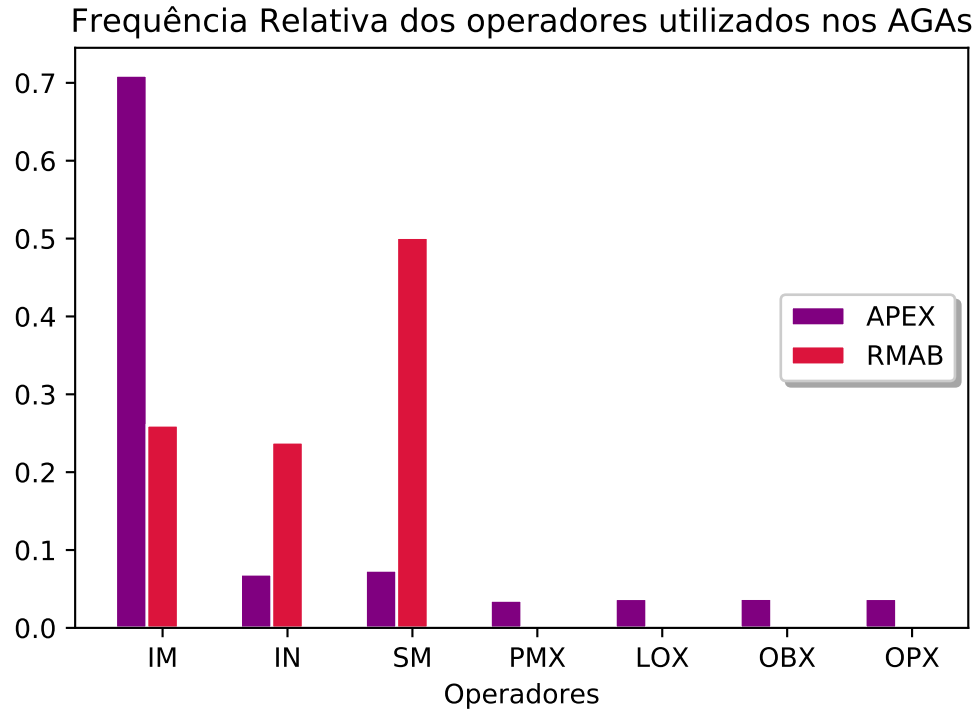


Figura 6.39: Frequência Relativa dos Operadores utilizados nos AGAs para o problema 15_15.

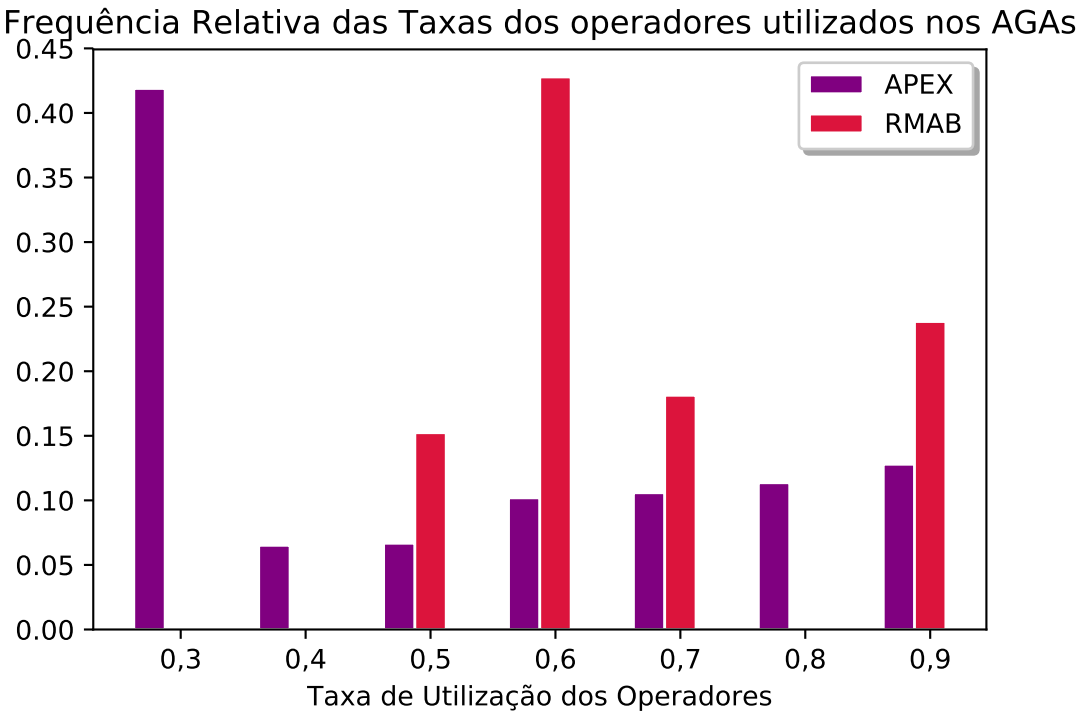


Figura 6.40: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 15_15.

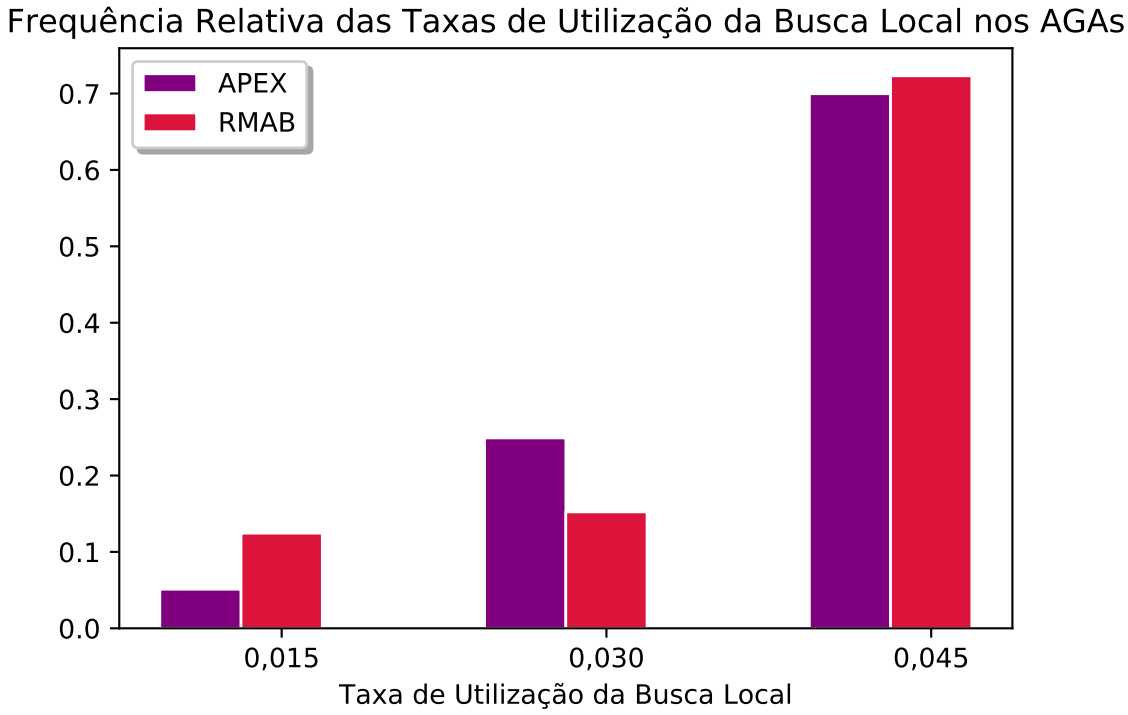


Figura 6.41: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 15_15.

no experimento anterior, com o APEX e o RMAB utilizando mais de 70% das vezes a taxa de 0,045.

O p-valor para este problema foi de 0,98. Logo, não existe diferença estatística entre os resultados gerados pelos algoritmos.

### 6.3.3 20 máquinas e 15 *jobs*

Para este tamanho de problema, o APEX obteve resultado bastante semelhante ao AG, se distanciando consideravelmente dos outros algoritmos.

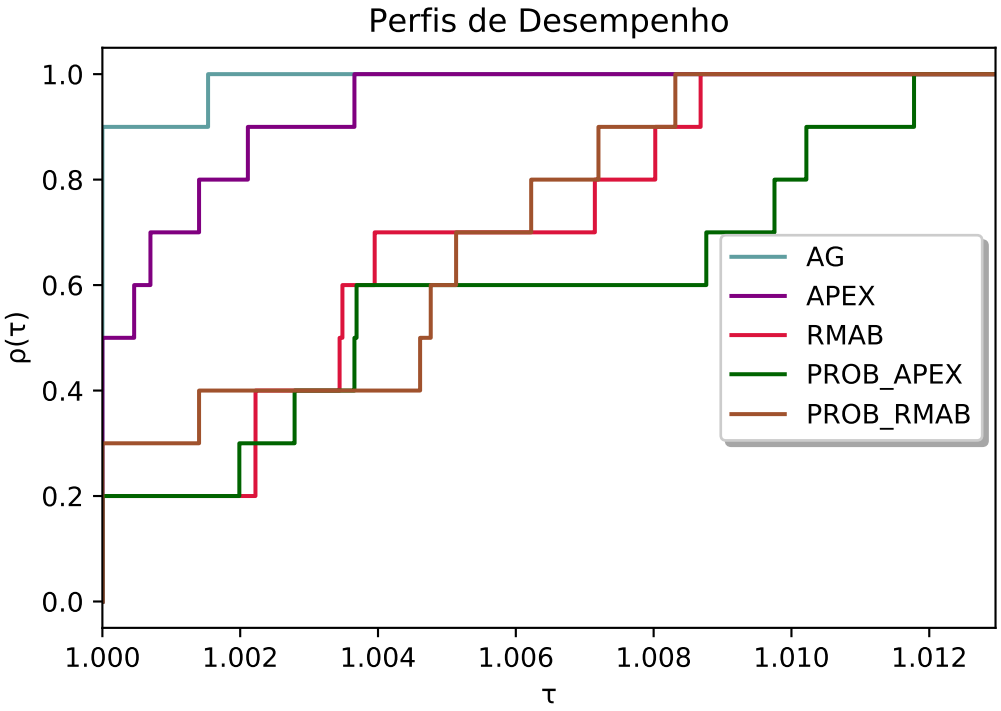


Figura 6.42: Perfis de Desempenho para o problema 20_15.

Os operadores mais utilizados pelo RMAB foram os de mutação e o PMX, enquanto o APEX utiliza a maioria dos operadores com frequência similar, diferindo apendas para o operador IM, onde a frequência de utilização é superior a 50%. O RMAB não tem frequências significativas para as taxas de utilização dos operadores abaixo de 0,5, se concentrando nas taxas 0,7 e 0,9. O APEX tem a maioria das utilizações para a taxa 0,3 e mantém similaridade de valores para o restante das taxas. As taxas de busca local, para os dois algoritmos, mantém o comportamento similar dos problemas de 15_15.

O p-valor para esta classe de problemas e os algoritmos considerados é de 0,96,

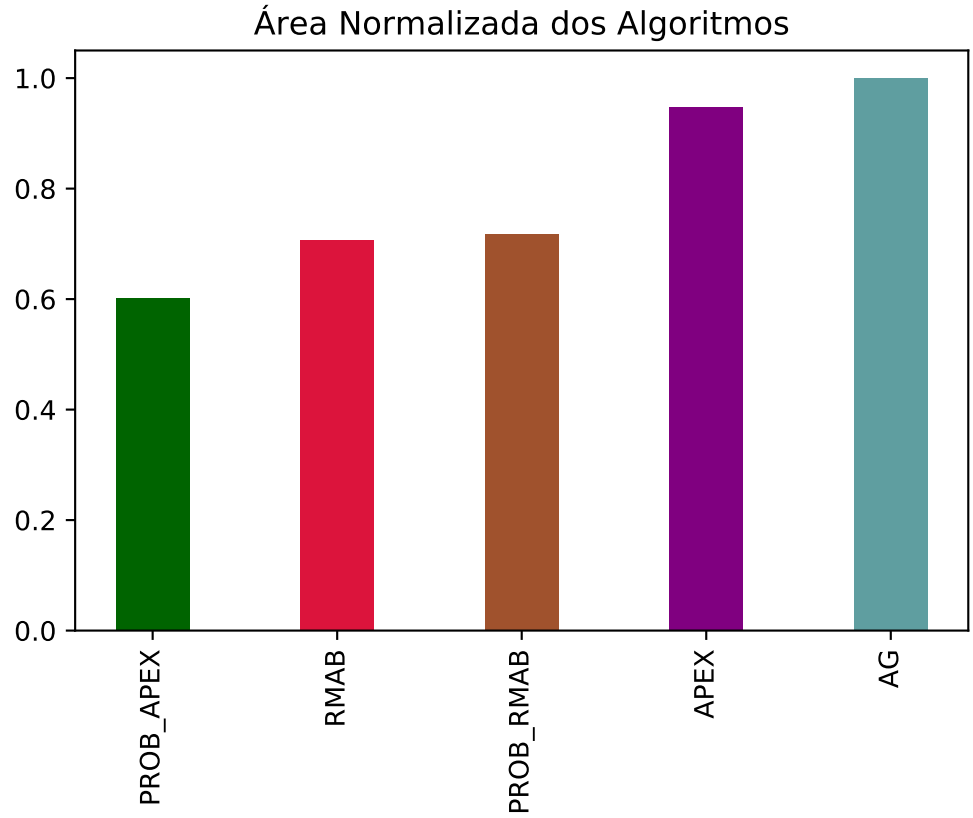


Figura 6.43: Área Normalizada dos PDs para o problema 20_15.

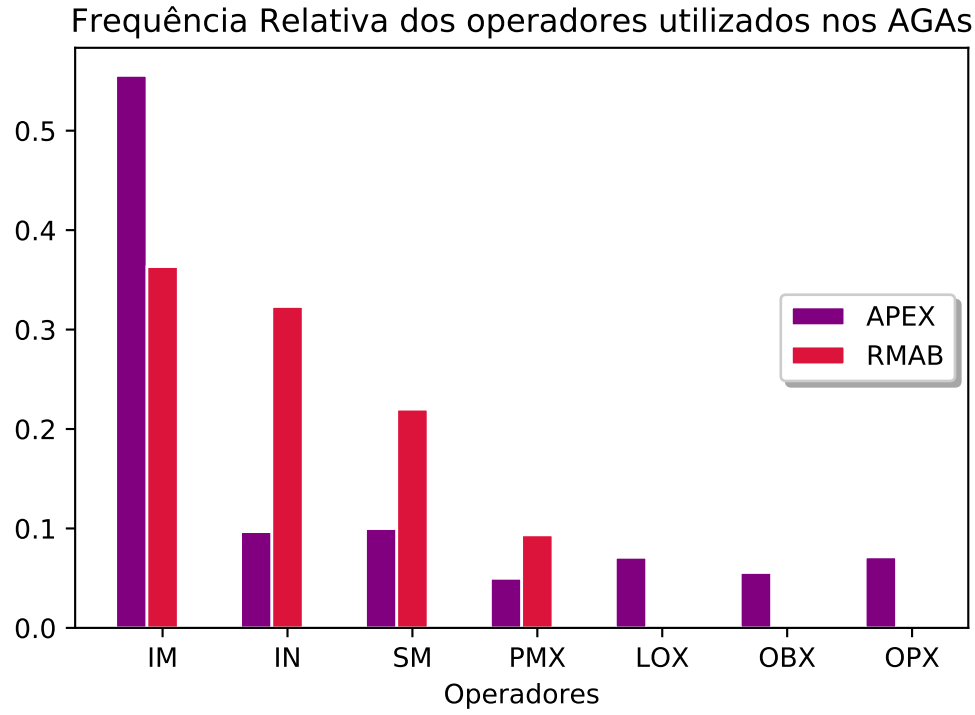


Figura 6.44: Frequência Relativa dos Operadores utilizados nos AGAs para o problema 20_15.

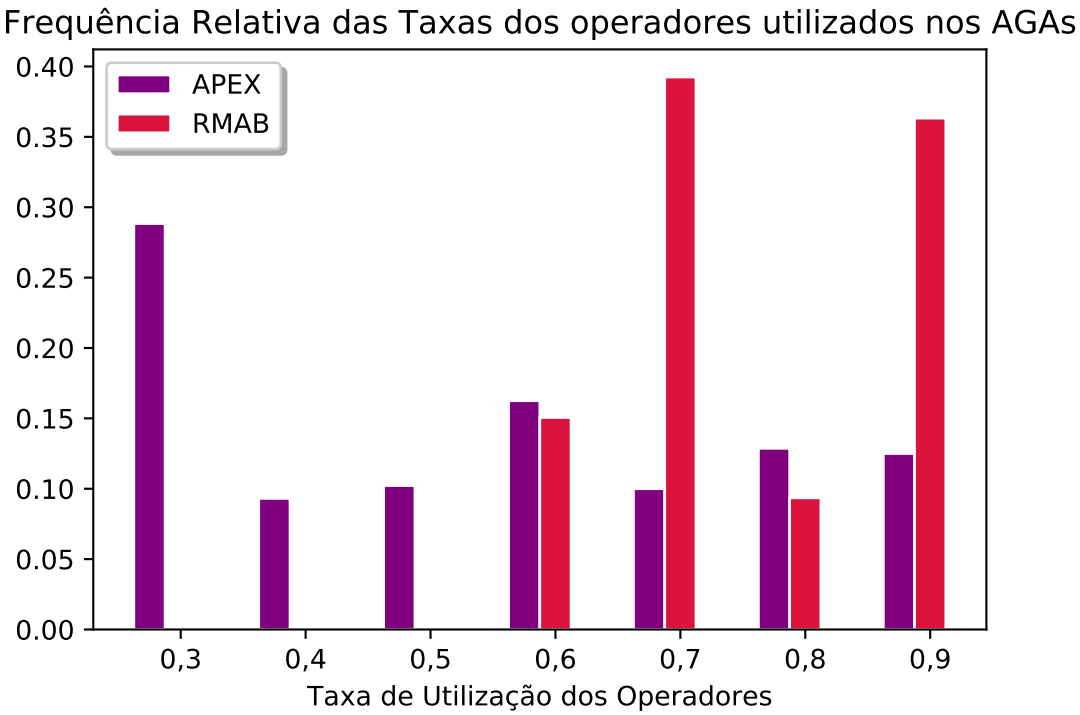


Figura 6.45: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 20_15.

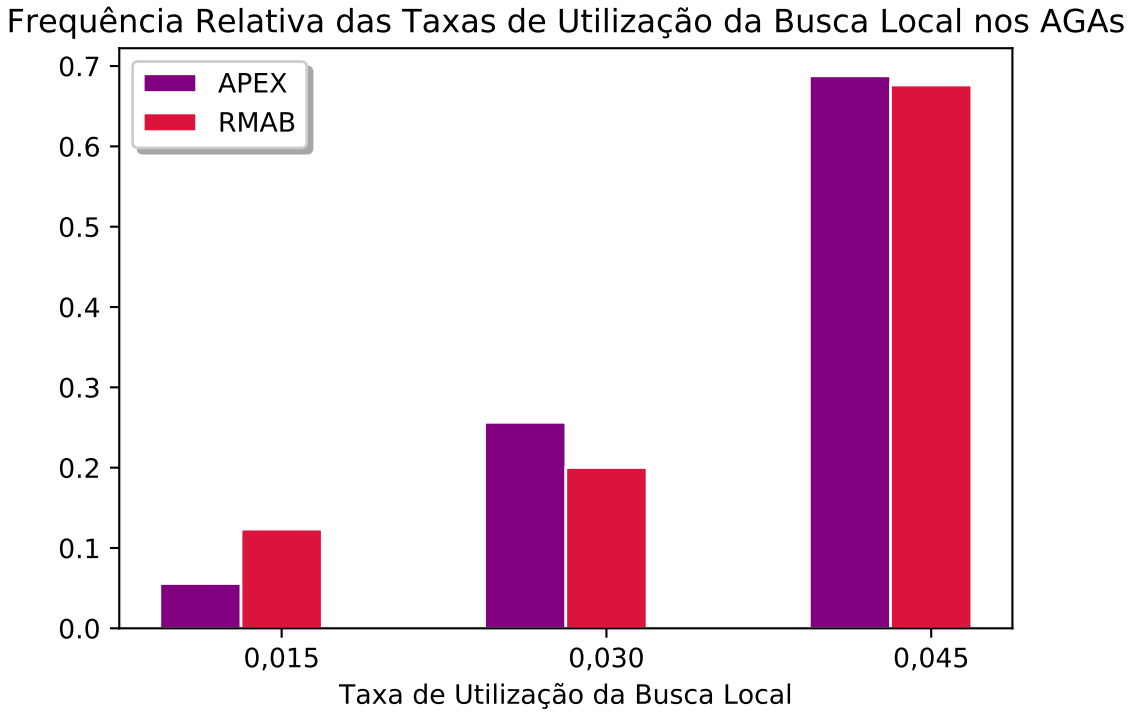


Figura 6.46: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 20_15.

não observando-se então, diferença estatística entre os desempenhos dos algoritmos.

### 6.3.4 20 máquinas e 20 *jobs*

Similar ao problema de 20_15, o AG e o APEX obtiveram comportamentos próximos, enquanto os outros algoritmos tiveram área abaixo da curva normalizada menores que 80%.

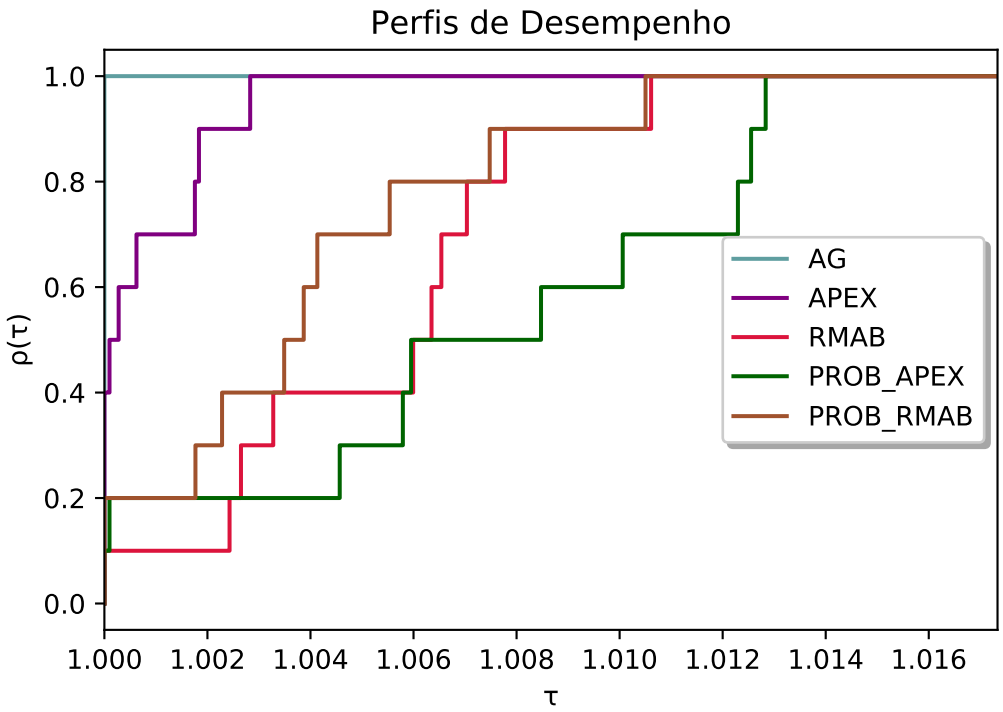


Figura 6.47: Perfis de Desempenho para o problema 20_20.

Considerando os operadores, o RMAB utilizou os operadores IM e IN na maioria das vezes e não teve frequência relativa significativa nos operadores de recombinação LOX, OBX e OPX. O APEX utilizou todos os operadores com frequências similares. Para as taxas de utilização dos operadores, o RMAB não obteve valores significativos para as taxas abaixo de 0,5, sendo as taxas 0,7 e 0,9 as com maior utilização observada. O APEX teve a maior frequência com a taxa 0,6 e o restante com valores entre 0,05 e 0,15.

Nas taxas de utilização da busca local, a maior frequência observada para os dois algoritmos foi na taxa de 0,045 com aproximadamente 70% de utilização. O p-valor desta classe de problemas é 0,88, o que permite a afirmação estatística da não diferença entre o desempenho dos algoritmos.

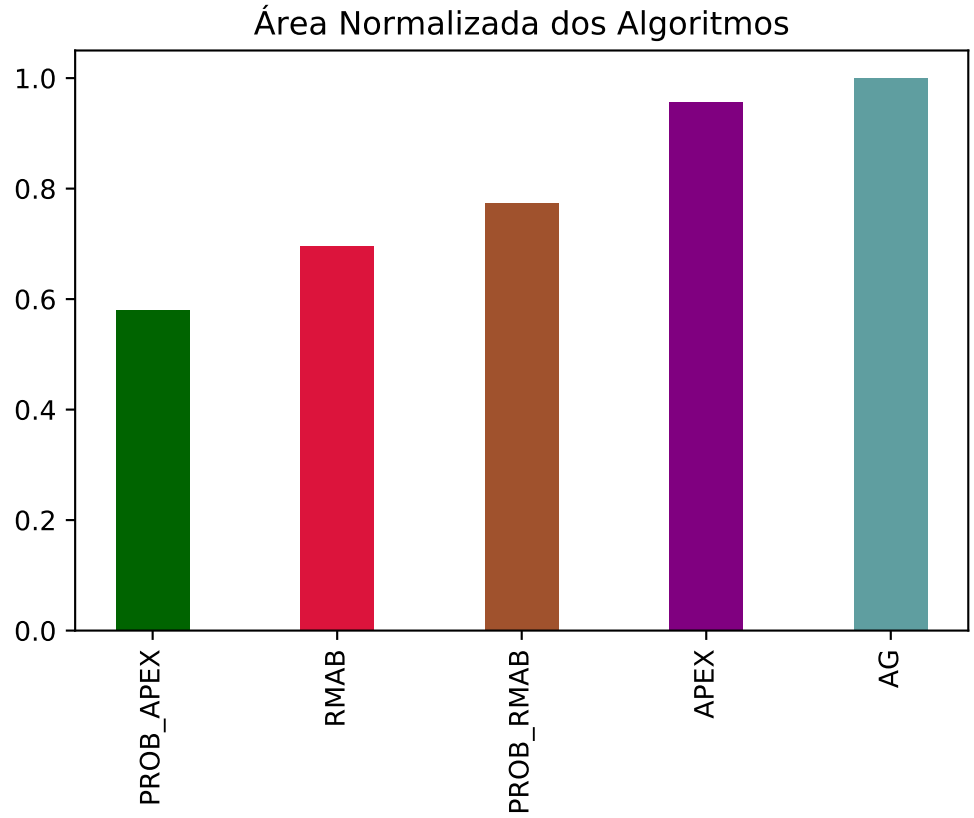


Figura 6.48: Área Normalizada dos PDs para o problema 20_20.

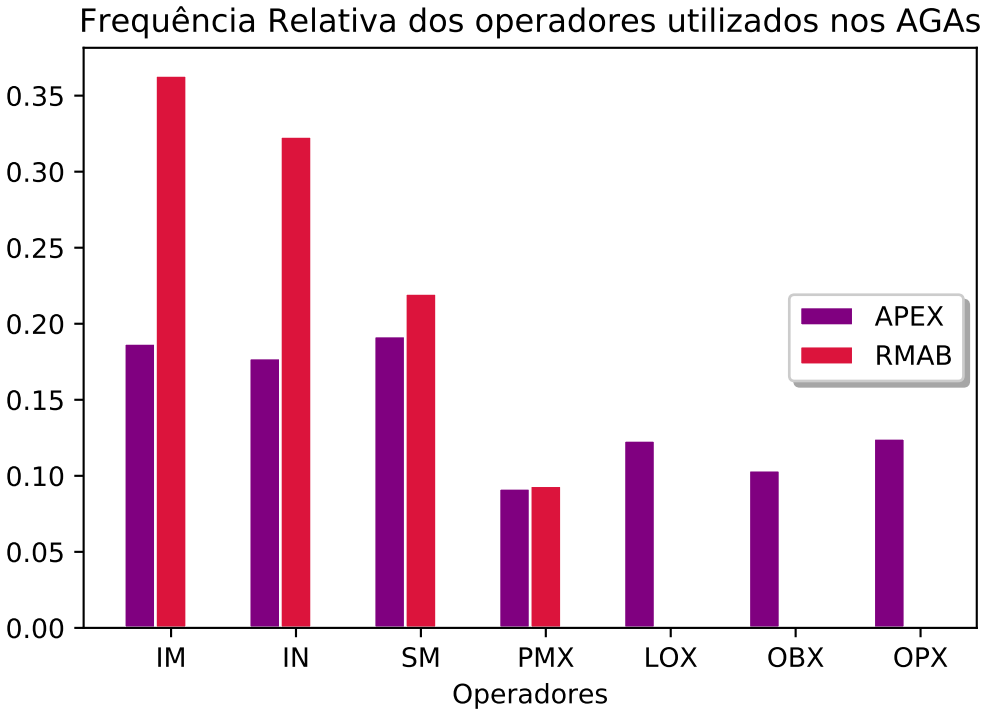


Figura 6.49: Frequência Relativa dos Operadores utilizados nos AGAs para o problema 20_20.

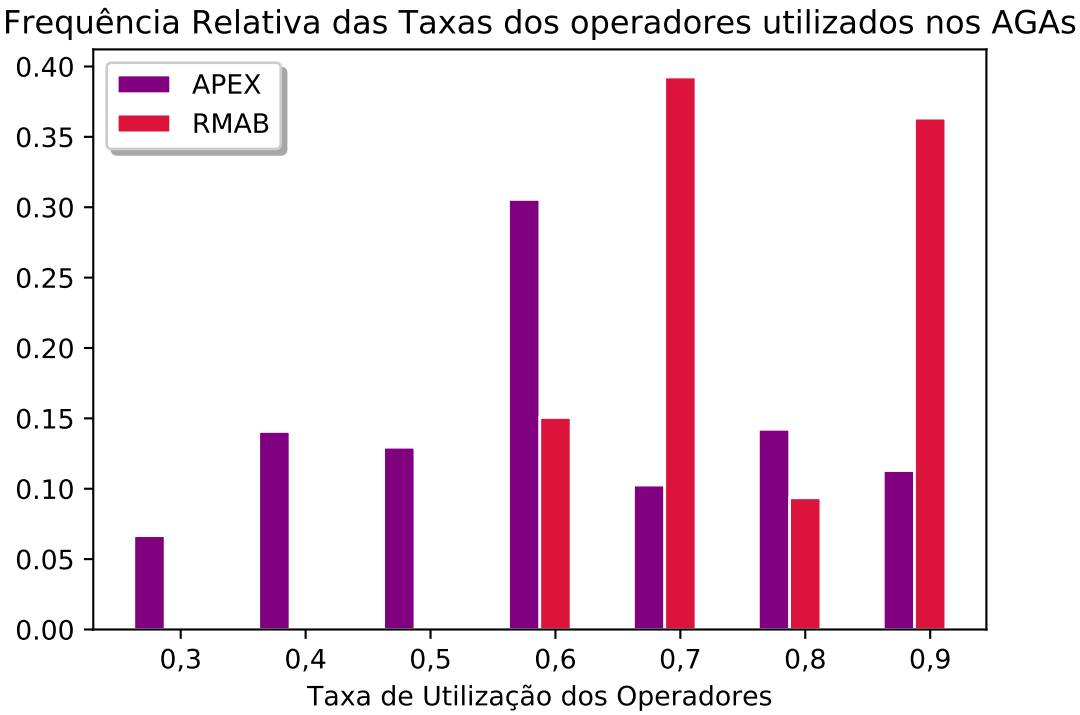


Figura 6.50: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 20_20.

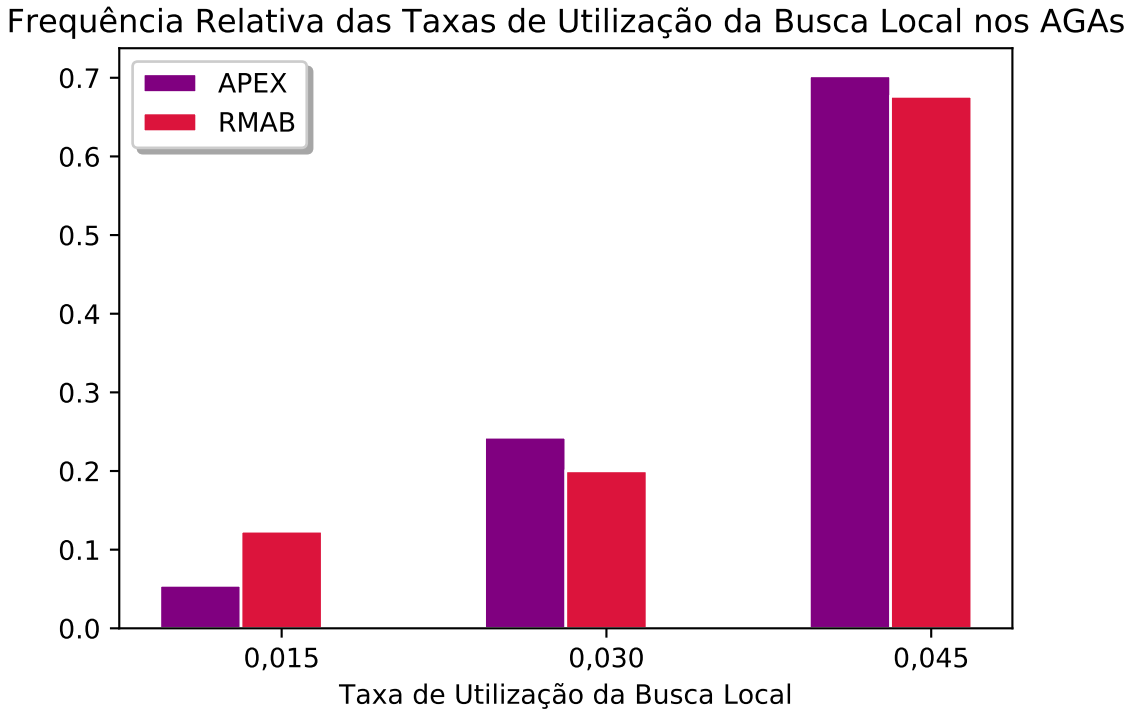


Figura 6.51: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 20_20.

### 6.3.5 30 máquinas e 15 *jobs*

Para este problema, com exceção do PROB_APEX que obteve o pior desempenho, o restante dos algoritmos tiveram desempenho muito similar. O APEX ficou mais próximo do AG, seguido do PROB_RMAB e do RMAB.

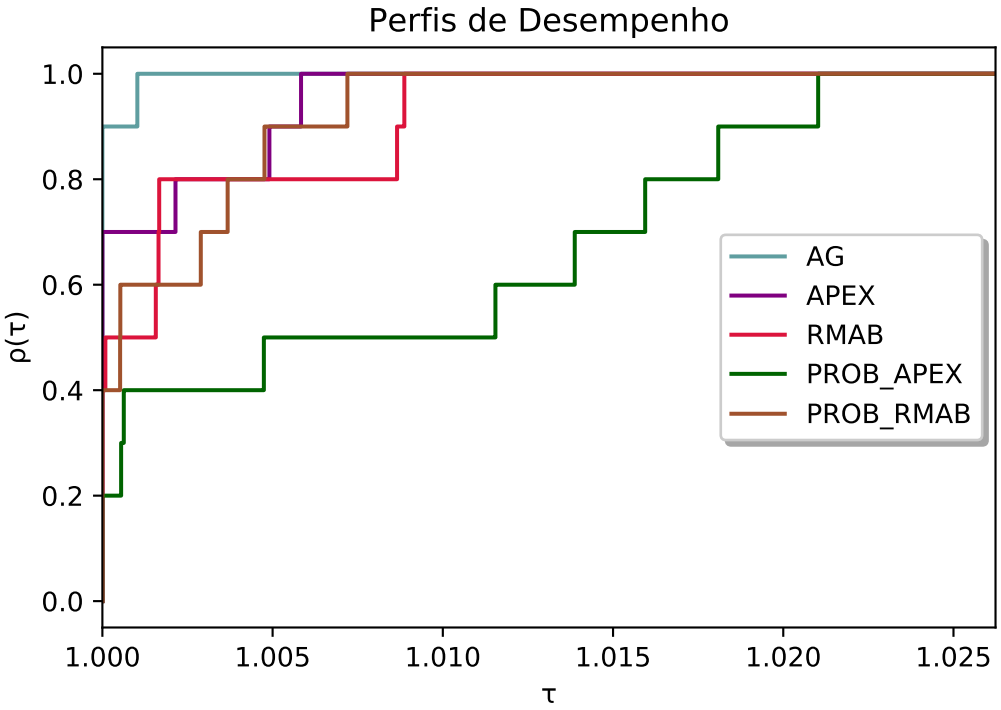


Figura 6.52: Perfis de Desempenho para o problema 30_15.

A frequência dos operadores ficou bem distribuída para o APEX, com nenhum operador superando os 20% de utilização. Já o RMAB, com o operador IN acumula mais de 70% da utilização dos operadores. O restante é preenchido com os operadores IM e OPX. Nas taxas de utilização nenhum dos algoritmos acumulou mais de 30% em cada taxa. Para as taxas 0,5, 0,7 e 0,8 o RMAB não acumulou frequências significativas. Para a taxa 0,015 de busca local, o RMAB não obteve frequência relativa significativa, acumulando o maior valor para a taxa de 0,030. O APEX, com 70% de utilização na taxa de 0,45, distribuiu o restante em 25% e 5% para as taxas de 0,030 e 0,015 respectivamente.

O p-valor encontrado para esta configuração de problema foi 0,96. É possível afirmar que não existe diferença estatística entre os resultados dos algoritmos.

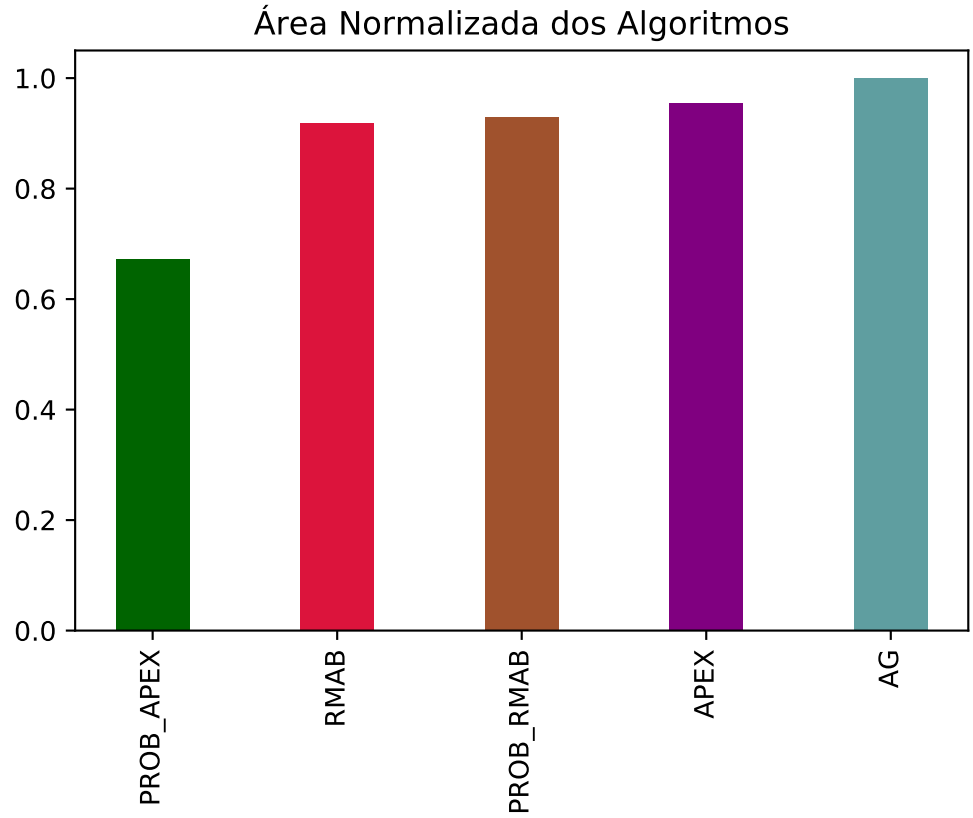


Figura 6.53: Área Normalizada dos PDs para o problema 30_15.

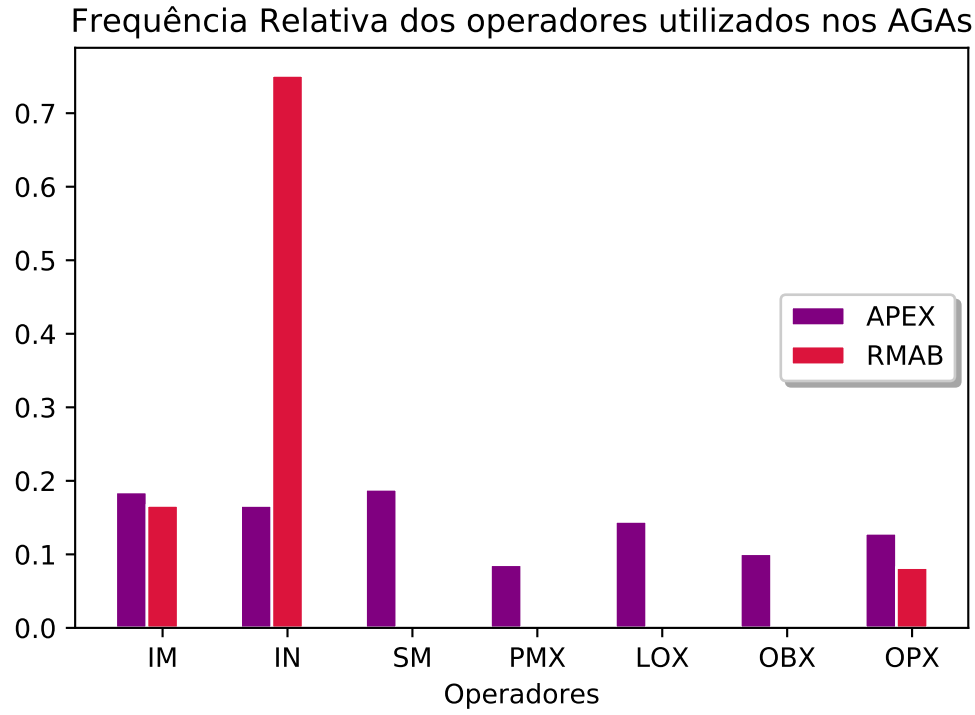


Figura 6.54: Frequência Relativa dos Operadores utilizados nos AGAs para o problema 30_15.

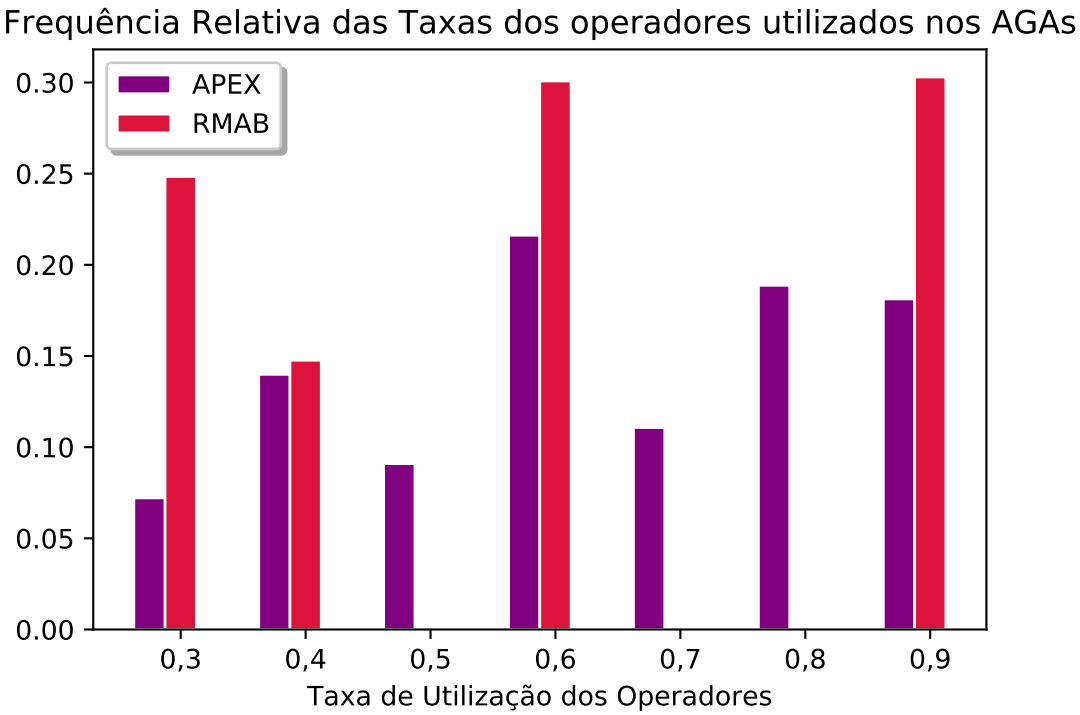


Figura 6.55: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 30_15.

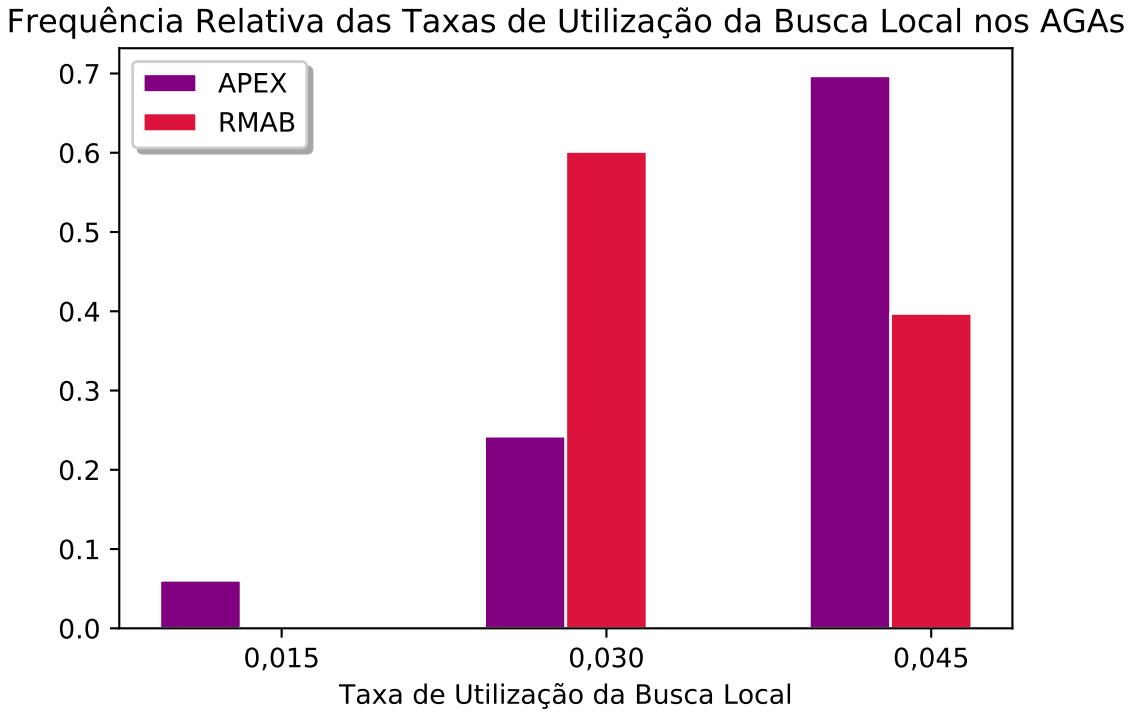


Figura 6.56: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 30_15.

### 6.3.6 30 máquinas e 20 *jobs*

Nesta configuração houve mudança de comportamento comparado ao problema de 30_15: o algoritmo PROB_RMAB obteve o melhor desempenho, seguido do RMAB e do APEX.

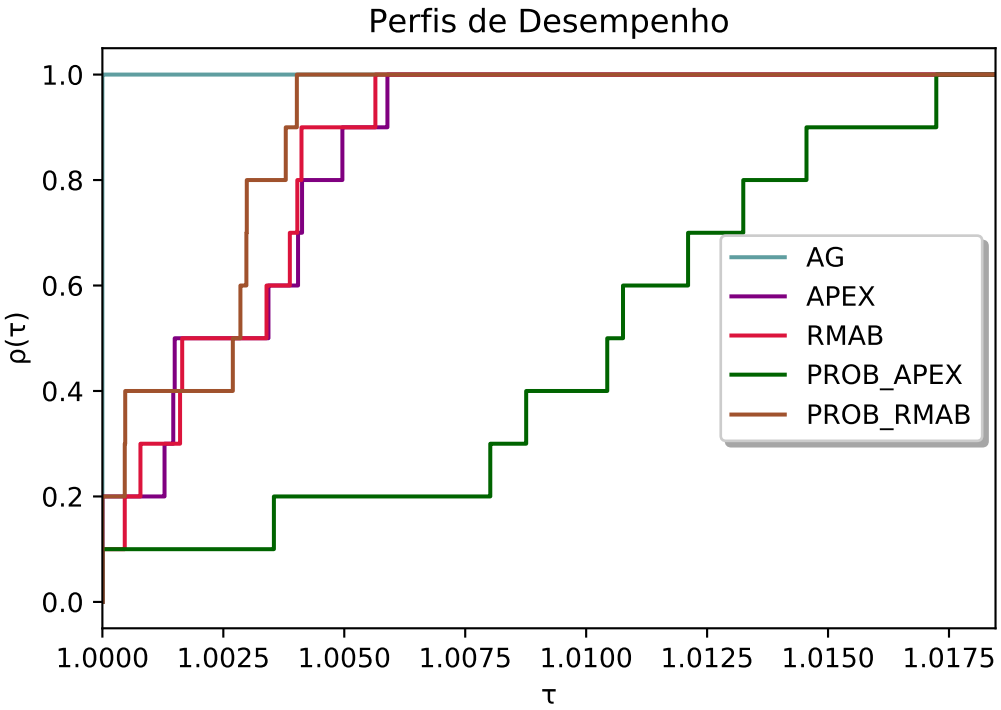


Figura 6.57: Perfis de Desempenho para o problema 30_20.

As frequências de utilização dos operadores e das taxas de utilização se comportaram de maneira similar ao problema 30_15, onde o operador IN foi utilizado mais de 70% pelo RMAB e no APEX todos os operadores não acumularam mais de 20% de utilização cada.

As taxas de busca local que obtiveram maior utilização foram as de 0,045 pelo APEX e a de 0,030 pelo RMAB. A taxa 0,015 não obteve utilizações significativas pelo RMAB.

O p-valor encontrado para este tipo de problema foi de 0,87. É valida a afirmação que os algoritmos não têm diferença estatística de desempenho.

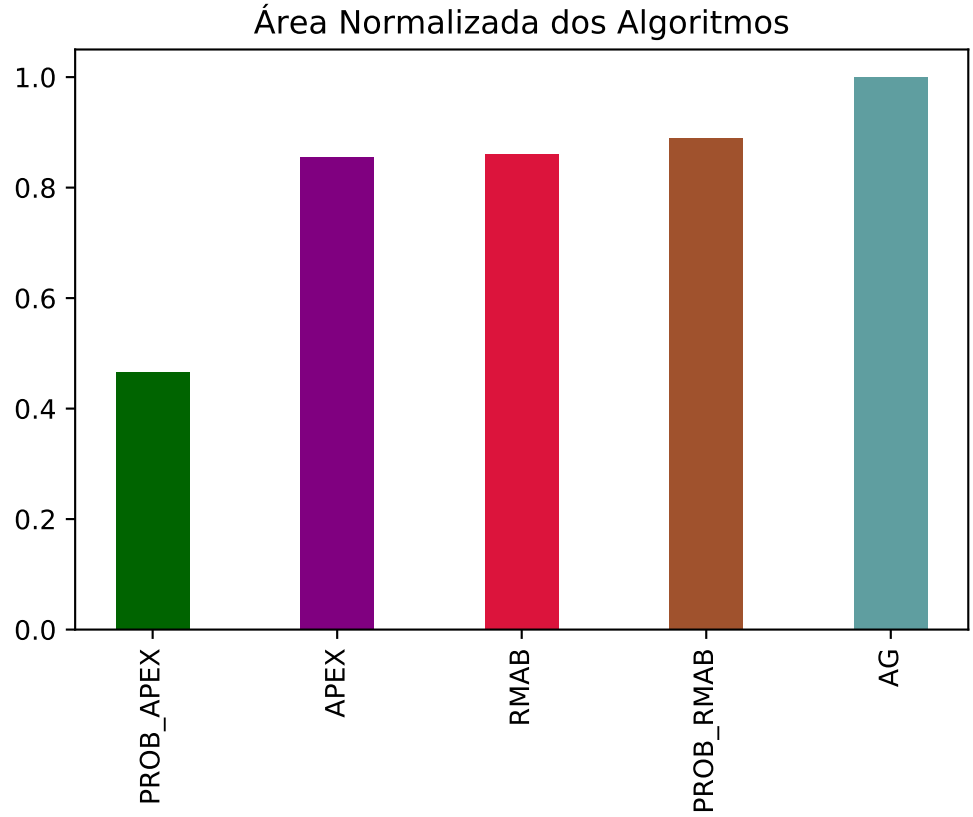


Figura 6.58: Área Normalizada dos PDs para o problema 30_20.

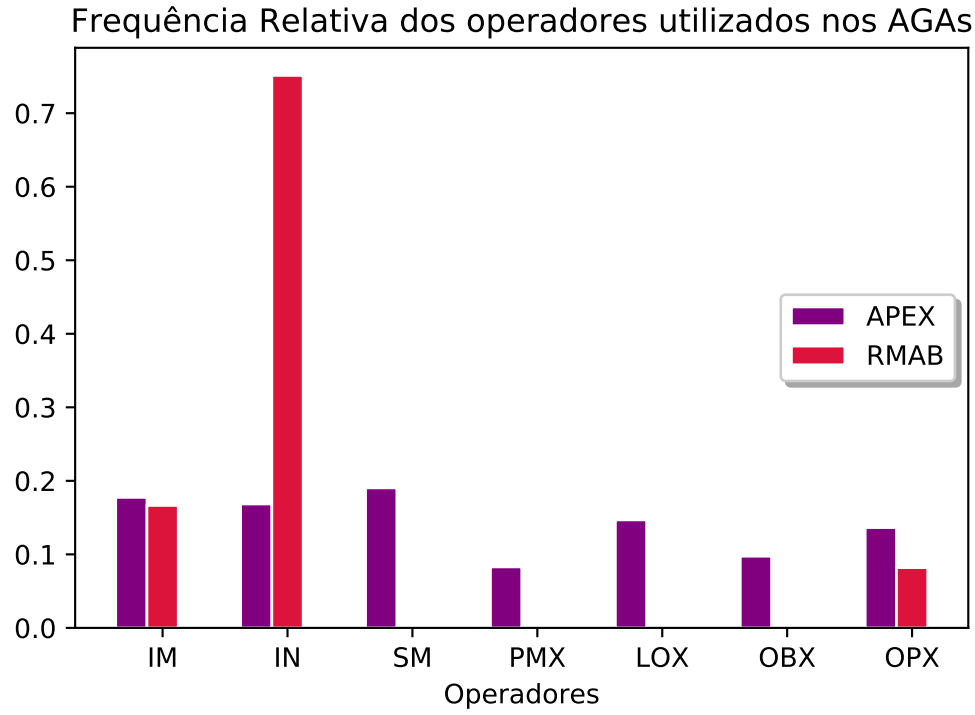


Figura 6.59: Frequência Relativa dos Operadores utilizados nos AGAs para o problema 30_20.

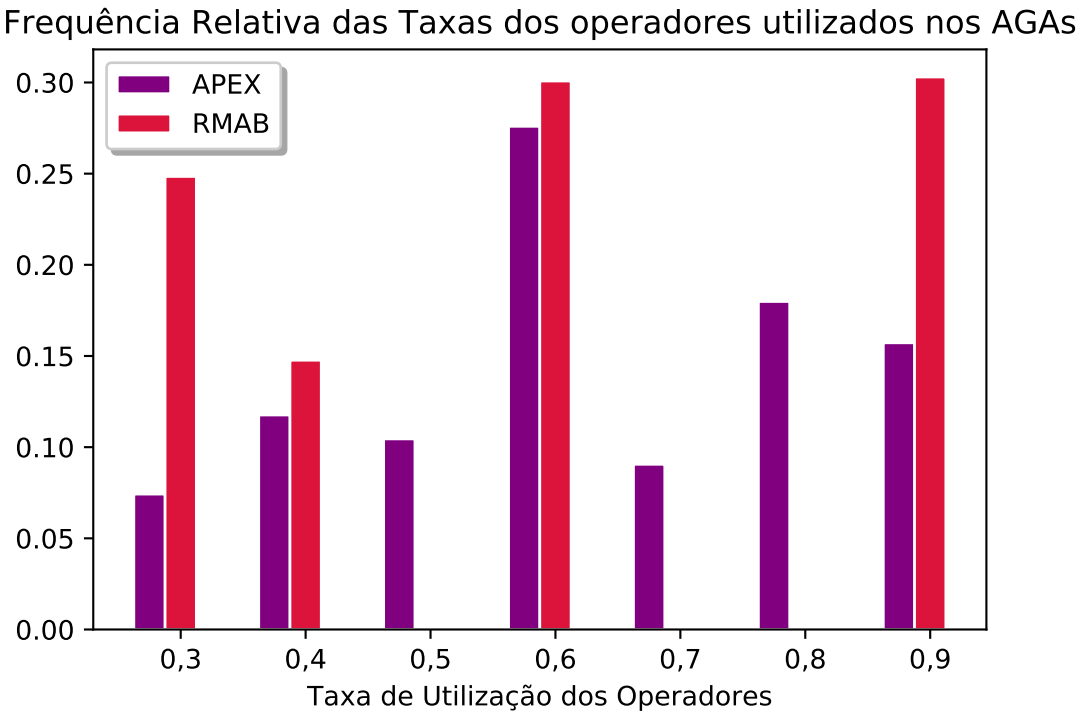


Figura 6.60: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 30_20.

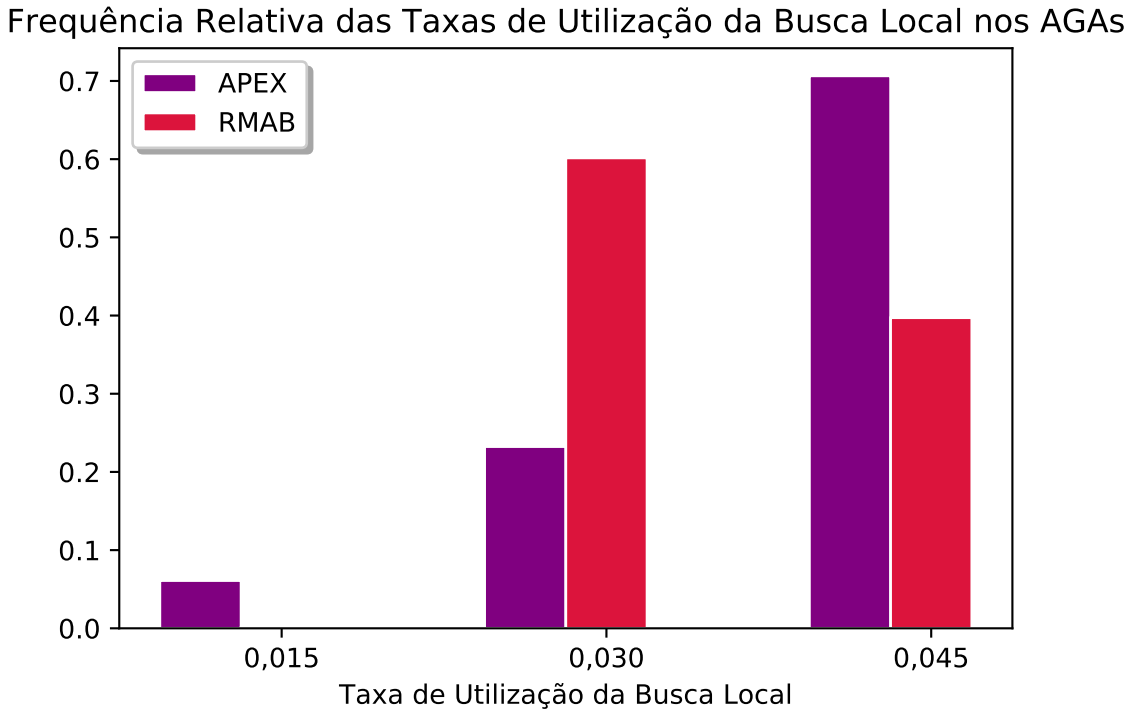


Figura 6.61: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 30_20.

### 6.3.7 50 máquinas e 15 *jobs*

Semelhante ao problema de 30_20, o PROB_RMAB obteve o segundo melhor desempenho, ficando atrás somente do AG. Com a menor área normalizada, o APEX ficou com o pior desempenho relativo nesta classe de problemas.

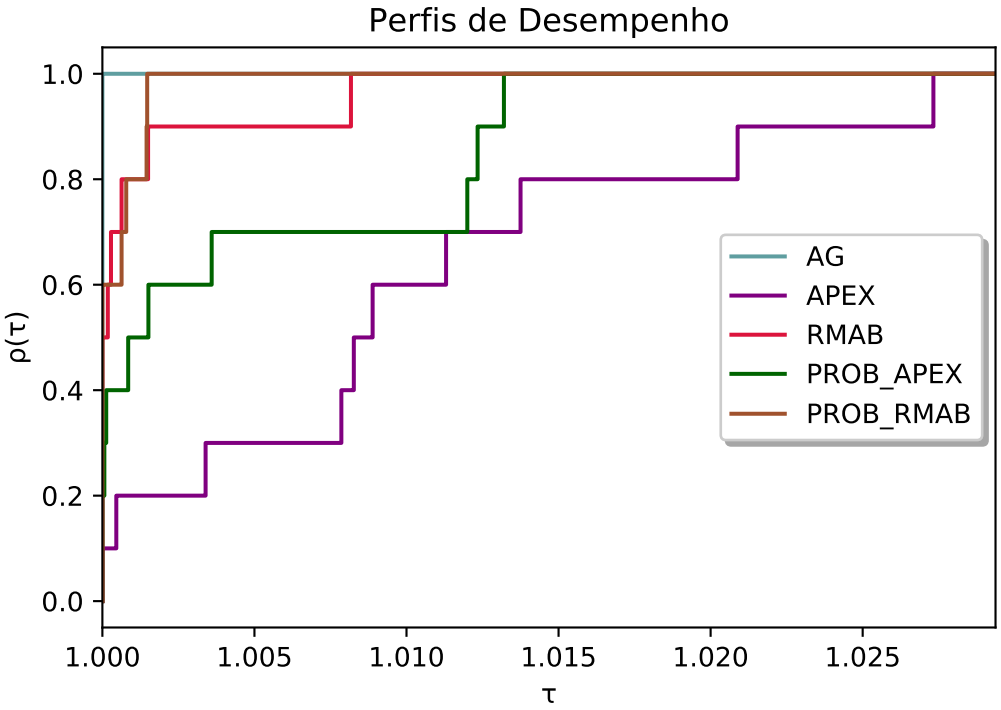


Figura 6.62: Perfis de Desempenho para o problema 50_15.

Considerando os operadores utilizados neste problema, o RMAB possui frequências relativas significativas somente para o IM e o IN. O APEX utiliza o operador IM em mais de 60% dos casos e distribui o restante para todos os outros operadores. As taxas de utilização dos operadores ficam mais concentradas em 0,3 para o APEX e 0,6 para o RMAB, sendo que o APEX distribui o restante para as outras taxas e o RMAB concentra o restante nas taxas de 0,5, 0,7 e 0,9. Para as taxas de utilização da busca local, o RMAB concentra 80% de utilização em 0,045 e não tem valores significativos para 0,015. O APEX tem aproximadamente 70% para a taxa de 0,045 e 20% para a taxa de 0,30. O p-valor encontrado para este problema é 0,93, o que permite a afirmação que os algoritmos não têm diferença estatística.

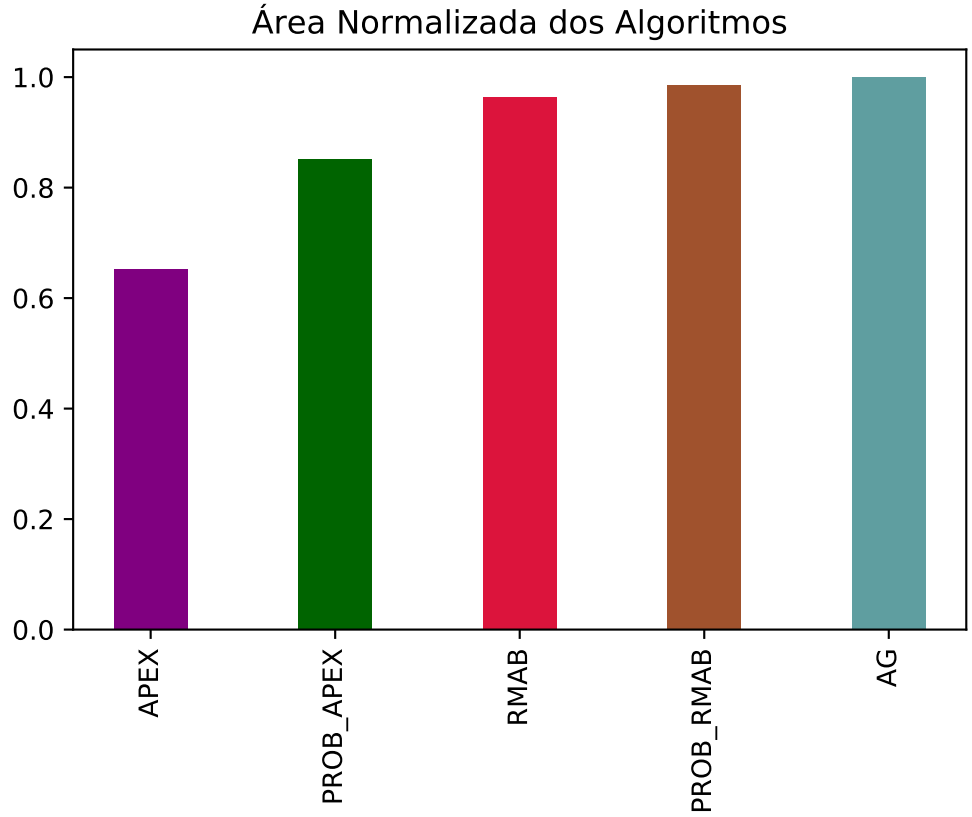


Figura 6.63: Área Normalizada dos PDs para o problema 50_15.

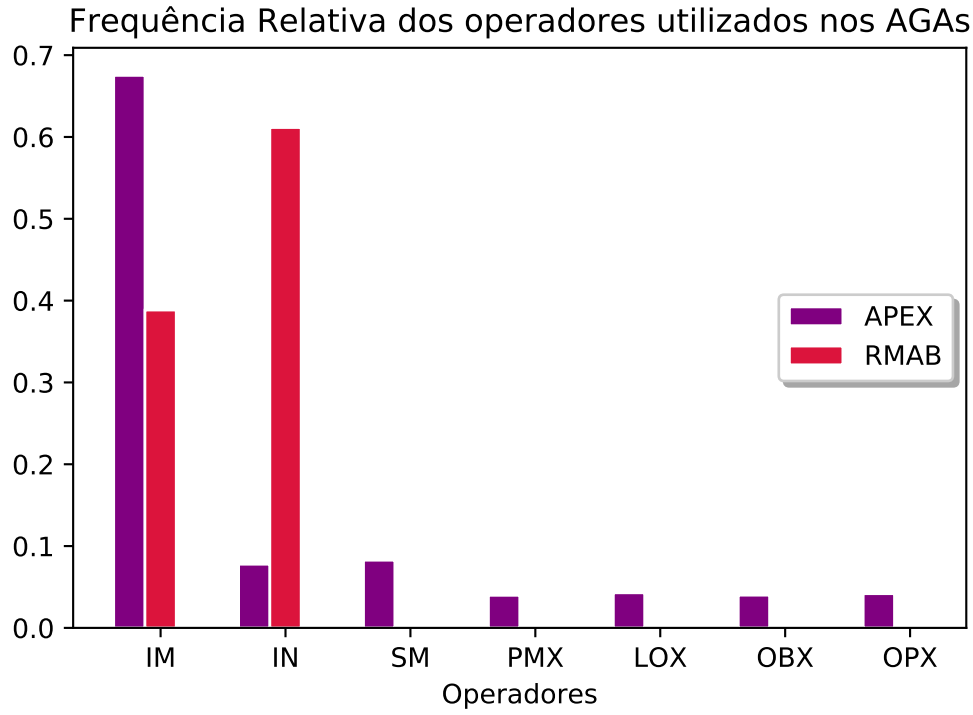


Figura 6.64: Frequência Relativa dos Operadores utilizados nos AGAs para o problema 50_15.

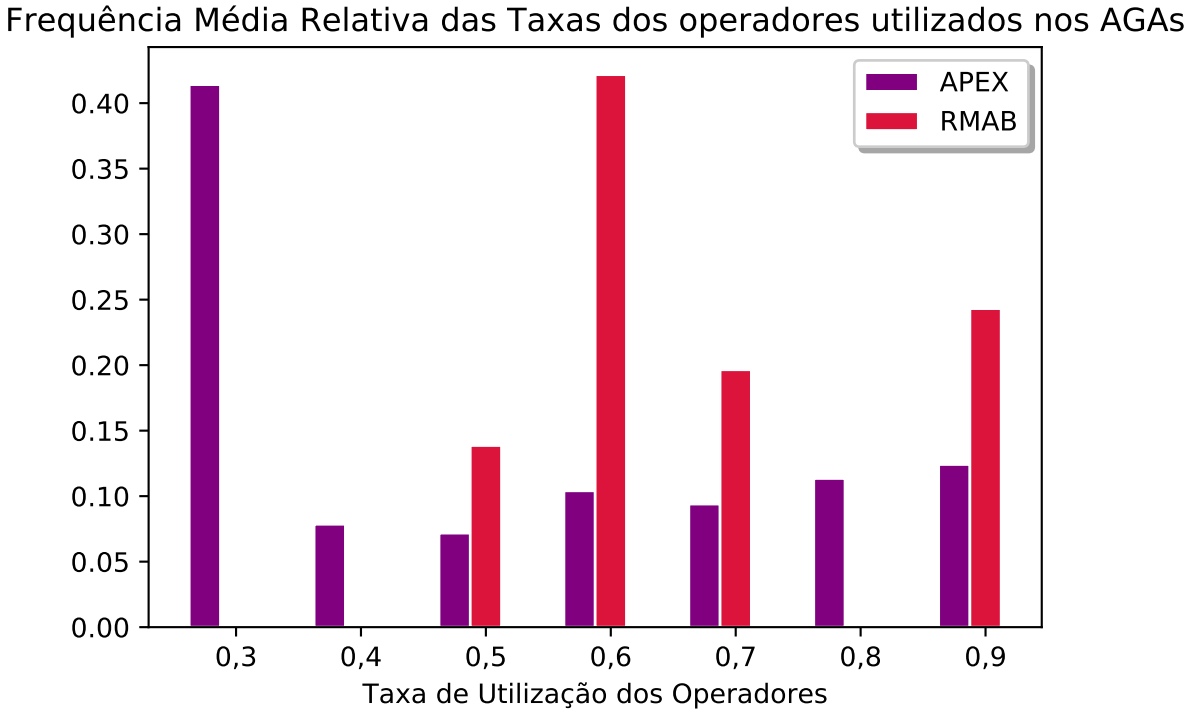


Figura 6.65: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para o problema 50_15.

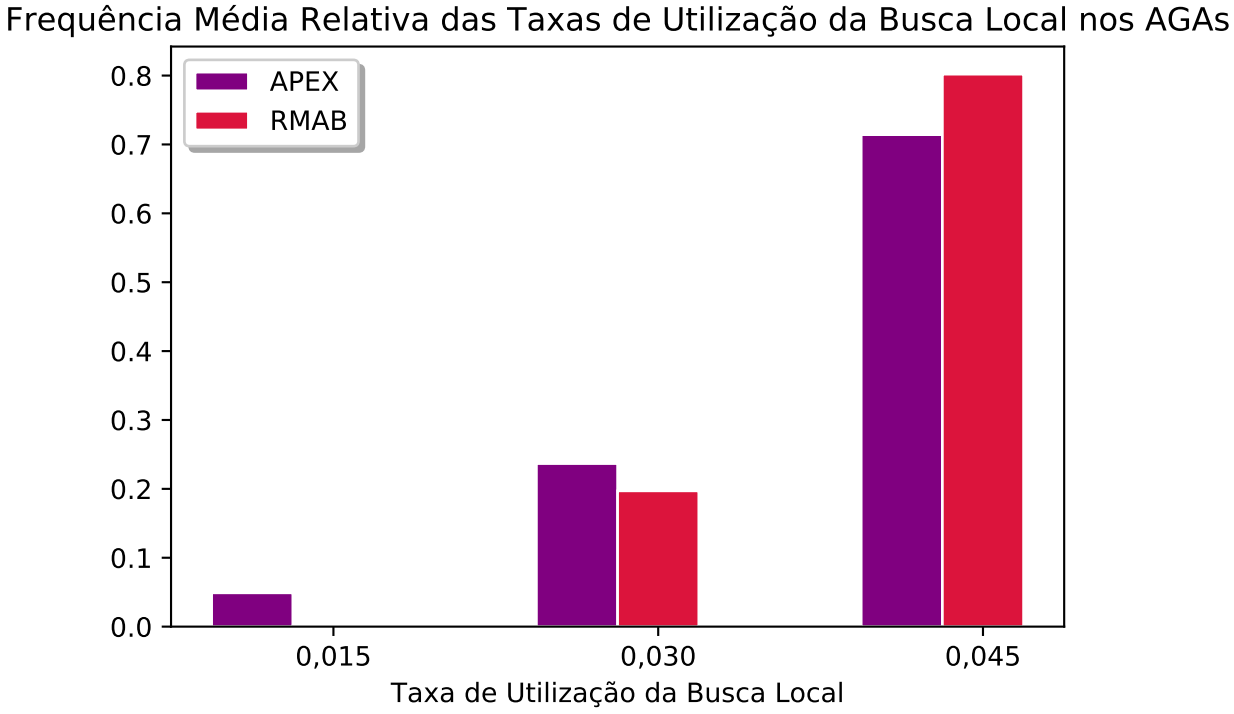


Figura 6.66: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para o problema 50_15.

### 6.3.8 Todos os problemas

Como observado na maioria dos problemas, o APEX obteve o segundo melhor desempenho, ficando somente atrás do AG.

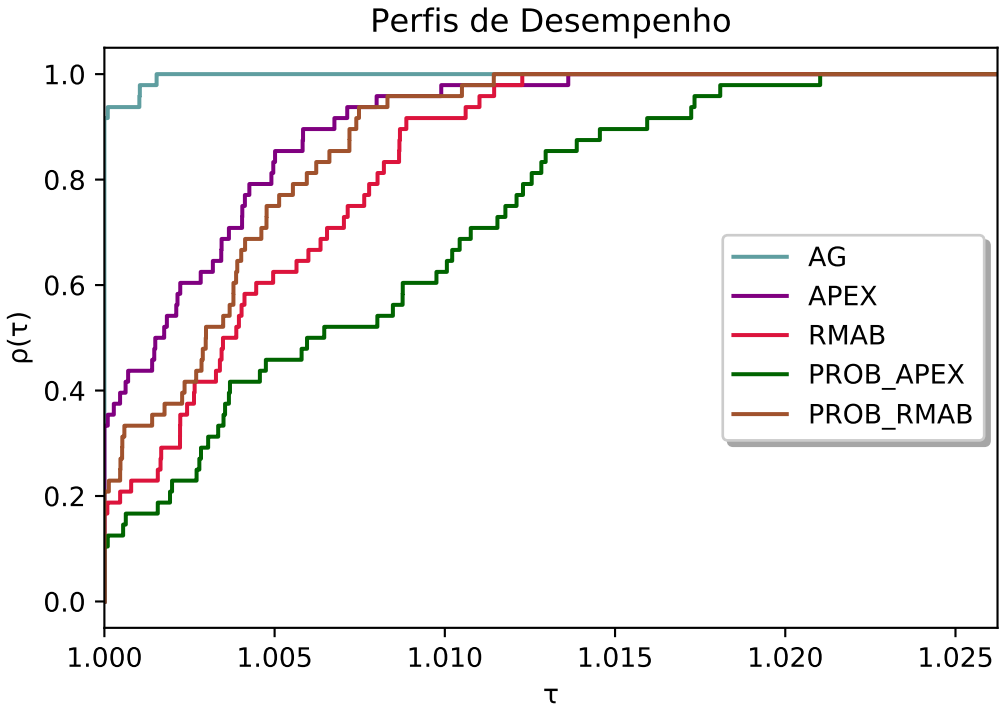


Figura 6.67: Perfis de Desempenho para todos os problemas .

Os operadores mais utilizados por ambos os algoritmos foram os de mutação (IM, IN e SM), sendo o IM o mais utilizado pelo APEX e o IN pelo RMAB. Considerando as taxas de utilização dos operadores, o APEX utiliza, em sua maioria, a menor taxa (0,3), enquanto o RMAB acumulou mais utilizações nas maiores taxas (0,6, 0,7 e,09).

Considerando as taxas de utilização da busca local, o comportamento visto anteriormente é mantido, com a taxa de 0,045 sendo a mais frequente nos dois algoritmos.

O p-valor considerando todos os problemas foi de aproximadamente 1,0. Logo, podemos afirmar que não existe diferença estatística entre os algoritmos quando considerados todos os problemas.

### 6.3.9 Análise do Experimento 2

Com comportamento similar ao Experimento 1, o APEX, quando considerado todos os problemas, foi o AGA que obteve valores próximos ao melhor AG configurado. Da mesma

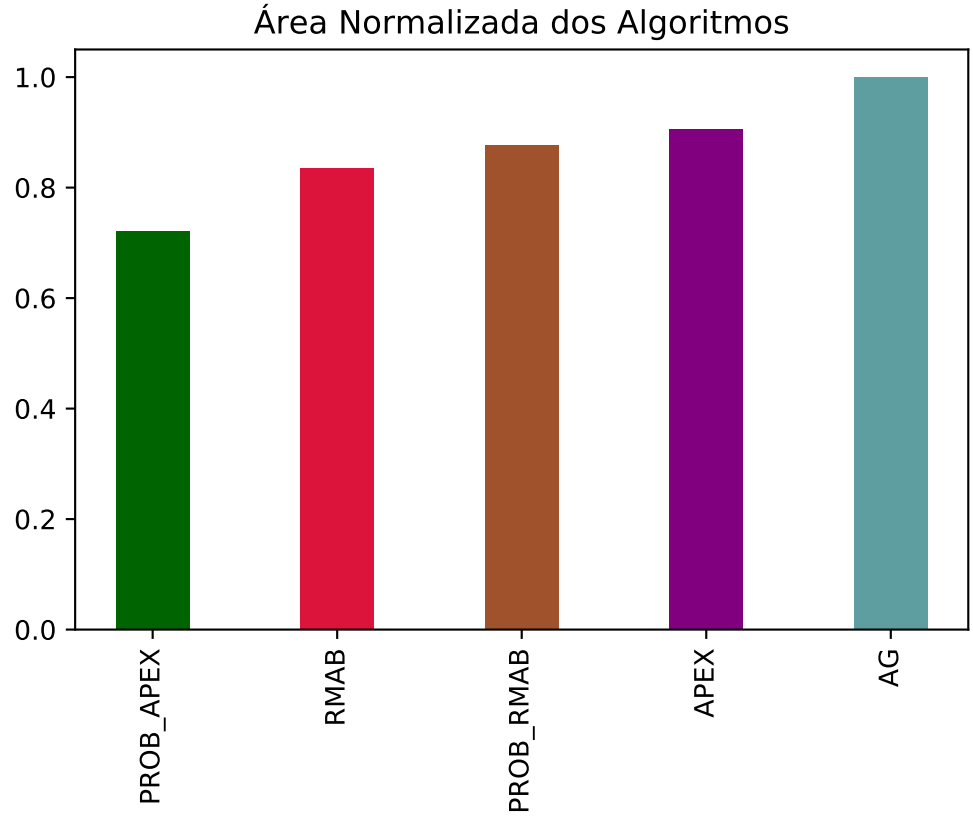


Figura 6.68: Área Normalizada dos PDs para todos os problemas .

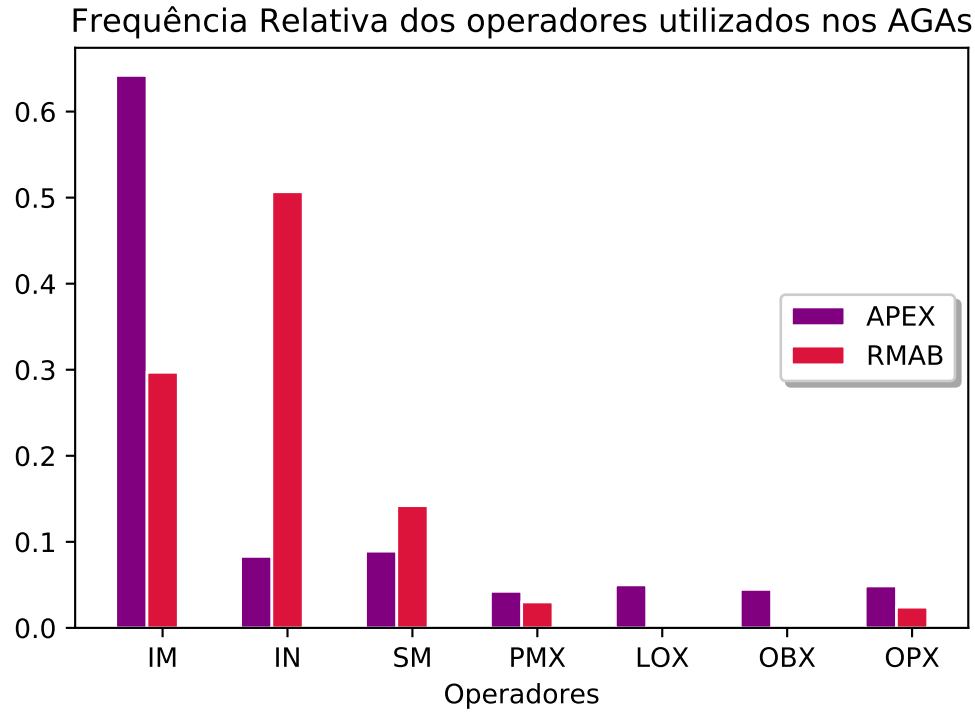


Figura 6.69: Frequência Relativa dos Operadores utilizados nos AGAs para todos os problemas.

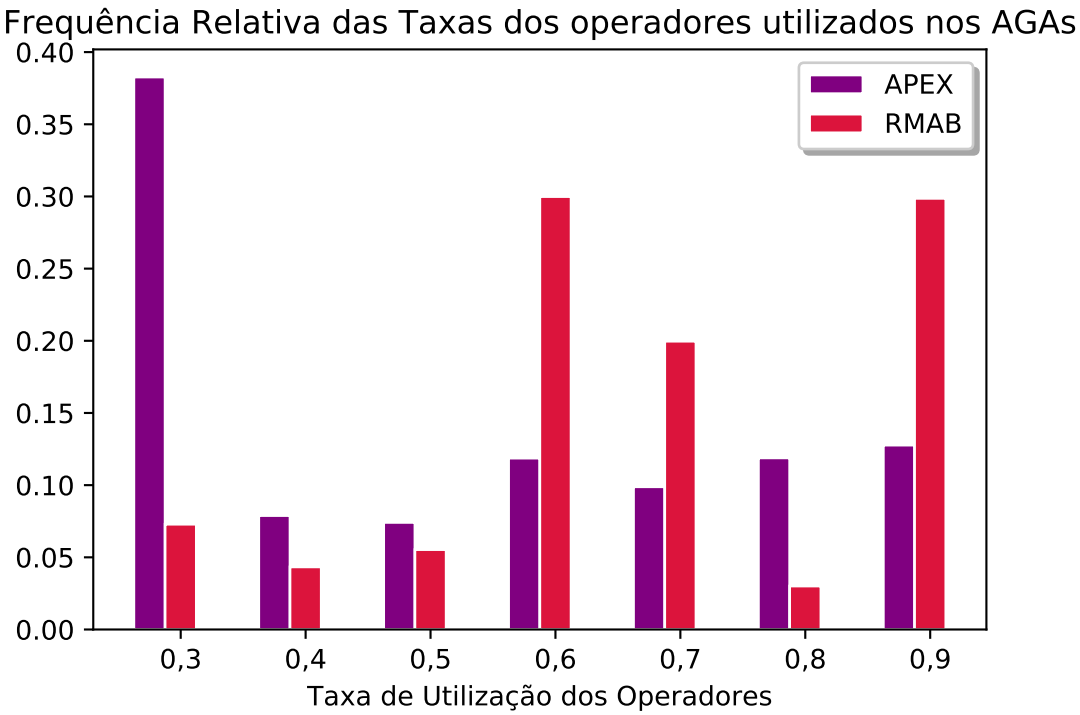


Figura 6.70: Frequência Relativa das Taxas de Utilização dos Operadores utilizados nos AGAs para todos os problemas.

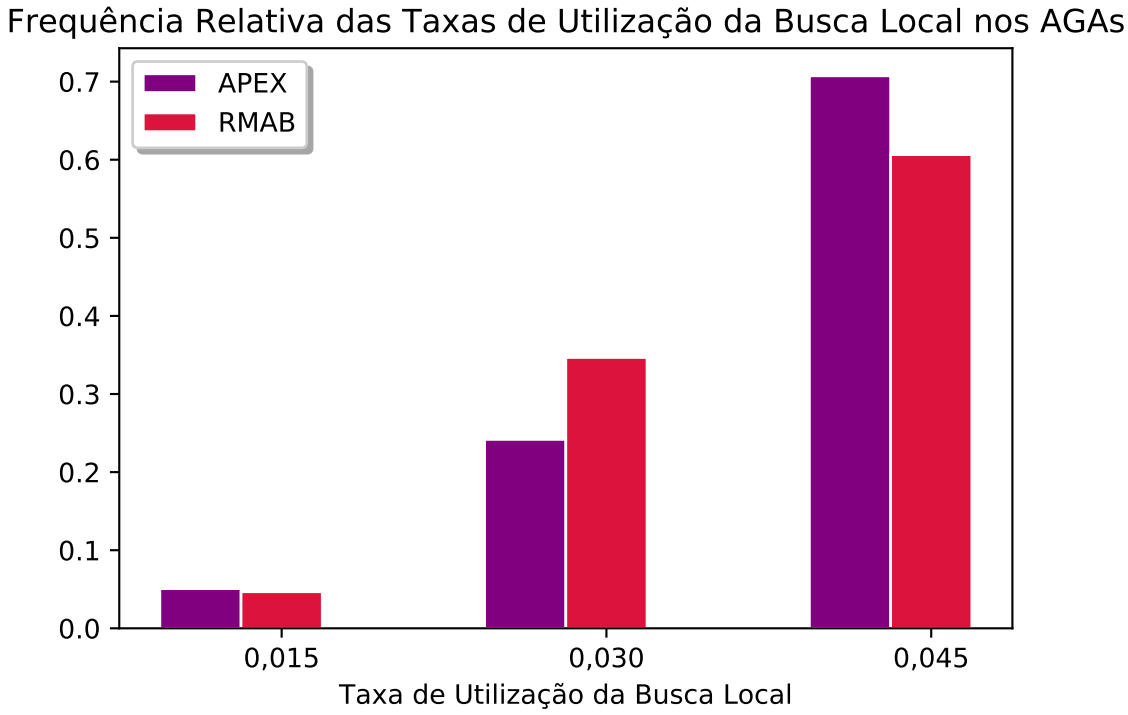


Figura 6.71: Frequência Relativa das Taxas de Utilização da Busca Local utilizados nos AGAs para todos os problemas.

forma, os maiores valores de $\tau$ foram observados próximos de 1,02, o que ressalta a proximidade numérica das diferenças entre os algoritmos. Como em nenhum caso o p-valor foi inferior à 0,05, não existem diferenças estatísticas entre as soluções considerando todos os casos analisados. O RMAB gerou melhores distribuições de frequências, fazendo com que o PROB_RMAB obtivesse melhores soluções comparado ao PROB_APEX.

Foram observadas maiores utilizações dos operadores de mutação, com as maiores taxas de utilização dadas pelo RMAB e o APEX com menores taxas de utilização. As buscas locais, nos dois algoritmos foram utilizadas com as maiores taxas em todos os casos.

# 7 Conclusões e Trabalhos Futuros

Foram propostas duas versões adaptativas para Algoritmos Genéticos: APEX e RMAB. Para cada um destes algoritmos, foram geradas frequências de utilização dos operadores e taxas de utilização e busca local que foram usados por um AG que sorteia uma operação, uma taxa de utilização de operador e uma taxa de utilização de busca local (PROB_APEX E PROB_RMAB). Em nenhum dos casos analisados estas propostas obtiveram resultado numericamente superior ao AG configurado manualmente para cada caso. Isto já era esperado, pois foi escolhido o melhor AG entre todas as variações possíveis de parâmetros e operadores combinados (mutação e recombinação).

O processo de se encontrar esse AG é muito exaustivo, dado que existiam 4 parâmetros e 2 operadores para serem escolhidos (tamanho da população, taxa de recombinação, taxa de mutação, taxa de busca local, operador de recombinação e operador de mutação). Em contrapartida, os AGAs necessitavam apenas da escolha de 2 hiper-parâmetros cada: para o APEX, $\alpha$ e $\beta$; e para o RMAB, $C$ e $D$. Dada a não existência de diferença estatística entre os resultados de todos os algoritmos em todos os experimentos, a escolha em adotar esquemas de adaptação de parâmetros em Algoritmos Genéticos para o problema de Escalonamento *Job-Shop* diminui o tempo na configuração dos melhores hiper-parâmetros definidos pelo usuário e produz soluções de qualidade.

A opção de utilização de uma operação de movimento nos AGAs e nos AGs probabilísticos demonstrou que neste problema, os operadores de mutação aliados à maiores taxas de busca local foram os mais importantes para a geração de soluções de qualidade, sendo que os operadores de recombinação LOX e OBX em todos os casos não obtiveram frequências de utilização relevantes.

Como propostas de trabalhos futuros, podem-se utilizar outros esquemas de adaptação de parâmetros, como o *Predictive Parameter Control*, *Dynamic Multi-Armed Bandit* e *Sliding Multi-Armed Bandit* aliados à esquema de ilhas em AGs, onde cada ilha utilizaria formas de adaptação diferentes.

# Bibliografia

ABEDNEGO, L.; HENDRATMO, D. Genetic programming hyper-heuristic for solving dynamic production scheduling problem. In: SYAICHU-ROHMAN, A. et al. (Ed.). *ICEEI*. Bandung, Indonesiaa: IEEE, 2011. p. 1–4. ISBN 978-1-4577-0753-7.

ALETI, A.; MOSER, I. Predictive Parameter Control. In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2011. (GECCO '11), p. 561–568. ISBN 978-1-4503-0557-0. Disponível em: ⟨http://doi.acm.org/10.1145/2001576.2001653⟩.

ALETI, A.; MOSER, I. A Systematic Literature Review of Adaptive Parameter Control Methods for Evolutionary Algorithms. *ACM Comput. Surv.*, v. 49, n. 3, p. 56:1–56:35, out. 2016. ISSN 0360-0300. Disponível em: ⟨http://doi.acm.org/10.1145/2996355⟩.

APPLEGATE, D.; COOK, W. J. A computational study of the job-shop scheduling problem. *INFORMS Journal on Computing*, v. 3, p. 149–156, 05 1991.

ARROYO, J. E. C.; PEREIRA, A. A. de S. A grasp heuristic for the multi-objective permutation flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 55, n. 5-8, p. 741–753, 2011.

ASADZADEH, L. A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Computers & Industrial Engineering*, v. 85, p. 376 – 383, 2015. ISSN 0360-8352. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0360835215001552⟩.

ASADZADEH, L.; ZAMANIFAR, K. An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. *Mathematical and Computer Modelling*, v. 52, n. 11, p. 1957 – 1965, 2010. ISSN 0895-7177. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0895717710002098⟩.

BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. *Evolutionary computation 1: Basic algorithms and operators*. Philadelphia, Pennsylvania: CRC press, 2018.

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, v. 6, n. 2, p. 154–160, 1994.

BIERWIRTH, C. A generalized permutation approach to job shop scheduling with genetic algorithms. *Operations-Research-Spektrum*, v. 17, n. 2-3, 1995. ISSN 0171-6468, 1436-6304.

BILKAY, O.; ANLAGAN, O.; KILIC, S. E. Job shop scheduling using fuzzy logic. *The International Journal of Advanced Manufacturing Technology*, v. 23, n. 7-8, p. 606–619, 2004. ISSN 0268-3768, 1433-3015. Disponível em: ⟨https://link.springer.com/article/10.1007/s00170-003-1771-2⟩.

CHAUDHRY, I. A.; KHAN, A. A. A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, v. 23, n. 3, p. 551–591, 2016. ISSN 1475-3995. Disponível em: ⟨https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12199⟩.

CHEN, H.; IHLOW, J.; LEHMANN, C. A genetic algorithm for flexible job-shop scheduling. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. Detroit, Michigan: IEEE, 1999. v. 2, p. 1120–1125 vol.2.

CHENG, R.; GEN, M.; TSUJIMURA, Y. A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation. *Computers & Industrial Engineering*, v. 30, n. 4, p. 983 – 997, 1996. ISSN 0360-8352. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/0360835296000472⟩.

COSTA, L. D. et al. Adaptive Operator Selection with Dynamic Multi-Armed Bandits. In: . ACM, 2008. p. 913–920. Disponível em: ⟨https://hal.inria.fr/inria-00278542/document⟩.

DOLAN, E. D.; MORé, J. J. Benchmarking optimization software with performance profiles. *Mathematical Programming*, v. 91, n. 2, p. 201–213, jan. 2002. ISSN 1436-4646. Disponível em: ⟨https://doi.org/10.1007/s101070100263⟩.

EIBEN, A. E.; HINTERDING, R.; MICHALEWICZ, Z. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, v. 3, n. 2, p. 124–141, jul. 1999. ISSN 1089-778X.

EIBEN, A. E.; SMITH, J. E. et al. *Introduction to evolutionary computing*. Heidelberg, Berlim: Springer, 2003. v. 53.

FERREIRA, G. de S.; BERNARDINO, H. An adaptive genetic algorithm for solving job-shop scheduling problems. In: . Florianopolis, Santa Catarina: ABMEC, 2017.

FERREIRA, G. S. *Algoritmos Genéticos Adaptativos para Solucionar Problemas de Sequenciamento do Tipo Job-Shop Flexível*. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, 2018.

FERREIRA, G. S.; BERNARDINO, H. S. An adaptive pursuit genetic algorithm for solving job-shop scheduling problems. In: *Congresso Brasileiro de Inteligência Computacional*. Niterói, RJ: ABRICOM, 2017. p. 1–12.

FIALHO, Á. *Adaptive operator selection for optimization*. Tese (Doutorado) — Université Paris Sud-Paris XI, 2010.

FIALHO, A. et al. Extreme Value Based Adaptive Operator Selection. In: *Parallel Problem Solving from Nature – PPSN X*. Springer, Berlin, Heidelberg, 2008. (Lecture Notes in Computer Science), p. 175–184. ISBN 978-3-540-87699-1 978-3-540-87700-4. Disponível em: ⟨https://link.springer.com/chapter/10.1007/978-3-540-87700-4_18⟩.

FIALHO, A.; SCHOENAUER, M.; SEBAG, M. Toward Comparison-based Adaptive Operator Selection. In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2010. (GECCO '10), p. 767–774. ISBN 978-1-4503-0072-8. Disponível em: ⟨http://doi.acm.org/10.1145/1830483.1830619⟩.

FISHER, H. Probabilistic learning combinations of local job-shop scheduling rules. *Industrial scheduling*, Prentice-Hall, p. 225–251, 1963.

GEIGER, C. D.; UZSOY, R. Learning effective dispatching rules for batch processor scheduling. *International Journal of Production Research*, Taylor & Francis, v. 46, n. 6, p. 1431–1454, 2008. Disponível em: ⟨https://doi.org/10.1080/00207540600993360⟩.

GITTINS, J.; GLAZEBROOK, K.; WEBER, R. *Multi-armed bandit allocation indices*. [S.l.]: John Wiley & Sons, 2011.

GITTINS, J. C. et al. *Multi-armed bandit allocation indices*. United Kingdom: Wiley Online Library, 1989. v. 25.

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, 2005. ISSN 0377-2217.

HEINONEN, J.; PETTERSSON, F. Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem. *Applied Mathematics and Computation*, v. 187, n. 2, p. 989–998, 2007. ISSN 0096-3003. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0096300306012355⟩.

HILDEBRANDT, T.; HEGER, J.; SCHOLZ-REITER, B. Towards improved dispatching rules for complex shop floor scenarios: A genetic programming approach. In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2010. (GECCO '10), p. 257–264. ISBN 978-1-4503-0072-8. Disponível em: ⟨http://doi.acm.org/10.1145/1830483.1830530⟩.

HO, N. B.; TAY, J. C. Evolving dispatching rules for solving the flexible job-shop problem. In: IEEE. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*. Edinburgh, Scotland, 2005. v. 3, p. 2848–2855.

HOLLAND, J. *Adaptation in Natural and Artificial Systems*. Ann Arbor, Michigan: University of Michigan Press, 1975.

HOLSAPPLE, C. W. et al. A genetics-based hybrid scheduler for generating static schedules in flexible manufacturing contexts. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 23, n. 4, p. 953–972, Jul 1993. ISSN 0018-9472.

JONES, T. A description of holland's royal road function. *Evolutionary computation*, MIT Press, v. 2, n. 4, p. 409–415, 1994.

KARAFOTIAS, G.; HOOGENDOORN, M.; EIBEN, A. E. Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Transactions on Evolutionary Computation*, v. 19, n. 2, p. 167–187, abr. 2015. ISSN 1089-778X.

KOLISCH, R.; HARTMANN, S. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, v. 174, p. 23–37, 10 2006.

KREMPSER, E.; FIALHO, A.; BARBOSA, H. J. C. Adaptive Operator Selection at the Hyper-level. In: *Parallel Problem Solving from Nature - PPSN XII*. Springer, Berlin, Heidelberg, 2012. (Lecture Notes in Computer Science), p. 378–387. ISBN 978-3-642-32963-0 978-3-642-32964-7. Disponível em: ⟨https://link.springer.com/chapter/10.1007/978-3-642-32964-7_38⟩.

KRUSKAL, W. H.; WALLIS, W. A. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, Taylor & Francis Group, v. 47, n. 260, p. 583–621, 1952.

KURDI, M. A new hybrid island model genetic algorithm for job shop scheduling problem. *Computers & Industrial Engineering*, v. 88, p. 273 – 283, 2015. ISSN 0360-8352. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0360835215003186⟩.

LAWRENCE, S. Resouce constrained project scheduling: An experimental investigation of heuristic scheduling techniques (supplement). *Graduate School of Industrial Administration, Carnegie-Mellon University*, 1984.

LIN, T.-L. et al. An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*, v. 37, n. 3, p. 2629–2636, 2010. ISSN 0957-4174. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0957417409007696⟩.

LVSHAN, Y.; DONGZHI, Y.; WEIYU, Y. Artificial bee colony algorithm with genetic algorithm for job shop scheduling problem. In: *2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. Xiamen, China: IEEE, 2017. p. 433–438.

NAWAZ, M.; ENSCORE, E. E.; HAM, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, v. 11, n. 1, p. 91–95, jan. 1983. ISSN 0305-0483.

NGUYEN, S. et al. A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Trans. Evolutionary Computation*, v. 17, n. 5, p. 621–639, 2013.

NGUYEN, S.; ZHANG, M.; TAN, K. C. Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems. In: *CEC*. Sendai, Japan: IEEE, 2015. p. 2781–2788.

NOWICKI, E.; SMUTNICKI, C. A fast taboo search algorithm for the job shop problem. *Management science*, INFORMS, v. 42, n. 6, p. 797–813, 1996.

ROY, B.; SUSSMANN, B. Les problemes d'ordonnancement avec contraintes disjonctives, 1964. *SEMA, Rapport de recherches*, n. 9, 1964.

RUIZ, R.; STüTZLE, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, v. 177, n. 3, p. 2033 – 2049, 2007. ISSN 0377-2217. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0377221705008507⟩.

SABUNCUOGLU, I.; BAYIZ, M. Job shop scheduling with beam search. *European Journal of Operational Research*, v. 118, n. 2, p. 390–412, 1999. ISSN 0377-2217. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0377221798003191⟩.

SRINIVAS, M.; PATNAIK, L. M. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 24, n. 4, p. 656–667, abr. 1994. ISSN 0018-9472.

SYSWERDA, G. A Study of Reproduction in Generational and Steady-State Genetic Algorithms. In: RAWLINS, G. J. E. (Ed.). Elsevier, 1991, (Foundations of Genetic Algorithms, v. 1). p. 94 – 101. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/B9780080506845500094⟩.

TAILLARD, E. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, Elsevier, v. 64, n. 2, p. 278–285, 1993.

TAMSSAOUET, K.; DAUZèRE-PéRèS, S.; YUGMA, C. Metaheuristics for the job-shop scheduling problem with machine availability constraints. *Computers & Industrial Engineering*, v. 125, p. 1 – 8, 2018. ISSN 0360-8352. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0360835218303802⟩.

THIERENS, D. An Adaptive Pursuit Strategy for Allocating Operator Probabilities. In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2005. (GECCO '05), p. 1539–1546. ISBN 1-59593-010-8. Disponível em: ⟨http://doi.acm.org/10.1145/1068009.1068251⟩.

WANG, L.; ZHENG, D.-Z. An effective hybrid optimization strategy for job-shop scheduling problems. *Computers & Operations Research*, v. 28, n. 6, p. 585 – 596, 2001.

WERNER, F. Genetic algorithms for shop scheduling problems: A survey. *Preprint Series*, v. 11, p. 1–66, 10 2011.

XIA, W.-j.; WU, Z.-m. A hybrid particle swarm optimization approach for the job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, v. 29, n. 3, p. 360–366, jun. 2006. ISSN 1433-3015. Disponível em: ⟨https://doi.org/10.1007/s00170-005-2513-4⟩.

ZHANG, R.; SONG, S.; WU, C. A hybrid artificial bee colony algorithm for the job shop scheduling problem. *International Journal of Production Economics*, v. 141, n. 1, p. 167 – 178, 2013. ISSN 0925-5273. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0925527312001430⟩.

ZHANG, X.-F. et al. An efficient hybrid particle swarm optimization for the job shop scheduling problem. In: *FUZZ-IEEE*. Taipei, Taiwan: IEEE, 2011. p. 622–626.

ZHOU, H.; CHEUNG, W.; LEUNG, L. C. Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm. *European Journal of Operational Research*, v. 194, n. 3, p. 637–649, maio 2009. ISSN 0377-2217. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0377221707012404⟩.

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 0956-5515, 1572-8145. Disponível em: ⟨https://link.springer.com/article/10.1007/s10845-013-0837-8⟩.