Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Hiper-Heurísticas com Programação Genética Gramatical para o Problema de Alocação de Corredores

Rafael Freesz Resende Corrêa

JUIZ DE FORA
JUNHO, 2023

Hiper-Heurísticas com Programação Genética Gramatical para o Problema de Alocação de Corredores

Rafael Freesz Resende Corrêa

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Stênio Sã Rosário Furtado Soares

Coorientador: Heder Soares Bernardino

JUIZ DE FORA

JUNHO, 2023

Hiper-Heurísticas com Programação Genética Gramatical para o Problema de Alocação de Corredores

Rafael Freesz Resende Corrêa

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:

Stênio Sã Rosário Furtado Soares
D.Sc. em Ciência da Computação

Heder Soares Bernardino
D.Sc. em Modelagem Computacional

Luciana Conceição Dias Campos
D.Sc. em Engenharia Elétrica

Carlos Cristiano Hasenclever Borges
D.Sc. em Engenharia Civil

Luciana Brugiolo Gonçalves
D.Sc. em Ciência da Computação

JUIZ DE FORA

30 DE JUNHO, 2023

Resumo

Redução de custos e aumento de eficiência são metas fundamentais presentes na indústria, na engenharia e na ciência. O posicionamento de salas em um edifício em construção, maquinários em linhas de produção e semicondutores em placas de circuito impresso são exemplos de contextos que afetam diretamente esses fatores. Diante dessa realidade, o *Corridor Allocation Problem* (CAP) trata da distribuição física de elementos, denominados facilidades, ao longo de um corredor de dois lados, de forma a reduzir o custo de comunicação entre eles. Este trabalho propõe o desenvolvimento de novas heurísticas através da aplicação de hiper-heurísticas com Programação Genética Gramatical (PGG) em diferentes instâncias do problema, a fim de desenvolver diferentes estratégias para o CAP, possibilitando a obtenção de *layouts* de baixo custo. Foram realizadas análises estatísticas de resultados obtidos por cada heurística gerada, a fim de destacar e explorar a melhor abordagem. A aplicação de hiper-heurística foi capaz de gerar heurísticas com bons resultados, se assemelhando a heurísticas bem definidas na literatura.

Palavras-chave: CAP, Hiper-heurísticas, Heurísticas, Programação Genética Gramatical

Abstract

Reducing costs and increasing efficiency are fundamental goals present in industry, engineering and science. The positioning of rooms in a building under construction, machinery in production lines and semiconductors in printed circuit boards are examples of contexts that directly affect these factors. Faced with this reality, the Corridor Allocation Problem (CAP) deals with the physical distribution of elements, called facilities, along a two-sided corridor, in order to reduce the cost of communication between them. This work proposes the development of new heuristics through the application of hyper-heuristics with Grammar-based Genetic Programming (GGP) in different instances of the problem, in order to develop different strategies for the CAP, making it possible to obtain low-cost layouts. Statistical analyzes of the results obtained by each generated heuristic were performed in order to highlight and explore the best approach. The application of hyper-heuristics was able to generate heuristics with good results, resembling well-defined heuristics in the literature.

Keywords: CAP, Hyper-heuristics, Heuristics, Grammar-based Genetic Programming

Agradecimentos

Ao concluir essa longa jornada, reconheço que o sucesso não é alcançado de forma individual, mas sim por meio de um esforço coletivo de todos aqueles que me auxiliaram de alguma forma, permitindo-me avançar mais um passo e subir mais um degrau.

Primeiramente, gostaria de expressar minha profunda gratidão à minha família. À minha mãe, Maria Tereza, pelo seu inabalável apoio e paciência ao longo de todos esses anos de graduação. Ao meu irmão Matheus, que sempre esteve ao meu lado nessa jornada. E ao meu querido avô Walter Freesz (*in memoriam*), que, durante o tempo que esteve presente, foi o alicerce de tudo aquilo que sou capaz, e que hoje vive em minha memória e em meu coração, inspirando-me com sua força.

Gostaria de agradecer especialmente à minha noiva Flávia pelo companheirismo e apoio crucial nos momentos mais difíceis da graduação.

Também expresso minha gratidão aos meus orientadores, Stênio, Luciana, Heder e Lorenza, bem como a todos os professores do Departamento de Ciência da Computação.

"Estude com afinco aquilo que mais lhe interessa da maneira mais indisciplinada, irreverente e original possível."

Richard Feynman

Conteúdo

Lista de Figuras **8**

Lista de Tabelas **9**

Lista de Abreviações **10**

1 Introdução **11**

2 Fundamentação Teórica **15**
2.1 Problemas de *Layout* . . . 15
2.1.1 *Single-Row Facility Layout Problem* . . . 15
2.1.2 *Double-Row Facility Layout Problem* . . . 16
2.1.3 *Multi-Row Facility Layout Problem* . . . 16
2.1.4 *Space-Free Multirow Facility Layout Problem* . . . 17
2.2 *Corridor Allocation Problem* . . . 18
2.3 Meta-heurísticas . . . 19
2.4 Hiper-Heurísticas . . . 20
2.5 Gramáticas . . . 22
2.6 Programação Genética . . . 23
2.6.1 Programação Genética Gramatical . . . 24
2.7 Considerações Finais . . . 24

3 Trabalhos Relacionados **27**
3.1 Problemas Predecessores . . . 27
3.1.1 *Single-Row Facility Layout Problem* . . . 27
3.1.2 *Double-Row Facility Layout Problem* . . . 28
3.1.3 *Multi-Row Facility Layout Problem* . . . 28
3.2 *Corridor Allocation Problem* . . . 28
3.3 *Space-Free Multirow Facility Layout Problem* . . . 31
3.4 Conclusões . . . 33

4 Abordagem Proposta **34**
4.1 Hiper-heurística com Programação Genética Gramatical . . . 34
4.2 Heurísticas de baixo nível . . . 36
4.2.1 Estratégias de Perturbação . . . 36
4.2.2 Estratégias de Busca Local . . . 38
4.3 Gramática Proposta . . . 42

5 Experimentos Computacionais **46**
5.1 Instâncias do Problema . . . 46
5.1.1 Instâncias de Treino . . . 47
5.2 Execução dos Experimentos . . . 47
5.3 Análise dos Resultados . . . 48
5.3.1 Perfis de Desempenho . . . 49
5.3.2 Resultados Comparativos . . . 52
5.4 Avaliação do Melhor Agente . . . 53

6 Conclusões **57**

A Descrição das instâncias **59**

B Descrição dos Agentes **65**
B.1 A_{S9} . . . 65
B.2 A_{S10} . . . 66
B.3 A_{S11} . . . 66
B.4 A_{Am15} . . . 67
B.5 $A_{N25.1}$ . . . 67
B.6 $A_{Sko42.1}$ . . . 68
B.7 $A_{Sko42.2}$ . . . 68
B.8 $A_{Sko42.3}$ . . . 69
B.9 $A_{Sko42.4}$ . . . 69
B.10 $A_{Sko49.1}$ . . . 70
B.11 A_{Geral} . . . 70

Bibliografia **71**

Lista de Figuras

2.1 Exemplo de aplicação do SRFLP. 16
2.2 Exemplo de aplicação do DRFLP. 16
2.3 Exemplo de aplicação do MRFLP. 17
2.4 Exemplo de aplicação do SF-MRFLP. 17
2.5 Exemplo de aplicação do CAP. 19
2.6 Classificação de Hiper-heurísticas (BURKE et al., 2013) 21
2.7 Exemplo da cadeia $x + 2$ sendo derivada pelas produções de uma gramática. 22
2.8 Exemplo da operação de recombinação (CORREA et al., 2022). 25
2.9 Exemplo da operação de mutação. 26

4.1 Operação *Shake* para $k = 2$ em *layout* de tamanho 6. 37
4.2 Operação *Swap* para $i = 0$ e $j = 4$ em *layout* de tamanho 6. 38
4.3 Operação *Shift* para $t = 1$ em *layout* de tamanho 6. 39
4.4 Operação *Reconstruction* para $r = 2$ em *layout* de tamanho 6. 40
4.5 Algoritmo *swapLS* aplicado a um *layout*. 41
4.6 Algoritmo de troca entre vizinhos aplicado a um *layout*. 42
4.7 Algoritmo de troca entre não vizinhos aplicado a um *layout*. 42
4.8 Algoritmo de troca facilidades de lados opostos aplicado a um *layout*. . . . 44
4.9 Algoritmo de inserção de facilidades lados opostos aplicado a um *layout*. . 45

5.1 PD utilizando os resultados médios como medida de desempenho. As áreas normalizadas sob as curvas dos PD são 1.0000, 0.9951, 0.9949, 0.9949, 0.9946, 0.9930, 0.9927, 0.9919, 0.9915, 0.9405 e 0.8931, respectivamente, para A_{Geral}, $A_{Sko42.01}$, $A_{Sko42.03}$, $A_{Sko42.04}$, $A_{N25.01}$, A_{S11}, $A_{Sko49.01}$, $A_{Sko42.02}$, A_{S10}, A_{Am15} e A_{S9}. 53
5.2 Melhor heurística gerada pela variante Geral. 54

B.1 Melhor heurística gerada pela variante A_{S9}. 65
B.2 Melhor heurística gerada pela variante A_{S10}. 66
B.3 Melhor heurística gerada pela variante A_{S11}. 66
B.4 Melhor heurística gerada pela variante A_{Am15}. 67
B.5 Melhor heurística gerada pela variante $A_{N25.1}$. 67
B.6 Melhor heurística gerada pela variante $A_{Sko42.1}$. 68
B.7 Melhor heurística gerada pela variante $A_{Sko42.2}$. 68
B.8 Melhor heurística gerada pela variante $A_{Sko42.3}$. 69
B.9 Melhor heurística gerada pela variante $A_{Sko42.4}$. 69
B.10 Melhor heurística gerada pela variante $A_{Sko49.1}$. 70
B.11 Melhor heurística gerada pela variante Geral. 70

Lista de Tabelas

3.1 Tabela comparativa dos trabalhos relacionados. 33

5.1 Melhores soluções para instâncias de tamanho $n \leq 56$. 49
5.2 Melhores soluções para instâncias de tamanho $n \geq 60$. 50
5.3 Solução média para instâncias de tamanho $n \leq 56$. 51
5.4 Solução média para instâncias de tamanho $n \geq 60$. 52
5.5 Número de instâncias em que cada variante alcançou os melhores valores, onde #Melhor representa a contagem para o melhor custo, #Média é a contagem para os custos médios, e #TesteEstat. é o número de instâncias em que cada variante encontrou os melhores valores médios ou suas soluções são estatisticamente semelhantes aos melhores. 52
5.6 Resultados comparativos para instâncias de tamanho $n \leq 56$. 55
5.7 Resultados comparativos para instâncias de tamanho $n \geq 60$. 56

A.1 Características da Instância S9 59
A.2 Características da Instância S10 60
A.3 Características da Instância S11 60
A.4 Características da Instância Am15 61
A.5 Características da Instância N25.1 61
A.6 Características da Instância Sko.42.1. 62
A.7 Características da Instância Sko.42.2. 62
A.8 Características da Instância Sko.42.3. 63
A.9 Características da Instância Sko.42.4. 63
A.10 Características da Instância Sko.49.1. 64

Lista de Abreviações

CAP	*Corridor Allocation Problem*
GLC	Gramática Livre de Contexto
HH	Hiper-heurística
PG	Programação Genética
PGG	Programação Genética Gramatical
HPGG	Hiper-heurística com Programação Genética Gramatical
SRFLP	*Single-Row Facility Layout Problem*
DRFLP	*Double-Row Facility Layout Problem*
MRFLP	*Multi-Row Facility Layout Problem*
SF-MRFLP	*Space-Free Multirow Facility Layout Problem*
MRFLP	*Multi-Row Facility Layout Problem*
SF-DRFLP	*Space-Free Double Facility Layout Problem*
AE	Algoritmos Evolutivos
ACO	*Ant Colony Optimization*
TS	*Tabu Search*
ILS	*Iterated Local Search*
SA	*Simulated Annealing*
IG	*Iterated Greedy*
MILP	*Mixed-integer Linear Programming*
PSO	*Particle Swarm Optimization*
AG	Algoritmo Genético
SS	*Scatter Search*
PR	*Path Relinking*
kPROP	*K-paralell Row Ordering Problem*
BVNS	*Basic Variable Neighborhood Search*
RVND	*Random Variable Neighborhood Descent*
PD	Perfis de Desempenho

1 Introdução

Problemas de *layout* são aqueles onde se deseja distribuir fisicamente uma certa quantidade de elementos físicos, denominados facilidades, em uma determinada área (CELLIN, 2017). Facilidades podem ser entendidas como salas de um edifício, máquinas de um centro de produção, departamentos de uma repartição, semicondutores em placas de circuito impresso, dentre outros.

Considerando que, em um mesmo contexto, as facilidades diferem-se umas das outras, a configuração final, dada pela forma como esses elementos são distribuídos, também o são, no sentido de que determinadas distribuições podem ser mais adequadas que outras. Torna-se então necessária a elaboração de uma estratégia de alocação para o melhor funcionamento do conjunto.

Como exemplos de ponto de vista prático, problemas de *layout* envolvem a distribuição das salas de um edifício, na qual é conveniente que facilidades que possuam alta demanda de tráfego entre si fiquem mais próximas, evitando o excesso de circulação de pessoas no ambiente, melhorando então a eficiência de trabalho. A distribuição de máquinas em uma linha de produção deve ser feita de forma a manter mais próximas aquelas que mais demandam umas das outras sucessivamente. Departamentos com grande volume de interações devem ser mantidos próximos em uma repartição. Componentes eletrônicos devem ser distribuídos de forma que se obtenham placas de circuito mais compactas, e com maior eficiência (HERRÁN; COLMENAR; DUARTE, 2021).

As organizações podem muitas vezes negligenciar a importância de um projeto estrategicamente elaborado, tendo como consequência prejuízos financeiros e ineficiência operacional. Essas condições colocam em risco a competitividade da organização. Porém, a quantidade de trabalhos desenvolvidos sobre o tema demonstra o interesse em tratar o problema com o rigor científico que o mesmo requer.

O projeto desses *layouts* através de deduções empíricas é subjetivo, estando sujeitos a equívocos e, uma vez realizado e consolidado, sua reorganização é complexa e, em muitos casos, inviável.

Problemas de *layout* são NP-Difícil e, até o momento do desenvolvimento deste trabalho, não são conhecidos algoritmos exatos para estes problemas que sejam capazes de produzir o melhor arranjo possível de forma eficiente. A quantidade elevada de possíveis combinações de um *layout* torna proibitiva a avaliação de todas as possibilidades em busca da melhor configuração.

Partindo dessas condições, é desejável a utilização de métodos e ferramentas que ofereçam o melhor de ambas as características: boas configurações de *layout* em um tempo de execução viável do ponto de vista prático.

O *Corridor Allocation Problem* (CAP), objeto de estudo deste trabalho, é um típico problema de *layout* cujo objetivo é distribuir facilidades de forma sequencial ao longo de um corredor de duas fileiras. Para este problema, não são permitidos espaços vazios entre duas facilidades vizinhas e a primeira facilidade de cada lado tem seu ponto inicial mantido na abscissa zero.

A literatura tem feito o uso de forma consolidada de novas estratégias exatas e, principalmente, meta-heurísticas para abordar o problema. Meta-heurísticas são técnicas da Inteligência Computacional comumente utilizadas em problemas de otimização onde, apesar de não garantirem a melhor solução, são capazes de encontrar soluções suficientemente boas e com demanda de tempo viável do ponto de vista prático (TALBI, 2009).

O desenvolvimento de novas técnicas é necessário tendo em vista o crescente volume de dados em aplicações reais, que muitas vezes trazem especificidades que requerem o projeto de técnicas de computação inteligente que melhor se aplicam a tais cenários.

Hiper-heurísticas (HH) são uma classe de meta-heurísticas que automatizam o desenvolvimento de novas estratégias. Diferente das meta-heurísticas tradicionais que, aplicadas a um problema, realizam a busca em um espaço de soluções, hiper-heurísticas realizam buscas no espaço de heurísticas, desenvolvendo novas estratégias através da combinação e seleção de heurísticas mais simples, denominadas heurísticas de baixo nível (CORREA et al., 2022).

Programação Genética (PG) é uma técnica de criação automatizada de programas inspirada na evolução natural, compondo a classe de algoritmos evolutivos (KOZA, 1992).

Gramáticas Livres de Contexto (GLC) compõem o conjunto de gramáticas gerativas da Hierarquia de Chomsky, atuando como instrumento formal de geração de cadeias de determinada linguagem. GLC podem ser utilizadas na síntese de programas através de PG, dando origem à Programação Genética Gramatical (PGG) (WHIGHAM, 1995). A vantagem do uso de gramática formal na síntese de programas está do fato de que esta limita a estrutura do programa gerado, evitando a geração de soluções sintaticamente erradas, mantendo mais alta a média de aptidão da população (FREITAS; SOUZA; BERNARDINO, 2018).

O estado da arte apresenta um grande portfólio de estratégias de exploração de vizinhança para a estrutura do problema. Estas, bem como novas estratégias a serem sugeridas, podem ser utilizadas para compor o conjunto de operações a serem exploradas pela PGG, na intenção de sintetizar novas soluções para o problema.

Diante deste cenário e do ferramental disponível, o objetivo deste trabalho é o desenvolvimento de uma Hiper-heurística baseada em Programação Genética Gramatical (HPGG) objetivando a geração de heurísticas eficientes para o CAP.

O trabalho se inicia com um estudo da literatura, avaliando e apresentando os métodos exatos e aproximados mais utilizados para o problema, bem como para problemas similares, avaliando seus resultados, tendo como foco o custo da solução obtida. Essa avaliação permitiu identificar diferentes padrões de *frameworks*, possibilitando o agrupamento de estratégias potencialmente relevantes a serem novamente exploradas por este trabalho. Em seguida, são analisadas as diversas estratégias de movimentos, perturbações e buscas locais utilizadas mais utilizadas. Aquelas que apresentam maior potencial de melhora para o problema, dadas suas características, foram selecionadas e adaptadas para compor o conjunto de heurísticas de baixo nível utilizadas pela hiper-heurística, sendo executadas posteriormente em instâncias do CAP. Paralelamente, novas estratégias idealizadas foram utilizadas. A hiper-heurística proposta foi executada em diferentes instâncias do CAP, a fim de desenvolver heurísticas voltadas para cada problema em específico, ou para um conjunto de problemas. Por fim, foi realizada uma análise detalhada dos resultados obtidos, bem como das melhores heurísticas geradas por cada um dos agentes.

O Capítulo 2 apresenta a fundamentação teórica do problema e dos métodos

utilizados. A revisão da literatura é feita no Capítulo 3. As abordagens propostas são exploradas no Capítulo 4. O Capítulo 5 apresenta os experimentos realizados, os resultados, bem como as análises estatísticas. Por fim, o Capítulo 6 apresenta as conclusões e trabalhos futuros.

2 Fundamentação Teórica

Para a familiarização do leitor com os elementos relacionados ao assunto do presente trabalho, este capítulo apresenta na Seção 2.1 problemas de *layout* predecessores e similares, introduzindo seus conceitos e citando soluções de diversos autores.

A Seção 2.2 apresenta, com mais detalhes, o *Corridor Allocation Problem*, problema tratado neste trabalho, sendo uma variação daqueles anteriormente apresentados. Nas Seções 2.3 e 2.4, são apresentados os conceitos e finalidades de Meta-heurísticas e Hiper-heurísticas, respectivamente. A Seção 2.5 apresenta a definição de Gramáticas. Os conceitos de Programação Genética são apresentados na Seção 2.6. Por fim, na Seção 2.7 são apresentadas as considerações finais a respeito do conteúdo apresentado.

2.1 Problemas de *Layout*

Problemas de *Layout* são caracterizados pelo arranjo físico de facilidades ao longo de uma determinada área (CELLIN, 2017).

2.1.1 *Single-Row Facility Layout Problem*

O *Single-Row Facility Layout Problem* (SRFLP) foi apresentado por (KARP; HELD, 1967) e (SIMMONS, 1969), sendo inicialmente nomeado pelos autores como *Module Placement Problem* (MPP) e *One-Dimensional Space-Allocation Problem* (ODSAP), respectivamente. O termo SRFLP foi introduzido por (HERAGU; KUSIAK, 1991), passando a ser amplamente adotado em trabalhos posteriores.

O SRFLP trata a ordenação de um conjunto de n salas de comprimentos l_i, não necessariamente iguais, em um espaço linear de uma dimensão, como pode ser visto na Figura 2.1. O custo para cada par de facilidades (i, j) é dado pelo produto de um escalar f_{ij} pela sua distância d_{ij} e o objetivo do problema se consiste na minimização da soma dos produtos de todos os pares. Como única restrição, não é permitido que haja sobreposição de salas ao longo do eixo horizontal (HERAGU; KUSIAK, 1991). O cálculo da distância

entre duas facilidades considera as abscissas do ponto central de ambas na sua respectiva coordenada no eixo horizontal (SIMMONS, 1969). O escalar $f_{i,j}$ pode ser entendido como a intensidade de tráfego entre as salas (AMARAL, 2006).

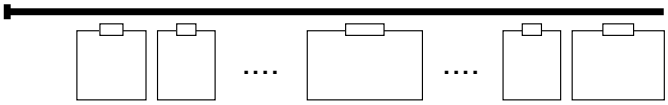


Figura 2.1: Exemplo de aplicação do SRFLP.

2.1.2 *Double-Row Facility Layout Problem*

O *Double-Row Facility Layout Problem* (DRFLP), ou *Double-Row Layout Problem* (DRLP), foi inicialmente abordado em (KUSIAK; HERAGU, 1987) como um caso particular do que foi chamado de *Quadratic Assignment Problem (QAP)*. O nome DRFLP, adotado pela literatura, foi introduzido em (CHUNG; TANCHOCO, 2010).

Este problema deriva daquele apresentado na Seção 2.1.1. Sua diferença está no fato de que as facilidades são alocadas em ambos os lados do corredor, como mostra a Figura 2.2. Da perspectiva de aplicações de *layout*, o DRLP geralmente lida com uma estrutura de fluxo de material mais eficiente e popular do que o SRFLP, uma vez que, na prática, salas são dispostas nos dois lados de um corredor (CHUNG; TANCHOCO, 2010). Os objetivos e as restrições deste problema se assemelham àqueles do problema apresentado na Seção 2.1.1.

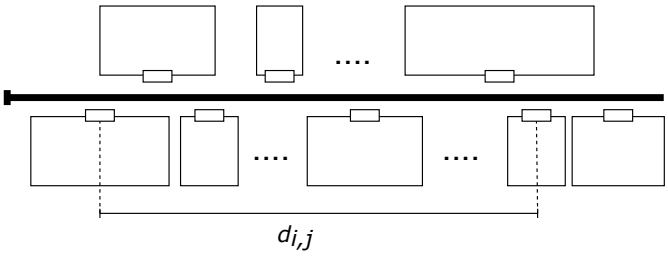


Figura 2.2: Exemplo de aplicação do DRFLP.

2.1.3 *Multi-Row Facility Layout Problem*

O Multi-Row Facility Layout Problem (MRFLP) é a generalização do problema apresentado na Seção 2.1.2. Uma vez que no DRFLP as facilidades são distribuídas em duas

fileiras, este problema dispõe de $n \geq 2$ fileiras para formação do *layout* (HASSAN, 1994), como apresenta a Figura 2.3. Em consequência das demais similaridades com os problemas já apresentados, seu objetivo, bem como restrições, também se assemelham ao apresentado na Seção 2.1.1. Algumas variações deste problema consideram também a distância vertical entre as salas (CELLIN, 2017).

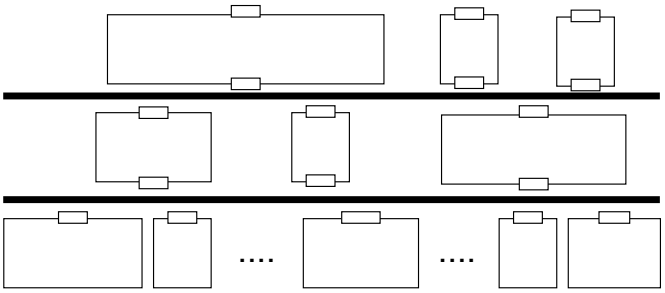


Figura 2.3: Exemplo de aplicação do MRFLP.

2.1.4 *Space-Free Multirow Facility Layout Problem*

Dentre os já apresentados, o *Space-Free Multirow Facility Layout Problem* (SF-MRFLP) é o problema de *layout* mais recente da literatura. Este problema objetiva a distribuição de facilidades ao longo de um corredor de $m \geq 2$ fileiras (ANJOS; VIEIRA, 2017). Além dos custos e e restrições semelhantes ao SRFLP, apresentado na Seção 2.1.1, este problema não permite espaços vazios entre um par de facilidades vizinhas. A Figura 2.4 apresenta um exemplo de aplicação do SFMRFLP para $m = 3$;

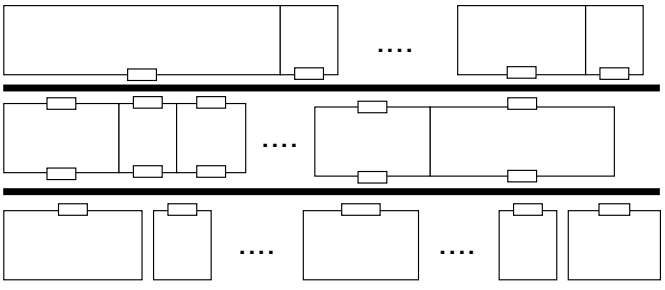


Figura 2.4: Exemplo de aplicação do SF-MRFLP.

2.2 *Corridor Allocation Problem*

O *Corridor Allocation Problem* (CAP), ou *Space-Free Double Row Facility Layout Problem* (SF-DRFLP) (FISCHER; FISCHER; HUNGERLÄNDER, 2019), objeto de estudo deste trabalho, foi proposto em (AMARAL, 2012). Seu objetivo esta na distribuição de facilidades ao longo de duas fileiras, estando sujeito as seguintes restrições:

1. Não são permitido espaços vazios entre duas facilidades vizinhas;
2. Para cada lado, a primeira facilidade estabelecida no início de cada eixo possui abscissa zero;;
3. Não é permitida a sobreposição de facilidades no *layout*.

Para um conjunto de n facilidades, cada sala possui um comprimento l_i. Cada par de facilidades possui um coeficiente de demanda f_{ij}. Cada facilidade no *layout* possui uma abscissa x_i, que representa a distância entre o início do eixo e o ponto médio da sala. O cálculo da abscissa compreende a soma dos comprimentos das facilidades que a antecedem no respectivo lado, somadas a metade de seu comprimento ($l_i/2$). Desta forma, é garantido o cumprimento da restrição 3.

A distância entre um par de facilidades é dado pelo módulo da diferença das abscissas, como mostra a Equação 2.2. O custo de cada par de facilidade (i, j) é dado pelo produto entre distância d_{ij} e o coeficiente f_{ij}. Sendo assim o objetivo do problema é a redução do custo total de tráfego, dado pela Equação 2.1

$$Z = \sum_{(i<j)} f_{ij} \times d_{ij}, \tag{2.1}$$

onde,

$$d_{ij} = |x_i - x_j|, \tag{2.2}$$

Diante da definição, observa-se que o CAP é uma versão restrita do DRFLP, apresentado na Seção 2.1.1, sendo também um caso específico do SF-MRFLP, apresentado na Seção 2.1.4 . A Figura 2.5, apresenta um exemplo de *layout* que atende as especificações do CAP.

As propostas apresentadas para este problema são exploradas com mais detalhes no Capítulo 3.

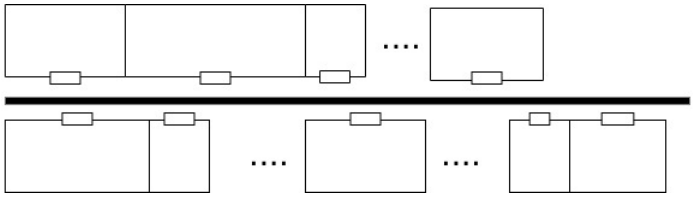


Figura 2.5: Exemplo de aplicação do CAP.

2.3 Meta-heurísticas

Muitos problemas de otimização são considerados NP-difícil. Para esses problemas, não foi possível desenvolver até o momento soluções algorítmicas que demandem tempo polinomial ($O(n^p)$, $p \in N$) (FESTA, 2014). Para problemas deste tipo, a solução exata pode ser obtida em tempo exponencial ($O(2^n)$ ou $O(n!)$). Especificamente, são problemas que demandam grande tempo de processamento para obtenção de soluções ótimas.

Algumas destas aplicações podem ser suficientemente satisfeitas se soluções de boa qualidade, não necessariamente ótimas, forem obtidas em tempo viável. Meta-heurísticas representam uma família de algoritmos aproximados de propósito geral que, quando adaptadas a um problema de otimização específico, embora não garantam a otimalidade da solução, conseguem obter resultados satisfatórios com tempo de execução aceitável (TALBI, 2009).

Algumas características definem uma meta-heurística:

- Meta-heurísticas podem ser construídas com base em observações da natureza (*nature inspired*), tendo como exemplo Algoritmos Evolutivos (AE), *Artificial Immune Systems* (AIS) e *Ant Colony Optimization* (ACO). Exemplos de algoritmos não inspirados pela natureza (*non-nature inspired)* são *Tabu Search* (TS) e *Iterated Local Search* (ILS) (GOGNA; TAYAL, 2013);

- O uso de memória pode ser adotado por uma meta-heurística para tomadas de decisões futuras, como a TS. Demais algoritmos, como *Simulated Annealing* (SA), consideram apenas os dados atuais presentes na busca;

- Alguns algoritmos, como Busca Local (BL) e TS, apresentam comportamento determinístico onde, para diversas execuções de uma dada solução inicial, as mesmas decisões são tomadas e o resultado obtido é o mesmo. Meta-heurísticas estocásticas podem fazer com que os resultados de diversas execuções variem, ainda que os dados de entrada sejam iguais. AE e SA são exemplos de métodos estocásticos.

- O algoritmo pode trabalhar inicialmente com uma única solução inicial, como *Iterated Greedy* (IG) e ILS, submetendo-a a sucessivas alterações ao longo de diversas iterações. Outros métodos, consideram a utilização de uma população de soluções, onde normalmente passam por processos de recombinação e mutação, como em AEs;

- Algoritmos iterativos iniciam-se a partir de uma solução inicial (ou uma população de soluções iniciais), a qual passa por modificações iterativamente até que um critério de parada seja atingido. Algoritmos gulosos se iniciam com uma solução vazia, construindo-a a partir de tomadas uma decisão feitas a cada passo, até que seja finalizada. Exemplos de algoritmos iterativos são IG e ILS.

2.4 Hiper-Heurísticas

Embora métodos heurísticos tenham, obtido consolidado sucesso na solução de diversos problemas de otimização, alguns aspectos ainda os tornam desafiadores. A grande quantidade de parâmetros, bem como a definição do algoritmo a ser usado, são exemplos de aspectos complexos destes métodos. Os métodos heurísticos existentes podem ser ineficientes se aplicados a determinadas variações de problemas já tratados, ou até mesmo novas instâncias de um mesmo problema (BURKE et al., 2013).

Ao contrário de métodos heurísticos, cujo espaço de busca abrange diferentes soluções de um problema, hiper-heurísticas atuam em um nível de abstração mais alto (COWLING; KENDALL; SOUBEIGA, 2001), realizando buscas em um espaços de heurísticas para um determinado problema. Seu objetivo é, para um dado problema e um conjunto de heurísticas mais simples, denominadas heurísticas de baixo nível, realizar diferentes combinações a fim de gerar uma heurística adequada (BURKE et al., 2013).

Hiper-heurísticas podem ser classificadas em duas dimensões diferentes, como

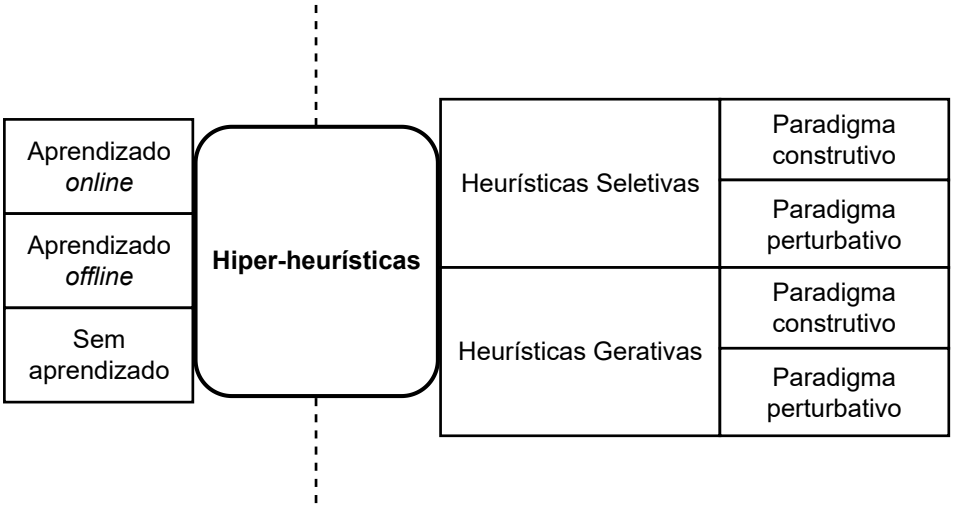


Figura 2.6: Classificação de Hiper-heurísticas (BURKE et al., 2013)

.

mostra a Figura 2.6:

- A natureza do espaço de busca das heurísticas:
  - **Heurística Seletiva**: Realiza a combinação ou seleção de heurísticas existentes;
  - **Heurística Gerativa**: Gera novas heurísticas através de componentes de outras heurísticas existentes;
- Os paradigmas a serem utilizados:
  - **Paradigma Construtivo**: A solução candidata é total ou parcialmente construída pela heurística gerada;
  - **Paradigma Perturbativo**: A heurística recebe como entrada uma solução candidata completa e realiza modificações em partes da solução;
- A fonte de *feedback* de informação:
  - **Aprendizado *Online***: O aprendizado ocorre ao longo do processo de execução da hiper-heurística para uma dada instância do problema;
  - **Aprendizado *Offline***: O aprendizado é realizado através de regras para um dado conjunto de instâncias de treinamento;

– **Sem Aprendizado**: Não há *feedback* para fins de aprendizado no processo de busca.

2.5 Gramáticas

Uma Gramática é um instrumento formal que possui a capacidade de construir conjuntos de sequências em uma linguagem específica. As gramáticas desempenham um papel significativo ao facilitar a descrição das características das linguagens (HOPCROFT; ULLMAN; MOTWANI, 2002).

Uma gramática é composta pela quádrupla de conjuntos finitos (N, T, P, S) onde:

- N representa um conjunto de não-terminais (ou variáveis);
- T é um conjunto de terminais (ou símbolos);
- P é um conjunto de regras, ou produções, através dos elementos de N e T;
- S é o não-terminal inicial da gramática ($S \in N$).

As produções de uma gramática permitem geração de cadeias de símbolos através da derivação de outras cadeias. Uma cadeia derivada com base nas produções de uma gramática pode ser representada por uma árvore, onde os nós internos são compostos por não terminais e as folhas são terminais (HOPCROFT; ULLMAN; MOTWANI, 2002). Como exemplo, a Figura 2.7 mostra uma árvore resultante da derivação da expressão "$x + 2$", onde os nós E e T são não-terminais e os nós "x","+" e "2" são terminais.

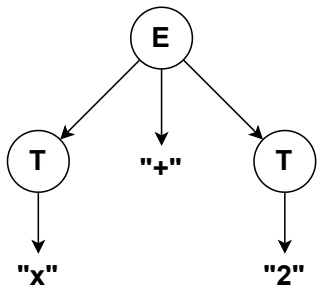


Figura 2.7: Exemplo da cadeia $x + 2$ sendo derivada pelas produções de uma gramática.

Gramáticas podem ser classificadas hierarquicamente de acordo com as restrições impostas na linguagem. Esta classificação é conhecida como Hierarquia de Chomsky.

Nesta classificação, gramáticas de tipo mais alto são mais restritas em relação as respectivas linguagens geradas (HOPCROFT; ULLMAN; MOTWANI, 2002). São elas:

- **Tipo 0**: Irrestrita;
- **Tipo 1**: Gramática Sensível ao Contexto;
- **Tipo 2**: Gramática Livre de Contexto;
- **Tipo 3**: Gramática Regular.

2.6 Programação Genética

Programação Genética (PG) (KOZA, 1992) é uma meta-heurística baseada na evolução natural destinada à geração de programas. Uma população de indivíduos compõe as possíveis soluções, onde cada indivíduo representa uma solução para o problema.

Na PG tradicional, indivíduos são normalmente codificados em árvores, onde os nós internos são operações - como estruturas de controle e operações matemáticas - e as folhas são operandos - como valores constantes ou variáveis.

O desempenho de um indivíduo é definido pela sua aptidão (*fitness*), baseada em uma função objetivo. Seu cálculo geralmente avalia o modelo sobre um conjunto de dados ou executa um controlador para realizar uma determinada tarefa.

Assim como em demais meta-heurísticas populacionais, a PG evolui sua população ao longo de gerações, funcionando da seguinte forma:

1. Uma população inicial é gerada e avaliada de acordo com uma função objetivo;
2. Indivíduos da população são selecionados, formando um conjunto de progenitores;
3. Os progenitores são recombinados (*crossover*), gerando novos indivíduos descendentes;
4. Os descendentes passam por processos de mutação;
5. As aptidões dos descendentes são avaliadas;
6. Os novos indivíduos passam a compor a população original;

7. Repetir os passos de 2 a 6 até que um critério de parada seja satisfeito, retornando o melhor indivíduo.

Para o processo de recombinação, normalmente são selecionados dois parentais. Nesta operação, descendentes são gerados através da combinação de comportamentos pertencentes aos parentais. Os processos de mutação são aplicados aos descendentes recém criados através da perturbação da solução candidata, sem considerar se a solução resultante será melhorada (BURKE et al., 2013).

2.6.1 Programação Genética Gramatical

A Programação Genética Gramatical (PGG) (WHIGHAM, 1995) é uma variação da PG que utiliza gramática formal na representação de indivíduos. A vantagem do uso desta técnica está na capacidade de restringir o espaço de busca e introduzir conhecimento prévio.

Desta forma, um indivíduo da PGG é estruturado através de uma árvore de derivação, criada de acordo com regras gramaticais. Neste caso, as folhas representam terminais e os nós internos são não-terminais.

O processo de recombinação é realizado da seguinte forma: Dois nós internos do mesmo tipo são selecionados aleatoriamente em cada um dos parentais. As sub-árvores desses nós são trocadas, gerando dois novos indivíduos. A operação é exemplificada através da Figura 2.8

Para o processo de mutação, um nó interno aleatório é selecionado, e sua respectiva subárvore é novamente gerada, derivada de acordo com as regras da gramática. A Figura 2.9 apresenta um exemplo de processo de mutação.

2.7 Considerações Finais

Neste capítulo foram apresentados e explorados os diversos problemas de *layout* presentes na literatura; em especial, o CAP.

Todas as variações de problemas de *layout* são da classe NP-Difícil (LOIOLA et al., 2007). Desta forma, métodos exatos não apresentam bom desempenho quando

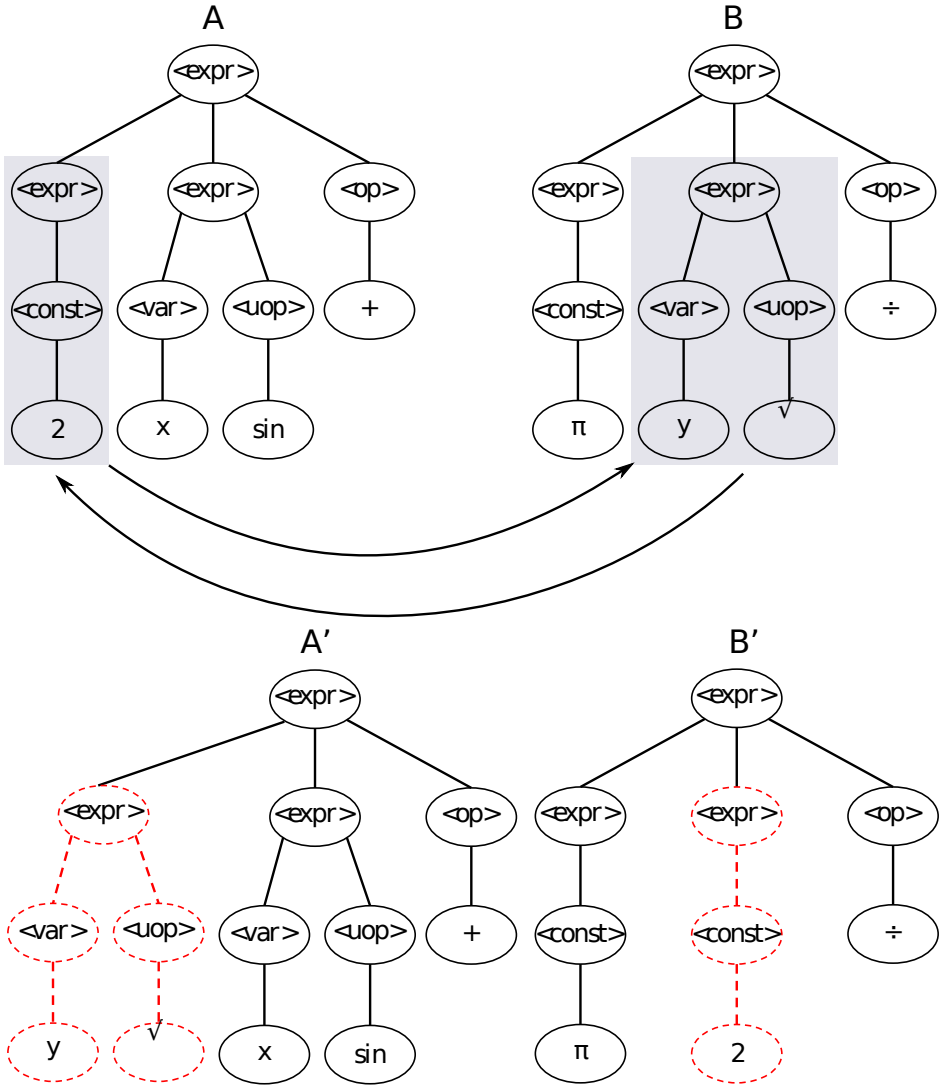


Figura 2.8: Exemplo da operação de recombinação (CORREA et al., 2022).

aplicados a esse tipo de problema quando são consideradas instâncias de dimensões médias ou grandes.

Esta condição permite afirmar que o CAP se enquadra no universo de problemas comumente abordados por meta-heurísticas, apresentada na Seção 2.3. Considerando o comportamento dinâmico e a complexidade intrínseca dos problemas, o desafio no uso de meta-heurísticas se dá pela dificuldade em adaptar a heurística de acordo com o problema específico, contando também com uma elevada quantidade de parâmetros a serem definidos.

Dada a demanda por novos métodos, bem como a existência *frameworks* para automatização do desenvolvimento de novas heurísticas, através de hiper-heurísticas, este trabalho propõe uma hiper-heurística através de PGG, como forma de contribuir para a

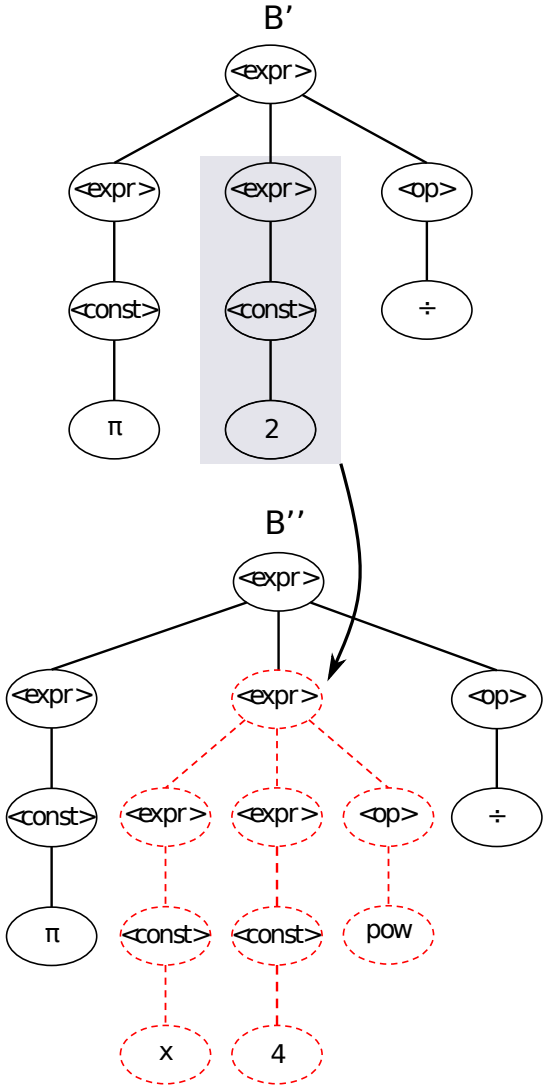


Figura 2.9: Exemplo da operação de mutação.

área de aplicação do problema.

3 Trabalhos Relacionados

Este capítulo tem a finalidade de apresentar e detalhar trabalhos anteriores relacionados ao CAP. Uma análise de trabalhos de problemas similares é brevemente realizada na Seção 3.1. Dado que o SF-MRFLP é a generalização do CAP, sua análise se mostra igualmente relevante, e será explorada separadamente na Seção 3.3. Trabalhos relacionados exclusivamente ao CAP são apresentados na Seção 3.2. A Seção 3.4 conclui este capítulo com uma análise geral comparativa dos métodos apresentados.

3.1 Problemas Predecessores

Esta seção faz uma breve revisão da literatura referente a problemas similares e predecessores ao CAP, apresentados no Capítulo 2.

3.1.1 *Single-Row Facility Layout Problem*

Um dos primeiros algoritmos para o SRFLP utiliza o método *branch-and-bound* e foi proposto em (SIMMONS, 1969). Em (AMARAL, 2006) foi proposto o uso de Programação Linear Inteira Mista, ou *Mixed-integer Linear Programming (MILP)*. Meta-heurísticas foram amplamente utilizadas por diversos autores. Uma proposta com SA foi apresentada em (HERAGU; ALFA, 1992a; PALUBECKIS, 2017). A mesma estratégia, aliada a TS, foi utilizada em (ALVARENGA et al., 2000). Uma abordagem com ACO, foi proposta em (SOLIMANPUR; VRAT; SHANKAR, 2005). Otimização por enxame de partículas, ou *Particle Swarm Optimization* (PSO), foi utilizada em (SAMARGHANDI; TAABAYAN; JAHANTIGH, 2010). Foi utilizada também estratégia de TS em (SAMARGHANDI; ESHGHI, 2010; KOTHARI; GHOSH, 2013). Abordagens com Algoritmos Genéticos (AG) foram propostas em (OZCELIK, 2012; DATTA; AMARAL; FIGUEIRA, 2011). Em (KUMAR et al., 2008; KOTHARI; GHOSH, 2014) são propostas variações de *Scatter Search* (SS). Um algoritmo de *Variable Neighborhood Search* (VNS) foi apresentado em (PALUBECKIS, 2015; GUAN; LIN, 2016). O segundo, utiliza juntamente um ACO. Por

fim, *Greedy Randomized Adaptive Search Procedure* (GRASP) é explorado em (CRAVO; AMARAL, 2019).

3.1.2 *Double-Row Facility Layout Problem*

Em (CHUNG; TANCHOCO, 2010; AMARAL, 2013) são propostos diferentes modelos de MILP. Propostas semelhantes são observadas em trabalhos mais recentes, como em (CHAE; REGAN, 2020; AMARAL, 2021). Com uma abordagem diferente, em (AMARAL, 2020) é apresentada uma solução através de *2-opt Local Search*. Diferentemente do problema anterior, o DRFLP não foi profundamente explorado pela literatura.

3.1.3 *Multi-Row Facility Layout Problem*

Encontra-se na literatura o uso de meta-heurísticas como SA em (HERAGU; ALFA, 1992b) e AG em (EL-BAZ, 2004; FICKO; BREZOCNIK; BALIC, 2004). Essas duas estratégias foram usadas em conjunto com TS em (LEE; LEE, 2002). Um AG também foi incorporado a TS em (MIAO; XU, 2009). Por fim, em (FICKO et al., 2010) é proposto um PSO.

3.2 *Corridor Allocation Problem*

Dentre os trabalhos relacionados ao CAP, essa seção descreve detalhadamente quatro trabalhos da literatura que obtém relevância histórica e resultados expressivos. Destes, quais três utilizam meta-heurística e um utiliza abordagem exata.

Em (AMARAL, 2012), trabalho que apresenta este problema, foi proposto de início a utilização de MILP, a fim de encontrar soluções ótimas para instâncias pequenas. Foram utilizados métodos determinísticos de refinamento, como 2-opt e 3-opt. Estes métodos foram utilizados como busca local, sendo precedidos por perturbações, dando origem a duas heurísticas. Os algoritmos de perturbação realizam esse par de operações repetidamente até que um critério de parada seja atingido.

Na primeira heurística, para uma instância de tamanho n, o método de perturbação proposto realiza n trocas aleatórias no arranjo, refinando o *layout* em seguida

através da busca local. O critério de parada do algoritmo é o parâmetro k que representa a quantidade de iterações no laço. A segunda heurística trabalha de forma semelhante, diferindo-se apenas no método de perturbação, onde este realiza a inversão de s elementos consecutivos do *layout* a partir de uma posição i, onde $s \leq n$.

Os algoritmos foram executados em um total de 13 instâncias de tamanho n, com $9 \leq n \leq 30$ facilidades. Todos os resultados obtidos são ótimos globais.

O segundo trabalho da literatura, realizado em (GHOSH; KOTHARI, 2012), foram desenvolvidos dois algoritmos evolutivos para o CAP. A primeira abordagem utiliza um AG aliado a uma estratégia de busca local, denominado CAP-GA. Para este algoritmo, uma população inicial de 60 indivíduos é aleatoriamente gerada, onde um indivíduo é uma solução do problema. Essa população passa por sucessivas operações de recombinação, mutação e busca local ao longo de 500 gerações. As operações de recombinação e mutação de indivíduos são aplicadas de acordo com uma probabilidade p_r e p_m, respectivamente. A busca local é aplicada a toda população a cada período de cinco gerações.

O segundo algoritmo se baseia no uso de SS aliado a *Path Relinking* (PR), denominado CAP-SS. Assim como o primeiro algoritmo, o SS constrói inicialmente uma população compostas por 100 indivíduos. A população inicial passa por refinamento utilizando a mesma busca local apresentada para o algoritmo genético. Um subconjunto desses indivíduos é denominado conjunto de referência, contendo as melhores soluções da população. Inicia-se um ciclo onde, o conjunto de referência é subdividido em pares, passando por recombinações entre si. Os indivíduos gerados nesse processo substituem aqueles contidos no antigo conjunto de referência. Ao término das recombinações, é realizada uma operação de PR em todas as novas soluções. O laço se mantém até que o custo médio do novo conjunto de referências seja superior ao antigo.

Os dois algoritmos foram executados em 45 instâncias de tamanhos $9 \leq n \leq 49$. Comparados a (AMARAL, 2012), ambas propostas foram capazes de atingir os mesmos custos.

Duas propostas diferentes também foram apresentadas em (AHONEN; ALVARENGA; AMARAL, 2014), onde as meta-heurísticas de TS (CAP-TS) e SA (CAP-GA) são adaptadas para o CAP. Uma busca local foi desenvolvida e utilizada por ambas abor-

dagens. Suas operações consistem nas trocas de todos os pares de facilidade, que ocorrem iterativamente, persistindo aquelas que reduzem o custo da função objetivo. As iterações da busca se reiniciam até que nenhum movimento seja capaz de melhorar a solução.

O CAP-TS gera inicialmente uma solução aleatória, a qual passa por refinamento com a busca local. O termo tabu se refere a determinados movimentos de exploração do espaço de solução que são utilizados com mais frequência ao longo da execução. A principal ideia do algoritmo é a realização de sucessivas trocas de forma que sejam utilizados movimentos "não tabu", permitindo a diversificação do espaço de soluções. Ao longo de 100.000 iterações, ou 50.000 iterações sem melhora, uma sequência de trocas "não tabu" são realizadas, seguidas de nova busca local. A solução corrente é atualizada em caso de redução de custo.

O CAP-SA se difere das demais meta-heurísticas pelo fato de, em determinados momentos, possibilitar a aceitação de uma solução de custo superior à anterior. A aceitação de soluções é definida de forma probabilística em relação a um parâmetro denominado temperatura. Ao longo das iterações, esse parâmetro sofre um aumento, sendo mais suscetível a aceitação de soluções ruins. O algoritmo se inicia com um *layout* gerado aleatoriamente, sendo refinado também pela busca local. Em seguida, são realizadas quatro iterações, nas quais o SA realiza sucessivas trocas até que a temperatura máxima seja atingida, retornando então uma solução. Essa solução passa pela busca local, realimentando posteriormente o algoritmo.

Os algoritmos propostos foram executados em 68 instâncias de tamanhos $9 \leq n \leq 70$. Comparadas as instâncias das literaturas anteriores, o CAP-TS apresentou melhores resultados para instâncias com $n \geq 42$. Para instâncias menores, as quais se conhece a solução exata, os custos foram igualados. O CAP-SA foi capaz de superár os custos do CAP-TS na maioria das instâncias com $n \geq 42$.

O CAP foi abordado em (SANTANA; SANTOS, 2016) através do uso de ILS, aplicando iterativamente uma perturbação, seguida por uma busca local. A perturbação é realizada através de uma série de trocas aleatórias de elementos do *layout*. A quantidade de movimentos varia ao longo da execução, aumentando a quantidade de trocas quando há insucesso na busca por soluções de custo menor.

A busca local é realizada através da troca de todo par i, j de facilidades, onde $i \leq j$. As trocas que reduzem o custo da solução são mantidas. Para este caso, o ciclo de trocas é reiniciado, e segue em execução até que nenhuma troca seja capaz de reduzir o custo.

O ILS foi executado em um total de 53 instâncias de tamanho $10 \leq n \leq 60$, com tempos de execução variando entre $0,004s \leq t \leq 22,984s$. Embora haja uma melhora significativa no tempo, a qualidade das soluções fica abaixo das heurísticas apresentadas até então.

3.3 *Space-Free Multirow Facility Layout Problem*

A recente exploração deste problema limita o volume de trabalhos relacionados. Apesar deste fato, foram identificados três trabalhos relevantes ao tema, sendo uma proposta exata e duas meta-heurísticas.

Em (TANIYAMA TAVARES, 2018) é proposto um algoritmo *Hill-Climbing* (HC) para o SF-MRFLP. Para este algoritmo, uma solução é gerada aleatoriamente onde, através de sucessivas iterações, são realizadas duas buscas locais simultaneamente, prevalecendo a solução de menor custo retornada. Esse procedimento ocorre repetidamente uma quantidade pré-fixada de vezes. A primeira busca local do algoritmo efetua uma permutação simples entre todos os pares de facilidade do *layout*, sem que seja alterada a quantidade de facilidades em cada fileira. São mantidas as trocas que reduzem o custo da solução, reiniciando o ciclo de iterações até que nenhuma troca melhore a solução. A segunda busca local realiza a troca de facilidades de diferentes fileiras, mantendo o critério de parada e de reinício da busca anterior. Foram executadas ao todo 18 instâncias de tamanhos $8 \leq n \leq 30$, com $m = \{2, 3\}$ fileiras.

Na busca por ótimos globais, utilizando as mesmas instâncias do problema anterior, em (FISCHER; FISCHER; HUNGERLÄNDER, 2019) é proposto um algoritmo exato para o SF-MRFLP através de MILP. O autor reduz a solução deste problema ao *K-paralell Row Ordering Problem* (kPROP), uma versão semelhante ao SF-MRFLP, no qual a fileira m onde cada facilidade i será posicionada é pré-definida pela instância. Essa restrição reduz o espaço de busca do kPROP em relação ao SF-MRFLP, reduzindo sua

complexidade.

De forma preliminar, seguindo os princípios do kPROP, é gerado um conjunto enumerado de todas as possíveis atribuições pré-definidas de fileiras para cada facilidade da instância, restringindo-se àquelas em que o comprimento de nenhum lado exceda o do outro em mais de um sala, condição que reduz consideravelmente a quantidade de elementos enumerados.

Diante desse conjunto pré-definido, o algoritmo proposto trabalha em cima de cada elemento, se limitando a testar todas as possíveis permutações em cada fileira, sem que os elementos de duas fileiras diferentes sejam trocados entre si, respeitando as restrições de lado definidas pela etapa anterior. Ao final, o algoritmo é capaz de retornar uma solução cujo custo representa o ótimo local da instância.

O algoritmo foi executado considerando os casos de $m = \{2, 3, 4, 5\}$ fileiras, utilizando 58 instâncias com $5 \leq n \leq 15$ facilidades.

Em (HERRÁN; COLMENAR; DUARTE, 2021) são exploradas meta-heurísticas através do *Basic Variable Neighborhood Search* (BVNS), que é composto por dois laços aninhados de procedimentos. Um laço externo realiza a execução do laço interno uma quantidade fixa de vezes. O segundo, por sua vez, que é não determinístico, realiza uma sequência de perturbações de tamanho k, seguido por uma busca local. O valor de k é consequentemente aumentado em caso de iterações que não ocasionam melhoria de custo, permitindo que perturbações maiores sejam realizadas, em buscar novas bacias de atração. Para cada iteração de melhoria, o valor de k é redefinido para seu valor de entrada ($k = k_0$). O critério de parada do laço interno é obtido quando $k = k_{max}$, onde k_{max} é um valor limite para o tamanho da perturbação.

A busca local proposta realiza dois conjuntos de movimentos. O primeiro se baseia em movimentos de troca onde, dada uma facilidade de índice i selecionada aleatoriamente, sua posição é trocada com todos os outros elementos do *layout*, permanecendo no índice de menor custo possível, quando a nova solução é melhor que a corrente. Esse procedimento ocorre iterativamente até que nenhuma facilidade ocasione melhora. O segundo conjunto de movimentos procede de forma semelhante ao anterior, se baseando porém em movimentos de inserção, onde a facilidade selecionada é testada sendo inserida

Tabela 3.1: Tabela comparativa dos trabalhos relacionados.

Referência	**Técnica**							
	MILP	GA	SS	TS	SA	ILS	HC	BVNS
(AMARAL, 2012)	✓							
(GHOSH; KOTHARI, 2012)		✓	✓					
(AHONEN; ALVARENGA; AMARAL, 2014)				✓	✓			
(SANTANA; SANTOS, 2016)						✓		
(TANIYAMA TAVARES, 2018)							✓	
(FISCHER; FISCHER; HUNGERLÄNDER, 2019)	✓							
(HERRÁN; COLMENAR; DUARTE, 2021)								✓

em um índice diferente, sem realizar sua troca com uma segunda sala. O método utilizado para perturbação efetua trocas aleatórias de facilidades presentes na mesma fileira.

O BVND deste trabalho foi executado em 89 instâncias de tamanho n, onde $9 \leq n \leq 70$, para quantidade de fileiras $m = \{2, 3, 4, 5\}$. O custo das soluções obtidas são os melhores da literatura até o momento de desenvolvimento deste trabalho.

3.4 Conclusões

Os trabalhos relacionados mostram uma vasta gama de abordagens ao CAP e problemas de semelhantes. Foi possível observar o uso tanto de métodos exatos quanto de métodos aproximados. A Tabela 3.1 apresenta um resumo dessas abordagens.

Os diferentes métodos de exploração de vizinhança observados na literatura tornam possível seu uso como ferramenta para o desenvolvimento de uma hiper-heurística, dada a demanda de métodos mais sofisticados, em função do crescente volume de dados em aplicações reais.

4 Abordagem Proposta

Diante dos conceitos e dos trabalhos relacionados previamente vistos, este capítulo apresenta na Seção 4.1 a hiper-heurística proposta. As heurísticas de baixo nível utilizadas são detalhadas na Seção 4.2. Por fim, na Seção 4.3 a gramática utilizada é apresentada.

4.1 Hiper-heurística com Programação Genética Gramatical

O presente trabalho propõe o desenvolvimento de uma Hiper-heurística com Programação Genética Gramatical (HPGG), modificada a fim de gerar novas heurísticas para o CAP. Neste método, a população de heurísticas é inicialmente gerada de forma aleatória, evoluindo seus indivíduos ao longo de gerações, até que um critério de parada seja atingido. Um indivíduo do ponto de vista da programação genética representa uma heurística presente no espaço de heurísticas. Para o algoritmo proposto, o critério de parada foi definido como o número de gerações.

Considerando as diferentes classificações apresentadas na Figura 2.6, a hiper-heurística desenvolvida compreende uma heurística seletiva, do ponto de vista da natureza do espaço de busca, utilizando paradigma perturbativo com aprendizado *offline*.

Em uma única geração, pares de indivíduos são selecionados para compor o conjunto de progenitores. Os progenitores são selecionados aleatoriamente dentre aqueles que compõem a população principal. Em seguida, cada par de progenitores é recombinado gerando dois novos indivíduos, estando sujeitos posteriormente ao processo de mutação. Em seguida, os novos indivíduos gerados são reavaliados e passam a compor a população original. Os processos de recombinação e mutação são semelhantes àqueles apresentados na Seção 2.6.1. Ao final de uma geração, uma porcentagem dos melhores indivíduos da população original é mantida, os demais são substituídos pelos novos indivíduos gerados. O processo de persistência de melhores indivíduos é denominado elitismo. O Algoritmo 1

apresenta um pseudocódigo da HPGG.

Algoritmo 1: *Pseudo-código para Programação Genética*

Input: $PSize, gen$
Output: *pop.best*
1 $pop \leftarrow CriaPopulacaoInicial(PSize)$
2 $avaliar(pop)$
3 **for** $i = 1, ..., gen$ **do**
4 $\quad parentais \leftarrow selecionar(pop)$
5 $\quad novos\_individuos \leftarrow recombinar(parentais)$
6 $\quad novos\_individuos' \leftarrow mutar(novos\_individuos)$
7 $\quad avaliar(novos\_individuos)$
8 $\quad substituir(pop, novos\_individuos)$
9 **end for**
10 **return** *pop.best*;

São recebidos como parâmetros o tamanho da população ($PSize$) e a quantidade de gerações a serem evoluídas (gen). Na linha 1, uma população de heurísticas é inicialmente gerada. A linha 2 avalia e ordena todos os indivíduos da população de acordo com a aptidão. O laço das linhas 3-8 compreende as operações a serem executadas ao longo das gerações. A linha 4 realiza a seleção aleatória de pares de indivíduos que irão compor o conjunto de progenitores. A recombinação é realizada na linha 5, gerando conjunto de novos indivíduos. Em seguida, a operação de mutação é realizada na linha 6. Os novos indivíduos são avaliados e classificados na linha 7, sendo trazidos para a população principal na linha 8. Por fim, na linha 10, é retornado o melhor indivíduo da população (*pop.best*) final, que é aquele que apresenta maior aptidão.

Duas estratégias distintas foram usadas para o cálculo da aptidão, baseado na forma como se dão os testes. Primeiramente, um subconjunto de instâncias do problema foi selecionado a fim de desenvolver indivíduos dedicados à solução de cada instância específica. Para este caso, a aptidão de um indivíduo ($f_1(p_i)$), para uma dada instância do problema, é definida como o hiato entre o custo do indivíduo (z_{p_i}) e o melhor resultado presente na literatura ($z^*_{p_i}$), como mostra a Equação 4.1.

$$f_1(p_i) = (z_{p_i} - z^*_{p_i})/z^*_{p_i} \tag{4.1}$$

Na segunda formulação da função objetivo, os indivíduos são desenvolvidos utilizando todas as instâncias de treino utilizadas pela abordagem anterior. Para este caso, a

aptidão é definida como

$$f_2(P) = \sum_{i=1}^{|P|} f_1(p_i) \ / \ |P|. \tag{4.2}$$

ou seja, f_2 representa a média de hiatos do indivíduo considerando todas as instâncias de treino.

O uso de duas estratégias estima comparar o desempenho da PGG considerando o uso de uma única instância, bem como o uso de um conjunto de instâncias. A premissa é de que, embora a primeira abordagem tenha tendência de demanda de tempo consideravelmente inferior, a segunda tende a gerar melhores indivíduos dada a heterogeneidade de suas instâncias de treino. Por se tratar de um problema de minimização, a aptidão é inversamente proporcional ao valor do hiato nos dois casos.

4.2 Heurísticas de baixo nível

As heurísticas de baixo nível utilizadas neste trabalho compõem o conjunto de recursos a ser utilizado pela HPGG. Pela análise das técnicas utilizadas nos trabalhos relacionados foi possível identificar heurísticas que se dividem em estratégias de perturbação e estratégias de busca local, sendo compostas majoritariamente pela permutação de facilidades do *layout*.

4.2.1 Estratégias de Perturbação

Em linhas gerais, movimentos de perturbação realizam uma série de modificações em parte da solução. As alterações são persistidas independente de sua qualidade após a operação. A intenção deste recurso é alterar a solução na expectativa de sair de bacias de atração, permitindo que novos ótimos locais sejam atingidos no espaço de buscas (TALBI, 2009).

Foi possível identificar este tipo de abordagem em diversos trabalhos, como em (SANTANA; SANTOS, 2016), através de sucessivas permutações aleatórias, e em (KOTHARI; GHOSH, 2013), que aceita novas soluções de custo superior.

Duas estratégias foram baseadas em trabalhos de outros autores, outras duas foram idealizadas por este trabalho. Estas operações são definidas como segue:

1. ***Shake*:**

Baseado no método de perturbação apresentado por (SANTANA; SANTOS, 2016), esta estratégia realiza a troca aleatória de k pares de facilidades. O valor de k é obtido proporcionalmente em relação ao tamanho da instância n, onde $k = p \times n$, $0 \leq p \leq 1 \in R$. A Figura 4.1 demonstra um exemplo da aplicação deste movimento para $k = 2$.

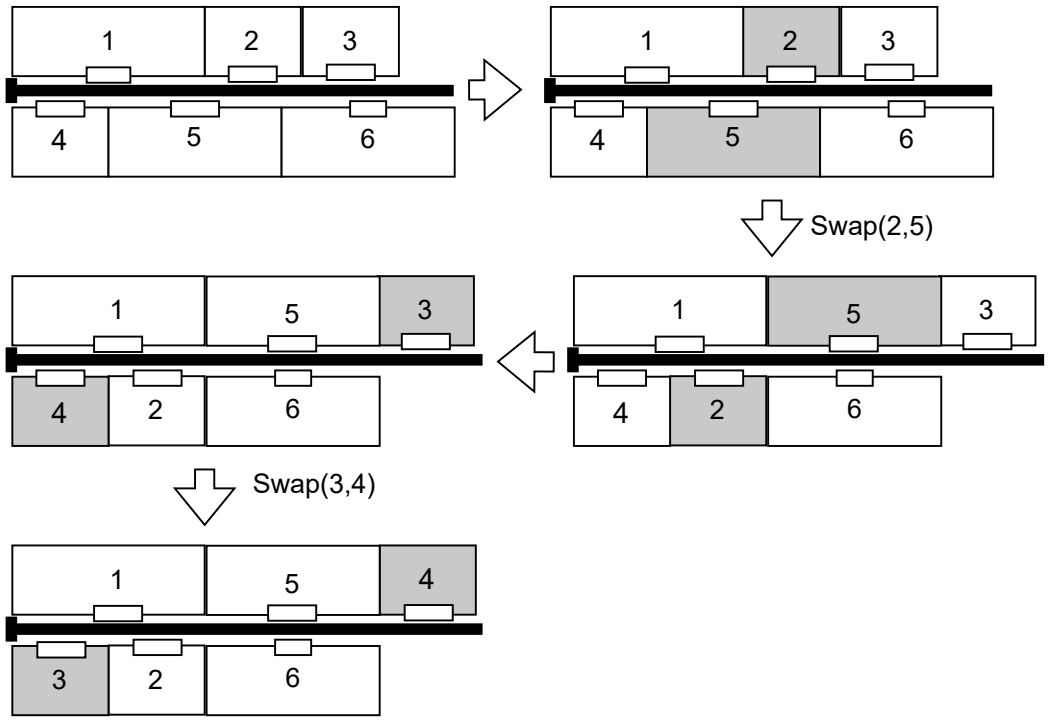


Figura 4.1: Operação *Shake* para $k = 2$ em *layout* de tamanho 6.

2. ***Swap*:**

O movimento *Swap* representa a troca exclusiva de um único par de índices (i, j) do *layout*. Os valores de (i, j) são definidos de maneira análoga à estratégia anterior, onde $i = p \times n$ e $j = p' \times n$, $p \neq p'$. O *Swap* pode ser entendido como um movimento único da estratégia anterior, se diferindo pelo fato de, neste caso, não haver aleatoriedade nos índices que serão trocados. A operação é exemplificada na Figura 4.2.

3. ***Shift*:**

Baseado na representação do corredor em um único vetor de inteiros onde t representa o início do lado direito do corredor, o *shift* realiza a movimentação do índice t em r posições, tal que $0 \leq t + r \leq n$. Para cada unidade de t acrescida, a primeira

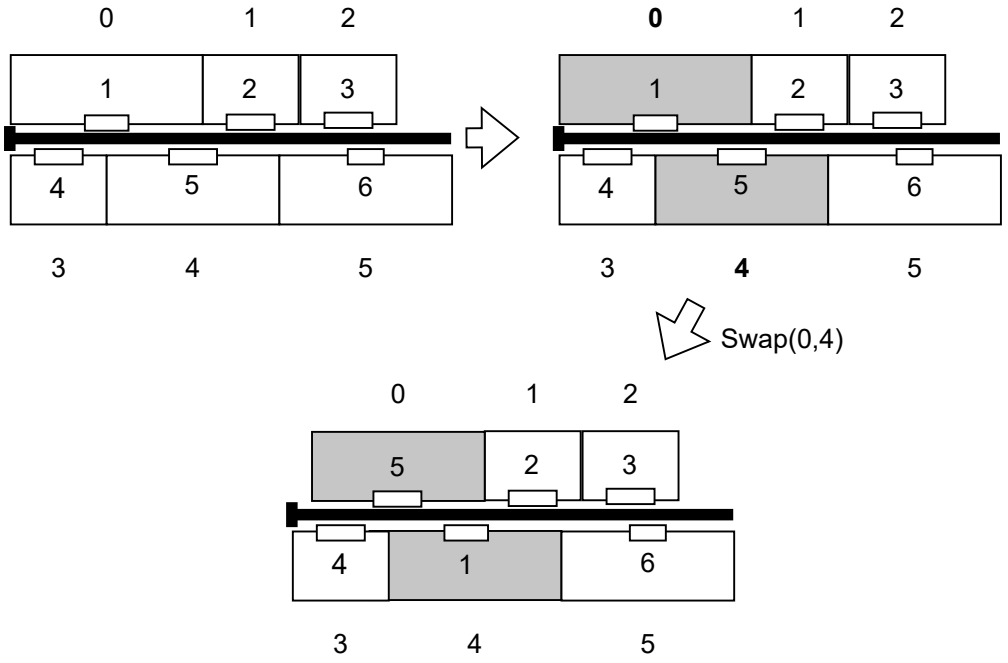


Figura 4.2: Operação *Swap* para $i = 0$ e $j = 4$ em *layout* de tamanho 6.

facilidade do lado direito do corredor passa a compor a última facilidade do lado esquerdo. A redução do valor de t corresponde a transferência da última facilidade do lado esquerdo para a primeira posição do lado direito. A Figura 4.3 demonstra um exemplo da representação vetorial com a aplicação de *shift* para $r = 1$.

4. ***Reconstruction*:**

   Neste processo, r facilidades são selecionadas aleatoriamente e removidas do layout. Posteriormente, cada uma é realocada na mesma ordem de remoção, sendo colocada no lado que possui o menor número de facilidades. Este procedimento é repetido até que as r facilidades tenham sido realocadas. A Figura 4.4 apresenta a operação de reconstrução para um corredor com $n = 6$, As facilidades 2 e 5 são removidas respectivamente.

4.2.2 Estratégias de Busca Local

Buscas locais são métodos de exploração de vizinhança onde, para uma dada solução inicial, são realizadas sucessivas modificações, em busca de melhores soluções. A cada iteração, a busca local substitui a solução de entrada por outra de melhor qualidade. A busca é finalizada quando todas as soluções vizinhas são piores que a atual. Quando

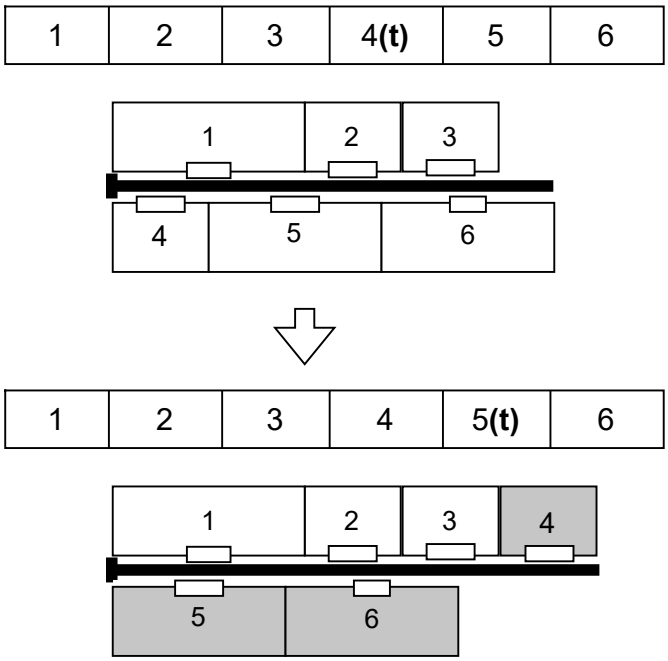


Figura 4.3: Operação *Shift* para $t = 1$ em *layout* de tamanho 6.

esta condição é atingida, entende-se que um ótimo local foi atingido no espaço de busca (TALBI, 2009).

Foi identificado na literatura de forma majoritária, o uso de estratégias de buscas locais através de permutações de facilidades, como *swapLS*, vistos em (AMARAL, 2012), (SANTANA; SANTOS, 2016), (GHOSH; KOTHARI, 2012) e (AHONEN; ALVARENGA; AMARAL, 2014).

Para este trabalho, diferentes estratégias de permutação foram utilizadas. Foi idealizado também um *Random Variable Neighborhood Descent (RVND)*. As buscas locais são definidas como segue:

1. **swapLS:**

   Esta estratégia abrange a busca local clássica, apresentada por (AMARAL, 2012), onde a troca de todos pares de posições do *layout* são testadas. O procedimento é reiniciado quando uma melhora na função objetivo é atingida, mantendo-se a nova solução. A busca local é finalizada quando nenhuma troca é capaz de melhorar o custo do *layout*. Considerando o *layout* representado com um único vetor e um índice p, como apresentado anteriormente, a Figura 4.5 exemplifica o algoritmo *swapLS* aplicado a um *layout* com $n = 6$. A coluna à esquerda da tabela identifica as iterações, e as células destacadas em cada linha identificam as facilidades que

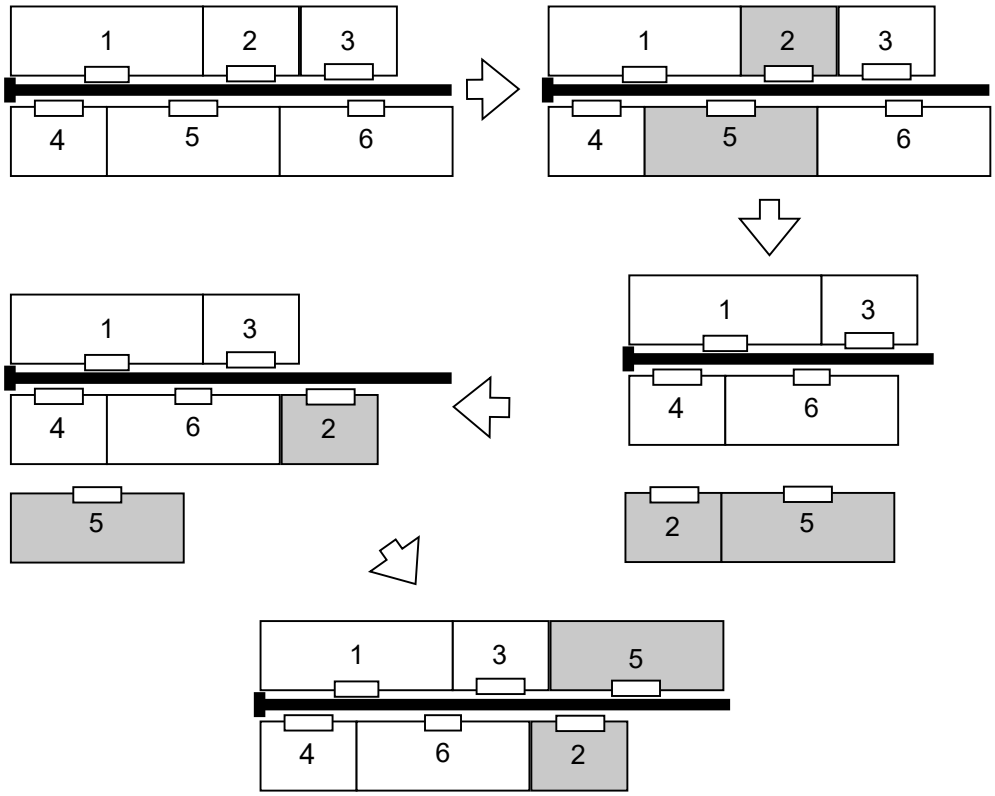


Figura 4.4: Operação *Reconstruction* para $r = 2$ em *layout* de tamanho 6.

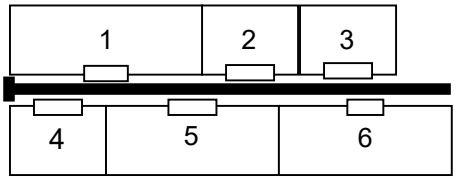
foram trocadas.

2. ***Neighbor swap*:**

   Esta estratégia, bem como as duas subsequentes, representa um subconjunto de movimentos realizados pelo algoritmo *swapLS*. Nesta, é realizada a troca de cada par de facilidades vizinhas no mesmo lado do *layout*. O processo é reiniciado quando uma melhoria é obtida e finalizado quando nenhuma melhoria é alcançada para todos os pares de instalações vizinhas. A Figura 4.6 apresenta um exemplo de sequências de trocas.

3. ***Non-Neighbor swap*:**

   Ao contrário da estratégia anterior, nesta busca local os pares de facilidade de mesmo lado de posição i, j, com $j > (i + 1)$, são trocadas. Ou seja, são realizadas trocas entre facilidades não vizinhas que pertencem ao mesmo lado. O critério de reinício e parada se assemelha às estratégias anteriores. A Figura 4.7 apresenta a estratégia de busca aplicada a um *layout*.



0	1	2	3	4 (p)	5	6
1	2	1	3	4 (p)	5	6
2	3	2	1	4 (p)	5	6
3	4	2	3	1(p)	5	6
4	5	2	3	4 (p)	1	6
5	6	2	3	4 (p)	5	1
6	1	3	2	4 (p)	5	6
7	1	4	3	2(p)	5	6
8	1	5	3	4 (p)	2	6
9	1	6	3	4 (p)	5	2
10	1	2	4	3(p)	5	6
11	1	2	5	4 (p)	3	6
12	1	2	6	4 (p)	5	3
13	1	2	3	5(p)	4	6
14	1	2	3	6(p)	5	4
15	1	2	3	4 (p)	5	6

Figura 4.5: Algoritmo *swapLS* aplicado a um *layout*.

4. ***Opposite-side swap*:**

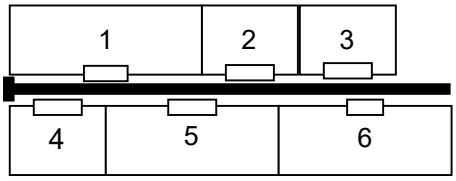
   Para esta estratégia, todo par de facilidades que estejam em lados opostos do *layout* é trocado. Os critérios de parada e de reinício são análogos aos anteriores. A Figura 4.8 apresenta as trocas efetuadas.

5. ***Insertion*:**

   Toda facilidade do *layout* é inserida em todas as possíveis posições do lado oposto, sem que seja realizada uma troca. O critério de parada e de reinício é semelhante ao das estratégias anteriores. A Figura 4.9 apresenta um exemplo da operação de inserções.

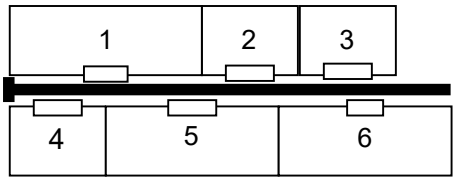
6. **RVND:**

   O conjunto de heurísticas de troca entre vizinhos, a troca entre não vizinhos, a troca entre lados opostos e a inserção é aplicado em ordem aleatória. Neste caso, o RVND é reiniciado quando uma melhoria é observada ao aplicar qualquer um



0	1	2	3	4 (p)	5	6
1	2	1	3	4 (p)	5	6
2	1	3	2	4 (p)	5	6
3	1	2	3	5(p)	4	6
4	1	2	3	4 (p)	6	5

Figura 4.6: Algoritmo de troca entre vizinhos aplicado a um *layout*.



0	1	2	3	4 (p)	5	6
1	3	2	1	4 (p)	5	6
2	1	2	3	6(p)	5	4

Figura 4.7: Algoritmo de troca entre não vizinhos aplicado a um *layout*.

desses quatro operadores de movimento; de forma semelhante aos outros casos, a busca pára quando nenhuma melhoria é observada ao usar os quatro operadores de movimento. O Algoritmo 2 demonstra a aplicação do RVND a uma solução de entrada.

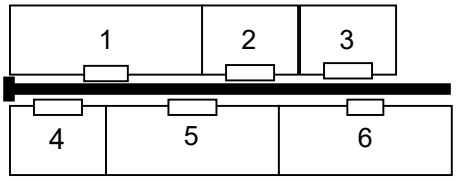
4.3 Gramática Proposta

Um componente importante da Programação Genética é a representação de suas soluções candidatas. O modelo adotado por este trabalho é o de Programação Genética Gramatical com uma gramática livre de contexto para representar as heurísticas candidatas. A gramática apresentada é composta majoritariamente pelas heurísticas de baixo nível apresentadas nas Seções 4.2.1 e 4.2.2.

Baseada na quádrupla (N, T, P, S) apresentada na Seção 2.5, obtemos a seguinte composição:

Algoritmo 2: *Pseudo-código para o RVND*

```
   Input: S
   Output: S''
 1 seq ← randomizar
 2 S'' = S
 3 for i = 1...|seq| do
 4 |   switch seq[i] do
 5 |   |   case 0
 6 |   |   |   S' ← blVizinhos(S)
 7 |   |   endsw
 8 |   |   case 1
 9 |   |   |   S' ← blNaoVizinhos(S)
10 |   |   endsw
11 |   |   case 2
12 |   |   |   S' ← blOpostos(S)
13 |   |   endsw
14 |   |   case 3
15 |   |   |   S' ← blInsercao(S)
16 |   |   endsw
17 |   endsw
18 |   if f(S') < f(S'') then
19 |   |   S'' = S'
20 |   |   i = 1
21 |   end if
22 end for
23 return S'';
```



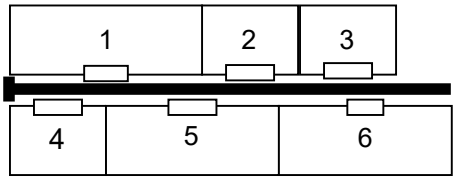
0	1	2	3	4 (p)	5	6
1	4	2	3	1(p)	5	6
2	5	2	3	4 (p)	1	6
3	6	2	3	4 (p)	5	1
4	1	4	3	2(p)	5	6
5	1	5	3	4 (p)	2	6
6	1	6	3	4 (p)	5	2
7	1	2	4	3(p)	5	6
8	1	2	5	4 (p)	3	6
9	1	2	6	4 (p)	5	3

Figura 4.8: Algoritmo de troca facilidades de lados opostos aplicado a um *layout*.

- N={<do>, <pert>, <ref>, <signOp>, <cons>};
- T={shake, swap, shift, reconstruction, nbrSwap, nonNbrSwap, oppSidesSwap, insertion, rvnd, swapLS, "(","")",",", "+", "-", 0.00, 0.05, ..., 1.00};
- S=<do>;
- P:

$$
\begin{aligned}
\text{<do>} &::= \text{<pert>} \mid \text{<ref>} \mid \text{<do>}\ \text{<do>} \\
\text{<pert>} &::= \mathit{shake}(\text{<cons>}) \mid \mathit{swap}(\text{<cons>}, \text{<cons>}) \\
&\quad \mid \mathit{shift}(\text{<signOp>}\ \text{<cons>}) \mid \mathit{reconstruction}(\text{<cons>}) \\
\text{<ref>} &::= \mathit{nbrSwap}() \mid \mathit{nonNbrSwap}() \mid \mathit{oppSidesSwap}() \\
&\quad \mid \mathit{insertion}() \mid \mathit{rvnd}() \mid \mathit{swapLS}(); \\
\text{<signOp>} &::= + \mid - \\
\text{<cons>} &::= 0.00 \mid 0.05 \mid ... \mid 0.95 \mid 1.00.
\end{aligned}
$$

O símbolo inicial <do> gera uma combinação de uma ou mais sequência de es-



0	1	2	3	4 (p)	5	6
1	2	3	1(p)	4	5	6
2	2	3	4(p)	1	5	6
3	2	3	4(p)	5	1	6
4	2	3	4(p)	5	6	1
5	1	3	2(p)	4	5	6
6	1	3	4(p)	2	5	6
7	1	3	4(p)	5	2	6
8	1	3	4(p)	5	6	2
9	1	2	3(p)	4	5	6
10	1	2	4(p)	3	5	6
11	1	2	4(p)	5	3	6
12	1	2	4(p)	5	6	3
13	4	1	2	3	5(p)	6
14	1	4	2	3	5(p)	6
15	1	2	4	3	5(p)	6
16	1	2	3	4	5(p)	6
17	5	1	2	3	4(p)	6
18	1	5	2	3	4(p)	6
19	1	2	5	3	4(p)	6
20	1	2	3	5	4(p)	6
21	6	1	2	3	4(p)	5
22	1	6	2	3	4(p)	5
23	1	2	6	3	4(p)	5
24	1	2	3	6	4(p)	5

Figura 4.9: Algoritmo de inserção de facilidades lados opostos aplicado a um *layout*.

tratégias de perturbação ou busca local. Cada não-terminal <pert> é substituído por uma chamada a uma única estratégia de perturbação descrita na Seção 4.2.1. Da mesma forma, cada não-terminal <ref> produz uma chamada a uma única busca local apresentada na Seção 4.2.2. O não-terminal <signOp> permite a produção de números negativos, introduzindo os sinais + e − em constantes numéricas. O não-terminal <cons> é substituído por um dos valores reais listados, que representa uma porcentagem do tamanho do problema n.

5 Experimentos Computacionais

Este capítulo abrange os experimentos realizados, bem como a análise das heurísticas desenvolvidas. Como mencionado no Capítulo 1, um subconjunto de instâncias do problema foi selecionado a fim de compor o conjunto de instâncias de teste. Aquelas selecionadas foram utilizadas para gerar diferentes variantes de heurísticas geradas por PGG. Uma outra variante foi desenvolvida utilizando todas as instâncias de teste simultaneamente. Ao final, para fins de comparação, cada variante gerada é analisada ao ser executada em todas as instâncias do problema.

A Seção 5.1 apresenta as instâncias do problema, bem como aquelas selecionadas para treino. As heurísticas desenvolvidas pela HPGG são analisadas na Seção 5.2. Por fim, na Seção 5.3, a análise estatística dos resultados é realizada.

5.1 Instâncias do Problema

Na literatura, o CAP é abordado com um conjunto frequente de 89 instâncias, desenvolvidas por diversos autores, com $9 \leq n \leq 70$, o número de facilidades. São elas:

- $S\alpha$, $\alpha \in \{9, 9H, 10, 11\}$ (SIMMONS, 1969), 4 instâncias com $n = \{9, 10, 11\}$;
- $Am\beta$, $\beta \in \{12a, 12b, 13a, 13b, 15\}$ (AMARAL, 2012), 5 instâncias com $n = \{12, 13, 15\}$;
- $N\gamma.\delta$, $\gamma \in \{25, 30, 40\}$, $1 \leq \delta \leq 5$ (ANJOS; VANNELLI, 2008), 15 instâncias com $n = \{25, 30, 40\}$;
- $Sko.\epsilon.\zeta$, $\epsilon \in \{42, 49, 56\}$, $1 \leq \zeta \leq 5$ e $Sko.64.5$ (AHONEN; ALVARENGA; AMARAL, 2014), 16 instâncias com $n = \{42, 49, 56, 64\}$;
- $P\theta$,$1 \leq \theta \leq 6$ (AMARAL, 2008), 6 instâncias com $n = 33, 35$;
- $ste36.0\eta$, $1 \leq \eta \leq 6$, (ANJOS; YEN, 2009), 5 instâncias com $n = 36$;
- $CAP.\iota.\kappa.\lambda$, $\iota = \{30, 60, 90\}$, $\kappa = \{30, 40, 50, 60\}$, $1 \leq \lambda \leq 3$ (AHONEN; ALVARENGA; AMARAL, 2014), 36 instâncias com $n = 60$;

- $AKV.\mu.05$, $\mu \in \{60, 70\}$ (ANJOS; KENNINGS; VANNELLI, 2005), 2 instâncias com $n = 60, 70$.

5.1.1 Instâncias de Treino

Como dito anteriormente, dez instâncias do problema foram selecionadas para compor o conjunto de instâncias de treino. O critério de escolha foi seu amplo uso na literatura e a heterogeneidade de sua configuração. As características das instâncias são detalhadas no Apêndice A. São elas:

1. S9 (SIMMONS, 1969), com $n = 9$ (Tabela A.1);
2. S10 (SIMMONS, 1969), com $n = 10$ (Tabela A.2);
3. S11 (SIMMONS, 1969), com $n = 11$ (Tabela A.3);
4. Am15 (AMARAL, 2012), com $n = 15$ (Tabela A.4);
5. N25.01 (ANJOS; VANNELLI, 2008), com $n = 25$ (Tabela A.5);
6. Sko42.01 (AHONEN; ALVARENGA; AMARAL, 2014), com $n = 42$ (Tabela A.6);
7. Sko42.02 (AHONEN; ALVARENGA; AMARAL, 2014), com $n = 42$ (Tabela A.7);
8. Sko42.03 (AHONEN; ALVARENGA; AMARAL, 2014), com $n = 42$ (Tabela A.8);
9. Sko42.04 (AHONEN; ALVARENGA; AMARAL, 2014), com $n = 42$ (Tabela A.9);
10. Sko49.01 (AHONEN; ALVARENGA; AMARAL, 2014), com $n = 49$ (Tabela A.10);

5.2 Execução dos Experimentos

Como dito anteriormente, a execução do experimento se dá em duas etapas. Na primeira, os agentes de cada variante são desenvolvidos através da PGG proposta. São geradas um total de onze variantes. Destas, dez foram geradas considerando como instância de treino cada um dos respectivos problemas apresentados na Seção 5.1.1. Uma última variante, denominada Variante Geral, foi gerada a partir do treino com todas as dez instâncias selecionadas.

A PGG foi parametrizada com uma população de 100 indivíduos sendo evoluídos por 100 gerações. A profundidade da árvore sintática foi limitada em 10 níveis. As probabilidades de recombinação e mutação foram definidas em 90% e 10%, respectivamente. Uma taxa de 5% dos melhores indivíduos são mantidos na população ao longo das gerações.

Na segunda etapa, o melhor agente de cada variante é avaliado a partir de 30 execuções de todas as instâncias do problema. Para ambas as etapas, uma execução do ponto de vista do algoritmo apresentado é representada por 50 laços da heurística desenvolvida em uma instância.

Os experimentos foram executados em um computador com processador AMD Ryzen 5 2600 3.40GHz, 16.0GB de memória RAM DDR4-2133GHZ com um sistema operacional Ubuntu 22.04 LTS. O código fonte, implementado em C++11, bem como as instâncias do problema e os agentes desenvolvidos estão disponíveis publicamente[1].

5.3 Análise dos Resultados

As análises foram feitas em relação aos melhores resultados obtidos por cada variante em cada instância, nas Tabelas 5.1 e 5.2, bem como os custos médios, nas Tabelas 5.3 e 5.4. Para ambas as tabelas, as colunas representam as variantes e seus respectivos resultados. Os melhores resultados são mantidos em negrito. Para a tabela de custos médios, o teste não paramétrico de Kruskal-Wallis (MCKIGHT; NAJAB, 2010) foi realizado para avaliar se os resultados com os melhores valores médios são estatisticamente melhores do que os demais. Os resultados estatisticamente similares aos melhores são marcados com um asterisco ($*$).

Por fim, a Tabela 5.5 apresenta o *score* de cada variante considerando todas as instâncias.

De acordo com esses resultados, observa-se que a variante A_{Geral} encontrou os melhores resultados na maioria das instâncias (30 das 89), a variante $A_{Sko42.03}$ em 28 instâncias e a variante $A_{Sko49.01}$ em 26 instâncias. Além disso, a variante A_{Geral} obteve a melhor média de resultados na maioria dos problemas (30 das 89), e a variante $A_{Sko42.01}$

[1] https://github.com/rafaelfreesz/capPG.git

Tabela 5.1: Melhores soluções para instâncias de tamanho $n \leq 56$.

Instancia	Agente										
	A_{S9}	A_{S10}	A_{S11}	A_{Am15}	$A_{N25.01}$	$A_{Sko42.01}$	$A_{Sko42.02}$	$A_{Sko42.03}$	$A_{Sko42.04}$	$A_{Sko49.01}$	A_{Geral}
S9	**1181.5**	**1181.5**	**1181.5**	**1181.5**	**1181.5**	**1181.5**	**1181.5**	**1181.5**	**1181.5**	**1181.5**	**1181.5**
S9H	**2294.5**	**2294.5**	**2294.5**	2329.5	**2294.5**	**2294.5**	**2294.5**	**2294.5**	**2294.5**	**2294.5**	**2294.5**
S10	**1374.5**	**1374.5**	**1374.5**	**1374.5**	**1374.5**	**1374.5**	**1374.5**	**1374.5**	**1374.5**	**1374.5**	**1374.5**
S11	3466.5	**3439.5**	**3439.5**	3479.5	**3439.5**	**3439.5**	**3439.5**	**3439.5**	**3439.5**	**3439.5**	**3439.5**
Am12a	**1529.0**	**1529.0**	**1529.0**	**1529.0**	**1529.0**	**1529.0**	**1529.0**	**1529.0**	**1529.0**	**1529.0**	**1529.0**
Am12b	**1609.5**	**1609.5**	**1609.5**	**1609.5**	**1609.5**	**1609.5**	**1609.5**	**1609.5**	**1609.5**	**1609.5**	**1609.5**
Am13a	2472.5	**2467.5**	**2467.5**	**2467.5**	**2467.5**	**2467.5**	**2467.5**	**2467.5**	**2467.5**	**2467.5**	**2467.5**
Am13b	2871.0	**2870.0**	**2870.0**	**2870.0**	**2870.0**	**2870.0**	**2870.0**	**2870.0**	**2870.0**	**2870.0**	**2870.0**
Am15	3207.0	**3195.0**	**3195.0**	**3195.0**	**3195.0**	**3195.0**	**3195.0**	**3195.0**	**3195.0**	**3195.0**	**3195.0**
N25.01	2306.0	**2302.0**	**2302.0**	**2302.0**	**2302.0**	**2302.0**	**2302.0**	**2302.0**	**2302.0**	**2302.0**	**2302.0**
N25.02	18662.5	18600.5	18600.5	18626.5	**18595.5**	18601.5	**18595.5**	18601.5	18601.5	**18595.5**	**18595.5**
N25.03	12195.0	12138.0	12136.0	12138.0	12137.0	12133.0	12128.0	12124.0	12139.0	12138.0	**12116.0**
N25.04	24344.5	**24192.5**	24214.5	24290.5	24196.5	24216.5	24218.5	24216.5	24218.5	24234.5	24214.5
N25.05	7852.0	**7819.0**	**7819.0**	7825.0	**7819.0**	**7819.0**	7827.0	**7819.0**	**7819.0**	**7819.0**	**7819.0**
N30.01	4127.0	**4115**.0	**4115.0**	**4115**.0	**4115.0**	**4115.0**	**4115.0**	**4115.0**	**4115.0**	**4115.0**	**4115.0**
N30.02	10829.5	10790.5	10789.5	10784.5	10784.5	10791.5	10789.5	10793.5	10785.5	10785.5	**10781.5**
N30.03	22725.0	22706.0	22706.0	22711.0	22706.0	22706.0	22713.0	22706.0	22706.0	22708.0	**22702.0**
N30.04	28558.5	**28401.5**	28417.5	28404.5	**28401.5**	**28401.5**	28404.5	**28401.5**	**28401.5**	28404.5	28411.5
N30.05	57752.0	57480.0	57471.0	57456.0	57453.0	57436.0	57481.0	**57424.0**	57447.0	57434.0	57437.0
P1	30422.5	30316.5	**30290.5**	30318.5	30310.5	30303.5	30309.5	**30290.5**	30295.5	30316.5	30312.5
P2	34135.0	34012.0	33991.0	34015.0	33997.0	33980.0	34018.0	**33976.0**	33986.0	33979.0	33985.0
P3	35131.5	35099.5	35081.5	35093.5	35080.5	35073.5	35098.5	**35062.5**	35074.5	35081.5	35068.5
P4	34812.5	**34667.5**	34677.5	34701.5	34670.5	34675.5	34675.5	34668.5	34679.5	34680.5	34675.5
P5	30880.0	30812.0	30791.0	30826.0	30814.0	30806.0	30801.0	30830.0	30803.0	**30786.0**	30787.0
P6	34506.5	34437.5	34437.5	34447.5	**34424.5**	34441.5	34447.5	34441.5	34437.5	**34424.5**	34438.5
ste36.01	5182.0	**4966.0**	**4966.0**	5486.0	5003.0	5014.0	5003.0	**4966.0**	**4966.0**	5003.0	**4966.0**
ste36.02	90682.0	**88382.0**	88431.0	89270.0	**88382.0**	88977.0	89041.0	88819.0	89152.0	88939.0	88466.0
ste36.03	51777.5	50824.5	50727.5	51362.5	50686.5	50529.5	**50426.5**	50688.5	50616.5	50703.5	50713.5
ste36.04	49093.5	**46884.5**	47306.5	48059.5	47224.5	47156.5	47007.5	47193.5	46940.5	47102.5	46992.5
ste36.05	46537.5	45183.5	45030.5	45398.5	44710.5	44964.5	44804.5	44688.5	44619.5	**44565.5**	44974.5
N40.01	53943.5	53746.5	53769.5	53858.5	53755.5	53748.5	53743.5	53745.5	53753.5	53764.5	**53740.5**
N40.02	49082.0	48932.0	48942.0	48973.0	48924.0	48954.0	48942.0	48924.0	**48908.0**	48923.0	48925.0
N40.03	39357.5	39282.5	**39260.5**	39270.5	39280.5	39264.5	39275.5	39266.5	39261.5	39265.5	39263.5
N40.04	38485.0	38381.0	38380.0	38381.0	38361.0	**38355.0**	38380.0	38379.0	38384.0	38377.0	38383.0
N40.05	51688.0	51529.0	51511.0	51544.0	**51505.0**	51532.0	51532.0	**51505.0**	51515.0	51518.0	51544.0
Sko42.01	12760.0	**12731.0**	**12731.0**	12741.0	**12731.0**	**12731.0**	**12731.0**	**12731.0**	**12731.0**	**12731.0**	**12731.0**
Sko42.02	108359.5	108055.5	108050.5	108083.5	108082.5	108059.5	108049.5	108053.5	108069.5	**108043.5**	108071.5
Sko42.03	86921.5	**86667.5**	86677.5	86744.5	86676.5	86683.5	86700.5	86672.5	86678.5	86704.5	86684.5
Sko42.04	69111.0	68785.0	68842.0	68808.0	68780.0	68771.0	68767.0	**68753.0**	68795.0	68811.0	68771.0
Sko42.05	124313.5	124111.5	124064.5	124105.5	124088.5	124118.5	**124055.5**	124095.5	124098.5	124123.5	124075.5
Sko49.01	20506.0	20478.0	20482.0	20478.0	20478.0	20478.0	20482.0	20478.0	20482.0	**20470.0**	**20470.0**
Sko49.02	208910.0	208159.0	208241.0	208450.0	208217.0	208176.0	208179.0	208242.0	208192.0	**208157.0**	208160.0
Sko49.03	162517.0	162430.0	162332.0	162631.0	162299.0	162349.0	**162238.0**	162389.0	162405.0	162273.0	162344.0
Sko49.04	118556.5	118345.5	118315.5	118370.5	118324.5	118376.5	**118298.5**	118337.5	118319.5	118306.5	118307.5
Sko49.05	333418.0	**332879.0**	332941.0	332946.0	333038.0	332943.0	333016.0	332953.0	332936.0	333020.0	333033.0
Sko56.01	32031.0	31983.0	31983.0	31984.0	31983.0	**31977.0**	31986.0	31985.0	31982.0	31984.0	31983.0
Sko56.02	248791.0	248334.0	248311.0	248628.0	248329.0	248310.0	248309.0	248286.0	248301.0	248285.0	**248279.0**
Sko56.03	85472.0	85215.0	85223.0	85318.0	85212.0	85242.0	85227.0	85221.0	85214.0	**85210.0**	85218.0
Sko56.04	157052.0	156770.0	156701.0	156804.0	156730.0	156707.0	156746.0	156812.0	156782.0	**156684.0**	156758.0
Sko56.05	296678.5	296283.5	296347.5	296428.5	296304.5	296350.5	296280.5	296298.5	**296267.5**	296316.5	296270.5

em 19 instâncias. Assim, a variante A_{Geral} apresentou os melhores valores na maioria dos casos. Um ponto importante a ser destacado é que a variante $A_{Sko42.01}$ utiliza uma única instância para gerar as heurísticas, mas também é capaz de alcançar resultados competitivos quando comparada à variante A_{Geral}.

5.3.1 Perfis de Desempenho

Os Perfis de Desempenho (PD) (DOLAN; MORÉ, 2002) são usados para analisar o desempenho relativo das técnicas de busca quando o conjunto de problemas é grande. Para um conjunto S de resolvedores s_i, $i \in 1, \ldots, n_s$, e um conjunto P de problemas/instâncias

Tabela 5.2: Melhores soluções para instâncias de tamanho $n \geq 60$.

Instancias	Agentes										
	A_{S9}	A_{S10}	A_{S11}	A_{Am15}	$A_{N25.01}$	$A_{Sko42.01}$	$A_{Sko42.02}$	$A_{Sko42.03}$	$A_{Sko42.04}$	$A_{Sko49.01}$	A_{Geral}
AKV.60.05	161497.0	159805.0	159806.0	160093.0	159777.0	159771.0	159823.0	159789.0	159766.0	**159761.0**	159763.0
CAP.30.30.1	205190.0	204308.0	204277.0	205459.0	204346.0	204293.0	204308.0	204306.0	**204245.0**	204451.0	204287.0
CAP.30.30.2	194606.5	193619.5	193469.5	193457.5	193466.5	193543.5	193460.5	193492.5	193430.5	193433.5	**193407.5**
CAP.30.30.3	162627.5	161735.5	161702.5	161744.5	161724.5	161705.5	**161600.5**	161779.5	161717.5	161634.5	161793.5
CAP.30.40.1	135977.5	135409.5	**135221.5**	135319.5	135330.5	135338.5	135291.5	135345.5	135312.5	135390.5	135294.5
CAP.30.40.2	160655.0	**159261.0**	159359.0	159388.0	159299.0	159368.0	159338.0	159331.0	159331.0	159357.0	159324.0
CAP.30.40.3	160339.5	159331.5	159188.5	159402.5	159149.5	159172.5	159116.5	**159093.5**	159406.5	159143.5	159103.5
CAP.30.50.1	111618.5	110816.5	110715.5	110983.5	**110617.5**	110898.5	110768.5	110840.5	110758.5	110732.5	110736.5
CAP.30.50.2	115951.0	115533.0	115503.0	115775.0	115507.0	**115439.0**	115486.0	115448.0	115489.0	115480.0	115513.0
CAP.30.50.3	115412.0	114455.0	114340.0	114749.0	114375.0	114401.0	114424.0	114365.0	114401.0	114406.0	**114337.0**
CAP.30.60.1	108788.0	108153.0	108203.0	**108123.0**	108241.0	108192.0	108214.0	108194.0	108168.0	108221.0	108184.0
CAP.30.60.2	110739.5	110124.5	110079.5	110512.5	110115.5	**110036.5**	110088.5	110094.5	110099.5	110091.5	110049.5
CAP.30.60.3	92510.0	92009.0	91920.0	92259.0	91967.0	91978.0	91891.0	**91801.0**	91996.0	91845.0	91882.0
CAP.60.30.1	446198.5	445655.5	445448.5	446109.5	445580.5	**445440.5**	445493.5	445687.5	445548.5	445609.5	445473.5
CAP.60.30.2	409318.5	408056.5	408005.5	408426.5	408020.5	408011.5	**407953.5**	408073.5	408034.5	408018.5	408051.5
CAP.60.30.3	417897.5	417045.5	417027.5	417067.5	417117.5	417054.5	417070.5	417059.5	417037.5	417068.5	**417003.5**
CAP.60.40.1	313945.0	313416.0	313381.0	313435.0	313472.0	313399.0	313432.0	313392.0	313384.0	313403.0	**313371.0**
CAP.60.40.2	321370.5	320875.5	320827.5	**320785.5**	320953.5	320815.5	320818.5	320832.5	320811.5	320829.5	320807.5
CAP.60.40.3	363827.5	363216.5	363173.5	363218.5	**363088.5**	363169.5	363219.5	363255.5	363288.5	363335.5	363137.5
CAP.60.50.1	274006.0	273587.0	273542.0	273576.0	273566.0	273548.0	273532.0	273575.0	273527.0	**273518.0**	273527.0
CAP.60.50.2	270218.5	269908.5	269838.5	269834.5	269886.5	269817.5	**269782.5**	269860.5	269821.5	269842.5	269917.5
CAP.60.50.3	295968.0	295569.0	295567.0	295568.0	295569.0	295536.0	295507.0	**295501.0**	295516.0	295567.0	295552.0
CAP.60.60.1	228230.0	227981.0	227979.0	228133.0	227973.0	227953.0	227978.0	228012.0	227980.0	227992.0	**227945.0**
CAP.60.60.2	247057.0	246712.0	**246625.0**	246978.0	246629.0	246659.0	246651.0	246707.0	246646.0	246655.0	246628.0
CAP.60.60.3	206959.5	206609.5	206603.5	206923.5	206593.5	**206576.5**	206604.5	206591.5	206624.5	206589.5	206635.5
CAP.90.30.1	629500.0	628957.0	628965.0	629079.0	629026.0	629059.0	628994.0	629075.0	628983.0	628958.0	**628934.0**
CAP.90.30.2	561801.5	561306.5	561269.5	561483.5	561275.5	561287.5	561245.5	561335.5	561306.5	561217.5	**561212.5**
CAP.90.30.3	588096.5	587861.5	587856.5	587908.5	587869.5	587841.5	587854.5	587849.5	587878.5	**587838.5**	587852.5
CAP.90.40.1	474373.0	474130.0	474096.0	474135.0	474138.0	474127.0	**474066.0**	474153.0	474098.0	474083.0	474088.0
CAP.90.40.2	480865.0	480011.0	**480006.0**	480469.0	480135.0	480034.0	480015.0	480243.0	480051.0	480044.0	480022.0
CAP.90.40.3	513048.0	512492.0	512483.0	512555.0	512523.0	512480.0	512529.0	**512475.0**	512493.0	512510.0	512509.0
CAP.90.50.1	480630.0	479820.0	**479705.0**	479835.0	479787.0	479780.0	479778.0	479780.0	479749.0	479742.0	479768.0
CAP.90.50.2	445505.0	445201.0	445151.0	445304.0	445128.0	445163.0	445099.0	445183.0	445140.0	**445094.0**	445105.0
CAP.90.50.3	495349.5	495153.5	495084.5	495245.5	495109.5	495097.5	495094.5	495138.5	495078.5	495122.5	**495054.5**
CAP.90.60.1	385720.5	385449.5	385450.5	385469.5	385467.5	385445.5	385455.5	**385441.5**	385445.5	385454.5	385456.5
CAP.90.60.2	344880.0	344788.0	344804.0	344817.0	344840.0	344837.0	344802.0	344811.0	344812.0	344808.0	**344784.0**
CAP.90.60.3	411582.0	411322.0	411268.0	411462.0	411323.0	**411259.0**	411273.0	411283.0	411280.0	411267.0	411284.0
Sko64.05	251144.5	250950.5	250923.5	250913.5	251014.5	250913.5	250942.5	**250904.5**	250912.5	250905.5	250924.5
AKV.70.05	2120309.5	2110546.5	2111485.5	2117174.5	2110405.5	2111787.5	2111356.5	**2110367.5**	2110894.5	2111087.5	2110528.5

p_j, $j \in 1, \ldots, n_p$, $t_{p,s}$ pode ser definido como uma medida de desempenho para o método $s \in S$ ao resolver o problema $p \in P$. A razão de desempenho

$$r_{p,s} = \frac{t_{p,s}}{\max t_{p,s} : s \in S}$$

é o desempenho relativo do método $s \in S$. A probabilidade de que a razão de desempenho $r_{p,s}$ do método s esteja dentro de um fator $\tau > 0$ da melhor razão observada para todas as técnicas em S pode ser definida como $\rho_s(\tau) = \frac{1}{n_p}, |p \in P : r_{p,s} \leq \tau|$, onde $\rho_s(\tau)$ denota as curvas dos PD.

A partir dos PDs (BARBOSA; BERNARDINO; BARRETO, 2010), podem ser identificados:

1. a abordagem que obteve os melhores resultados para a maioria dos problemas (maior $\rho(1)$);

2. a abordagem mais confiável (menor τ tal que $\rho(\tau) = 1$);

Tabela 5.3: Solução média para instâncias de tamanho $n \leq 56$.

Instancia	Agente										
	A_{S9}	A_{S10}	A_{S11}	A_{Am15}	$A_{N25.01}$	$A_{Sko42.01}$	$A_{Sko42.02}$	$A_{Sko42.03}$	$A_{Sko42.04}$	$A_{Sko49.01}$	A_{Geral}
S9	1190.13	**1181.50***	**1181.50***	**1181.50***	**1181.50***	**1181.50***	**1181.50***	**1181.50***	**1181.50***	**1181.50***	**1181.50***
S9H	2298.70	2295.00∗	2295.00∗	2329.50	2295.10	**2294.60***	2295.20	2294.80∗	2295.00∗	**2294.60***	**2294.60***
S10	1386.47	**1374.50***	1374.77∗	1378.07	1374.63∗	1375.53	**1374.50***	1374.63∗	**1374.50***	**1374.50***	1374.90∗
S11	3524.67	3451.87∗	3448.30∗	3489.03	3453.70∗	3445.33∗	3449.27∗	3447.10∗	3450.73∗	3448.87∗	**3445.27***
Am12a	1531.57∗	1531.90∗	1533.20∗	1532.90∗	1531.73∗	1534.00	1533.50∗	1533.73∗	1533.57∗	1533.03∗	**1531.30***
Am12b	1661.87	**1612.10***	1619.97	1617.53∗	1612.27∗	1620.03	1622.13	1616.47∗	1618.77∗	1627.13	1614.87∗
Am13a	2485.83	2467.70∗	2467.80∗	2469.50	2467.70∗	2467.93∗	2467.90∗	**2467.67***	2467.70∗	2467.70∗	2468.00∗
Am13b	2904.93	2870.50∗	2870.83∗	2878.77	2871.87∗	2871.17∗	2872.00∗	2870.60∗	2871.50∗	2872.53∗	**2870.47***
Am15	3243.53	3201.83∗	3205.57	3205.13∗	3206.70	3204.00∗	3205.50∗	3204.67	3207.70	3208.33	**3199.90***
N25.01	2317.60	2308.20	2305.97	2310.47	2306.63	**2302.83***	2306.67	2305.87	2306.13	2307.37	2303.60∗
N25.02	18801.27	18647.67∗	18651.27∗	18659.97	**18635.77***	18643.53∗	18652.47∗	18652.47	18650.87∗	18651.33∗	18639.47∗
N25.03	12300.70	12169.10	12167.47	12187.67	12167.80∗	12164.77∗	12167.37	12166.23	12165.47	12163.50∗	**12157.57***
N25.04	24471.57	24273.43∗	24306.13	24319.33	24276.60∗	24294.50	24300.60	24306.73	24299.83	24308.23	**24272.43***
N25.05	7909.60	**7834.87***	7844.70	7847.67	7836.73∗	7843.87	7845.87	7842.03	7838.23∗	7839.33∗	7835.67∗
N30.01	4162.20	4115.87∗	4116.83∗	4117.73	4116.70	4116.03∗	4116.07∗	4116.33∗	4116.03∗	4117.27	**4115.80***
N30.02	10901.33	10805.13∗	10813.33	**10800.33***	10803.97	10809.03	10812.63	10812.27	10808.60	10809.63	10805.07∗
N30.03	22883.53	22723.70∗	22728.53∗	22745.40	22725.93∗	22723.20∗	22733.50	22729.40∗	**22722.90***	22725.03∗	22723.73∗
N30.04	28814.80	28459.87	28456.27	28503.73	28456.57∗	**28439.90***	28454.43∗	28460.57	28442.90∗	28454.03	28447.10∗
N30.05	58078.23	57627.63∗	57652.83	57605.00∗	57619.00∗	57591.70∗	57623.73∗	57659.13	**57588.90***	57635.30∗	57596.13∗
P1	30519.27	30383.37	30360.17∗	30406.27	30374.67	30368.07∗	**30356.07***	30367.67∗	30363.10∗	30368.33∗	30359.87∗
P2	34282.67	34073.17	34061.27∗	34137.43	34069.27	**34042.73***	34064.97∗	34057.47∗	34043.60∗	34054.50∗	34048.27∗
P3	35415.90	35161.70	35146.23∗	35344.20	35149.57∗	**35139.50***	35165.17∗	35157.50	35175.87∗	35142.40∗	35143.53∗
P4	34980.63	34734.17	34738.00	34791.20	34745.93	**34719.37***	34752.67	34757.73	34754.97	34736.83∗	34725.67∗
P5	30976.63	30873.00	30866.87∗	30883.47	30874.07	30857.23∗	30864.03∗	30860.00∗	30862.00∗	30863.73∗	**30856.60***
P6	34620.77	34478.93∗	34486.27	34504.47	**34468.60***	34471.43∗	34485.47	34474.97∗	34477.73∗	34473.23∗	34470.27∗
ste36.01	5324.90	5133.00	5137.33	5541.00	5106.33∗	5140.83	5160.03	**5088.03***	5151.23	5156.90	5122.57∗
ste36.02	96169.23	91018.17	90937.17	93826.20	90430.40∗	90878.30	90557.47∗	**90201.60***	90697.57	90759.97	90303.13∗
ste36.03	53853.77	51531.33∗	51525.97	53319.37	51470.70∗	51576.93	51640.00	**51301.17***	51519.07	51578.70	51392.97∗
ste36.04	50855.73	48409.63∗	48561.90∗	49995.83	48237.17∗	48414.50∗	48912.67	48488.63∗	48253.97∗	48537.57∗	**48207.30***
ste36.05	48249.23	46096.47	45943.40	47473.53	46090.00	45888.20	45847.37	**45501.90***	45749.93∗	45883.80	45694.90∗
N40.01	54296.17	53845.97∗	53851.80∗	54633.50	53838.60∗	**53820.07***	53861.43	53864.00	53862.80	53831.77∗	53828.20∗
N40.02	49377.07	49002.27	49006.07	49049.17	48986.70∗	**48980.43***	48994.43	49008.40	48984.83∗	48984.40∗	48990.40∗
N40.03	39534.33	39304.60	39304.00	**39293.77***	39319.93	39307.77	39302.53∗	39298.00∗	39306.17	39301.80∗	39301.50∗
N40.04	38698.77	38429.20∗	38420.10∗	38478.90	**38413.77***	38427.57∗	38427.77∗	38442.00	38424.57∗	38419.93∗	38419.33∗
N40.05	51948.33	51595.80	51589.50∗	51969.03	51578.93∗	**51569.97***	51584.83∗	51589.93∗	51574.07∗	51576.40∗	51576.50∗
Sko42.01	12859.70	12744.13	12747.87	12760.17	12744.30	**12734.63***	12747.20	12745.00	12742.77	12746.17	12742.40
Sko42.02	108600.90	108168.43∗	108158.10∗	108221.67	108189.77	108156.20∗	108175.20∗	108180.50	**108141.43***	108160.07∗	108158.73∗
Sko42.03	87285.23	86870.80	86836.77∗	86938.43	86848.17	86836.50∗	86847.43	86820.17∗	86835.23∗	86840.40∗	**86794.50***
Sko42.04	69315.93	68900.20	68926.73	69046.47	68887.17∗	68883.43∗	68888.47∗	68919.23	68905.10	68890.60∗	**68876.23***
Sko42.05	125046.60	124250.27∗	124235.20∗	124584.53	124227.60∗	124236.40∗	**124198.43***	124225.63∗	124232.80∗	124215.40∗	124225.87∗
Sko49.01	20550.17	20495.20∗	20498.37	20506.90	20496.07∗	**20489.60***	20498.20	20492.67∗	20499.50	20495.77	20489.97∗
Sko49.02	209659.57	208544.57∗	208472.70∗	209527.33	208504.77∗	**208439.67***	208506.77∗	208584.77∗	208461.73∗	208468.00∗	208486.17∗
Sko49.03	163539.87	162603.40	162549.63	163403.00	**162480.70***	162489.23∗	162586.87∗	162604.47	162622.30	162505.20∗	162528.53∗
Sko49.04	118882.17	118503.17	118475.20∗	118649.03	118458.60∗	118521.20	118478.20∗	118468.63∗	118466.17∗	118465.57∗	**118441.57***
Sko49.05	334706.50	333302.10∗	333389.50	333324.07	333316.27	**333156.23***	333312.23	333272.33	333241.77∗	333344.33	333263.97∗
Sko56.01	32163.27	32010.47	32024.03	32010.40	32022.33	**31998.90***	32021.00	32010.50	32022.27	32012.57	32008.10
Sko56.02	249867.20	248603.10	248506.20∗	250451.13	248562.97∗	248474.87∗	248481.67∗	248605.73	248495.03∗	248537.57	**248448.23***
Sko56.03	85670.10	85380.07	85368.07	85574.73	**85309.50***	85355.93	85355.10	85418.57	85339.60∗	85356.00	85321.63∗
Sko56.04	157604.40	156911.97	156878.77∗	157053.57	156908.43	156858.03∗	156915.37	156960.87	156897.97	**156851.63***	156861.17∗
Sko56.05	298502.87	296615.67	296527.70	297572.67	296431.53∗	296473.30∗	296454.97∗	296657.07	296449.63∗	296518.37∗	**296425.97***

3. o melhor desempenho geral (maior área sob as curvas dos PD).

Os PD são apresentados na Figura 5.1, onde pode-se observar que a variante A_{Geral} obteve a melhor média de resultados obtidos na maioria dos problemas (30 das 89 instâncias) e alcançou a maior área sob as curvas dos PD. Além disso, essa variante é a segunda mais confiável, com $\rho(1.006787) = 1$. Por outro lado, a variante que utiliza $A_{Sko42.01}$ obteve a segunda maior área sob as curvas dos PD e é a segunda que encontrou os melhores resultados médios na maioria dos problemas (19 instâncias). No entanto, essa variante alcançou $\rho(\tau) = 1$ apenas quando $\tau = 1.010377$. $A_{Sko42.03}$ é a segunda que encontrou os melhores resultados na maioria dos problemas (28 instâncias) e obteve a terceira maior área sob as curvas dos PD. Também é a mais confiável, alcançando $\rho(1.005836) = 1$ com o menor valor de τ.

Tabela 5.4: Solução média para instâncias de tamanho $n \geq 60$.

Instancia	Agente										
	A_{S9}	A_{S10}	A_{S11}	A_{Am15}	$A_{N25.01}$	$A_{Sko42.01}$	$A_{Sko42.02}$	$A_{Sko42.03}$	$A_{Sko42.04}$	$A_{Sko49.01}$	A_{Geral}
AKV.60.05	162939.60	159999.80	160179.10	160744.27	**159921.13***	160074.70	160130.03	160135.83	160140.57	160029.47∗	160091.30
CAP.30.30.1	207162.30	205051.17	204863.30	206542.60	204999.87	**204576.83***	204929.57	205068.97	204911.30	204986.37	204772.20∗
CAP.30.30.2	195460.10	194030.90	193900.43	194614.80	194079.47	193823.63∗	193859.70∗	193881.43∗	193886.37∗	193933.73	**193768.87***
CAP.30.30.3	164022.00	162506.00	162422.77∗	162512.93	162544.47	162372.73∗	**162210.20***	162469.47∗	162332.20∗	162309.70∗	162277.60∗
CAP.30.40.1	136956.50	135785.37	135588.13∗	135722.00	135732.60	**135527.10***	135666.40	135720.37	135738.73	135658.30	135650.23
CAP.30.40.2	161618.30	160002.03	159843.53	160161.80	159844.93	159807.57	159739.83	159893.33	159622.87∗	159589.13∗	**159567.23***
CAP.30.40.3	161196.80	159935.70∗	159793.93∗	160322.70	159988.40	**159765.03***	159939.07∗	159904.97∗	159940.80∗	159971.83∗	159799.60∗
CAP.30.50.1	112343.10	111329.53	111118.73∗	111800.37	111182.33∗	111228.87	111134.13∗	111366.83	111080.13∗	111092.13∗	**111079.37***
CAP.30.50.2	117288.60	115826.73	115683.77∗	116616.03	115860.43	**115623.83***	115682.00∗	115761.67	115673.97∗	115683.40∗	115643.17∗
CAP.30.50.3	116124.80	114814.17	114676.70∗	115722.90	114727.87∗	114680.40∗	114689.40∗	114756.63∗	114675.53∗	114682.27∗	**114672.93***
CAP.30.60.1	109472.33	108382.90	108351.07∗	108435.33	108377.83	108412.33	108354.10∗	108405.63	108315.27∗	108372.67	**108309.13***
CAP.30.60.2	111516.47	110608.17	110444.03	111374.73	110485.23	110307.13∗	110399.27	110550.90	110407.03	110425.87	**110273.40***
CAP.30.60.3	93209.17	92382.60	92328.00	92753.77	92402.47	92228.00∗	92343.53	92306.43∗	92311.37∗	92269.00∗	**92199.77***
CAP.60.30.1	447474.50	446211.87	445945.07∗	447057.63	446015.43∗	446034.50	**445871.57***	446342.50	445994.97∗	445975.77∗	445940.83∗
CAP.60.30.2	410759.47	409007.93	408500.33∗	409964.13	408577.53∗	408628.83∗	**408370.97***	408921.50	408527.10∗	408506.60∗	408382.00∗
CAP.60.30.3	419055.23	417273.73	417223.90∗	418315.77	417328.53	417215.10∗	417234.13∗	417344.80	417222.70∗	417203.77∗	**417200.13***
CAP.60.40.1	314557.63	313692.57	**313570.93***	313627.20∗	313823.30	313678.73∗	313628.70∗	313605.03∗	313633.20	313602.87∗	313670.33∗
CAP.60.40.2	321860.13	321150.20	320995.47∗	321221.27	321154.00	321051.93	321007.07∗	321221.20	321036.00∗	**320981.47***	321031.00∗
CAP.60.40.3	365073.17	363750.37	**363517.97***	364387.33	363707.83	363655.43∗	363555.27∗	363941.17	363518.80∗	363554.10∗	363554.57∗
CAP.60.50.1	274682.83	273846.50	**273701.50***	274255.80	274008.53	273718.43∗	273738.23∗	273853.00	273713.07∗	273774.33∗	273713.20∗
CAP.60.50.2	271264.37	270321.37	270143.80∗	270296.60	270344.13	270150.67∗	**270091.00***	270331.30	270131.23∗	270202.87∗	270145.33∗
CAP.60.50.3	296749.93	295740.97∗	295701.00∗	295927.17	295765.20	295697.17∗	295699.43∗	295751.20∗	295710.10∗	295713.87∗	**295680.60***
CAP.60.60.1	228867.57	228139.63	228123.20	228277.07	228095.60∗	228103.77∗	228121.30	228207.07	228130.87	228123.67	**228078.37***
CAP.60.60.2	248104.40	247097.93	246938.20∗	247630.23	246971.50	247027.80	246912.50∗	247280.03	246937.77	246908.63∗	**246820.03***
CAP.60.60.3	207583.23	206804.07∗	**206768.13***	207379.73	206826.23∗	206783.97∗	206798.10∗	206833.33∗	206890.50	206830.93∗	206825.80∗
CAP.90.30.1	630429.30	629220.70∗	629185.90∗	629529.83	629251.43	629247.03	629195.00∗	629286.20	629177.57∗	629177.90∗	**629142.10***
CAP.90.30.2	563100.30	561650.43	**561466.03***	562536.37	561665.93	561586.67∗	561530.80∗	561810.50	561514.87∗	561524.87∗	561498.57∗
CAP.90.30.3	589359.93	588073.87	587966.33∗	588880.40	588055.00	588138.83	**587953.70***	588138.67	587958.67∗	587979.33∗	587996.00
CAP.90.40.1	475573.90	474496.43	474320.97∗	474865.40	474556.80	474293.53∗	**474289.77***	474716.03	474321.40∗	474373.03∗	474304.17∗
CAP.90.40.2	481478.90	480485.27∗	480518.70∗	481098.93	480627.67	480522.00∗	480462.67∗	480717.33	480502.43∗	**480453.37***	480465.67∗
CAP.90.40.3	514154.07	512995.20∗	**512756.47***	513583.37	512970.87	513009.33∗	512811.40∗	513156.07	512853.23∗	512798.73∗	512767.67∗
CAP.90.50.1	481386.33	480068.30	479966.07∗	480797.57	480012.37	479994.97∗	479991.73∗	480114.27	480036.60∗	**479954.70***	480016.97∗
CAP.90.50.2	446432.77	445620.50	445491.23	446184.73	445475.40∗	445510.17	445466.13∗	445652.47	**445383.63***	445480.70∗	445463.17∗
CAP.90.50.3	496209.43	495570.03	**495402.27***	495964.43	495464.40∗	495516.73∗	495458.13∗	495556.37∗	495510.97∗	495521.30∗	495442.97∗
CAP.90.60.1	386222.77	385601.97	385551.83∗	386057.43	385597.70	385537.30∗	385552.27∗	385609.93∗	385550.27∗	385562.90∗	**385537.07***
CAP.90.60.2	345175.43	**344852.13***	344868.20∗	344857.67∗	344935.03	344872.20	344868.93∗	344853.77∗	344867.93	344874.80	344875.77
CAP.90.60.3	412100.47	411474.10∗	411438.80∗	411667.37	411469.53	411470.83	411424.70∗	411504.10	411420.43∗	**411414.43***	411434.80∗
Sko64.05	251745.90	251088.13	251095.43	251080.70	251291.40	251021.83∗	251053.60	251145.13	251036.50∗	251037.53∗	**250997.17***
AKV.70.05	2124248.43	2112045.67	2114540.07	2119937.77	**2111324.63***	2117322.53	2114944.67	2112827.83	2114852.27	2115584.83	2115388.40

Tabela 5.5: Número de instâncias em que cada variante alcançou os melhores valores, onde #Melhor representa a contagem para o melhor custo, #Média é a contagem para os custos médios, e #TesteEstat. é o número de instâncias em que cada variante encontrou os melhores valores médios ou suas soluções são estatisticamente semelhantes aos melhores.

Score	Agente										
	A_{S9}	A_{S10}	A_{S11}	A_{Am15}	$A_{N25.01}$	$A_{Sko42.01}$	$A_{Sko42.02}$	$A_{Sko42.03}$	$A_{Sko42.04}$	$A_{Sko49.01}$	A_{Geral}
#Melhor	5	22	20	11	20	21	21	28	18	26	**30**
#Média	0	5	8	3	8	19	10	6	6	8	**30**
#TesteEstat.	1	34	54	9	43	63	57	37	62	64	**82**

5.3.2 Resultados Comparativos

Os resultados obtidos pela variante A_{Geral} foram comparados com BVND (HERRÁN; COLMENAR; DUARTE, 2021), melhor resultado conhecido na literatura. As Tabelas 5.6 e 5.7 apresentam as comparações.

Para ambas as tabelas, a primeira coluna mostra o nome das instâncias. A segunda, apresenta o melhor resultado obtido em (HERRÁN; COLMENAR; DUARTE, 2021). A terceira coluna apresentam o resultado da variante A_{Geral}. Por fim, a última coluna apresenta o hiato obtido entre os dois trabalhos.

Pode-se observar diferenças menores que 1.0% para todas as instâncias testadas aqui, exceto para ste36.02, ste36.03 e ste36.05. A diferença média para todas as instâncias

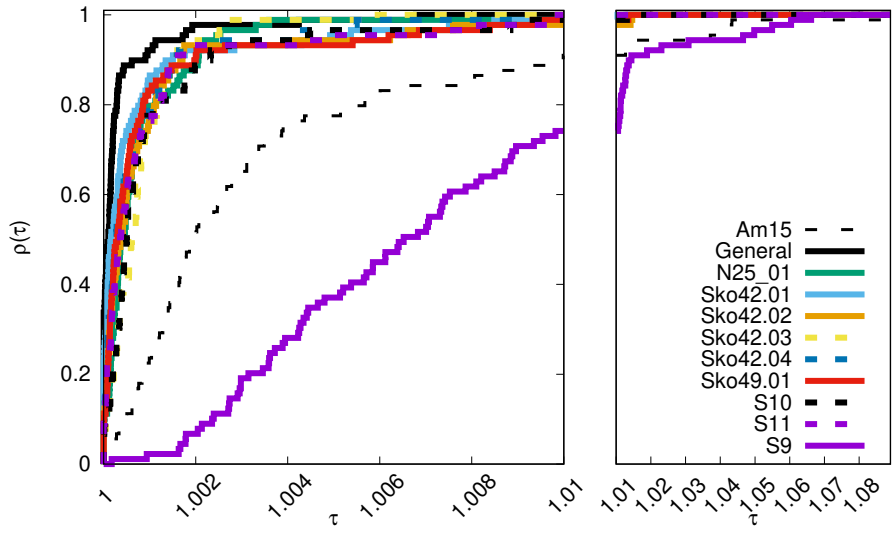


Figura 5.1: PD utilizando os resultados médios como medida de desempenho. As áreas normalizadas sob as curvas dos PD são 1.0000, 0.9951, 0.9949, 0.9949, 0.9946, 0.9930, 0.9927, 0.9919, 0.9915, 0.9405 e 0.8931, respectivamente, para A_{Geral}, $A_{Sko42.01}$, $A_{Sko42.03}$, $A_{Sko42.04}$, $A_{N25.01}$, A_{S11}, $A_{Sko49.01}$, $A_{Sko42.02}$, A_{S10}, A_{Am15} e A_{S9}.

é de 0.09%, e, portanto, os resultados encontrados pela proposta são similares aos melhores encontrados na literatura.

5.4 Avaliação do Melhor Agente

Uma das principais vantagens das técnicas de PG é suas soluções simbólicas. A Figura 5.2 apresenta a melhor heurística gerada pela HPGG e que foi obtida usando a variante A_{Geral}. Essa solução contém um padrão nos quatro blocos de instruções agrupados nas linhas 4-7. Cada uma dessas linhas começa com uma perturbação e é seguida por uma ou mais buscas locais. Essas etapas podem ser observadas em técnicas de busca bem conhecidas, como o ILS (TALBI, 2009). Por outro lado, há uma predominância do uso de reconstruções da solução (linhas 4-6). Essas instruções são comuns ao IG, que realiza operações de desconstrução e reconstrução em alguns elementos da solução até que um critério de parada seja alcançado (LOZANO; MOLINA; GARCíA-MARTíNEZ, 2011). As heurísticas de cada variante gerada são apresentadas em detalhes no Apêndice B.

```
1
2  void execute(I){
3      S = randomlyGenerate(I); Sb=S;
4      for(i={1,...,50}){
5          S.reconstruction(0.40);
6          S.rvnd();
7          S.2-opt();
8          S.insertion();
9          S.reconstrucion(0.65);
10         S.rvnd();
11         S.insertion();
12         S.reconstruction(0.50);
13         S.rvnd();
14         S.swap(0.40,0.25);
15         S.rvnd();
16         if (F(S)<F(Sb)){Sb=S;}
17     }
18 }
```

Figura 5.2: Melhor heurística gerada pela variante Geral.

Tabela 5.6: Resultados comparativos para instâncias de tamanho $n \leq 56$.

Instância	Melhor Custo		GAP %
	BVNS	Agente	
S9	1181.5	**1181.5**	0.00
S9H	2294.5	**2294.5**	0.00
S10	1374.5	**1374.5**	0.00
S11	3439.5	**3439.5**	0.00
Am12a	1529.0	**1529.0**	0.00
Am12b	1609.5	**1609.5**	0.00
Am13a	2467.5	**2467.5**	0.00
Am13b	2870.0	**2870.0**	0.00
Am15	3195.0	**3195.0**	0.00
N25.01	2302.0	**2302.0**	0.00
N25.02	18595.5	**18595.5**	0.00
N25.03	12114.0	12116.0	0.02
N25.04	24192.5	24214.5	0.09
N25.05	7819.0	**7819.0**	0.00
N30.01	4115.0	**4115.0**	0.00
N30.02	10779.5	10781.5	0.02
N30.03	22702.0	**22702.0**	0.00
N30.04	28401.5	28411.5	0.04
N30.05	57400.0	57437.0	0.06
P1	30282.5	30312.5	0.10
P2	33974.0	33985.0	0.03
P3	35052.5	35068.5	0.05
P4	34666.5	34675.5	0.03
P5	30771.0	30787.0	0.05
P6	34424.5	34438.5	0.04
ste36.01	4966.0	**4966.0**	0.00
ste36.02	87390.0	88466.0	1.23
ste36.03	50084.5	50713.5	1.26
ste36.04	46737.5	46992.5	0.55
ste36.05	44488.5	44974.5	1.09
N40.01	53722.5	53740.5	0.03
N40.02	48908.0	48925.0	0.03
N40.03	39250.5	39263.5	0.03
N40.04	38354.0	38383.0	0.08
N40.05	51496.0	51544.0	0.09
Sko42.01	12731.0	**12731.0**	0.00
Sko42.02	108006.5	108071.5	0.06
Sko42.03	86644.5	86684.5	0.05
Sko42.04	68701.0	68771.0	0.10
Sko42.05	124017.5	124075.5	0.05
Sko49.01	20470.0	**20470.0**	0.00
Sko49.02	208058.0	208160.0	0.05
Sko49.03	162182.0	162344.0	0.10
Sko49.04	118246.5	118307.5	0.05
Sko49.05	332836.0	333033.0	0.06
Sko56.01	31972.0	31983.0	0.03
Sko56.02	248219.0	248279.0	0.02
Sko56.03	85184.0	85218.0	0.04
Sko56.04	156646.0	156758.0	0.07
Sko56.05	296176.5	296270.5	0.03

Tabela 5.7: Resultados comparativos para instâncias de tamanho $n \geq 60$.

Instância	**Melhor Custo**		**GAP %**
	BVNS	**Agente**	
AKV.60.05	159643.0	159763.0	0.08
CAP.30.30.1	204089.0	204287.0	0.10
CAP.30.30.2	193199.5	193407.5	0.11
CAP.30.30.3	161475.5	161793.5	0.20
CAP.30.40.1	135133.5	135294.5	0.12
CAP.30.40.2	159114.0	159324.0	0.13
CAP.30.40.3	158967.5	159103.5	0.09
CAP.30.50.1	110493.5	110736.5	0.22
CAP.30.50.2	115302.0	115513.0	0.18
CAP.30.50.3	114137.0	114337.0	0.18
CAP.30.60.1	108117.0	108184.0	0.06
CAP.30.60.2	109908.5	110049.5	0.13
CAP.30.60.3	91619.0	91882.0	0.29
CAP.60.30.1	445377.5	445473.5	0.02
CAP.60.30.2	407890.5	408051.5	0.04
CAP.60.30.3	416912.5	417003.5	0.02
CAP.60.40.1	313304.0	313371.0	0.02
CAP.60.40.2	320758.5	320807.5	0.02
CAP.60.40.3	363020.5	363137.5	0.03
CAP.60.50.1	273420.0	273527.0	0.04
CAP.60.50.2	269680.5	269917.5	0.09
CAP.60.50.3	295419.0	295552.0	0.05
CAP.60.60.1	227896.0	227945.0	0.02
CAP.60.60.2	246523.0	246628.0	0.04
CAP.60.60.3	206486.5	206635.5	0.07
CAP.90.30.1	628850.0	628934.0	0.01
CAP.90.30.2	561170.5	561212.5	0.01
CAP.90.30.3	587816.5	587852.5	0.01
CAP.90.40.1	474046.0	474088.0	0.01
CAP.90.40.2	479963.0	480022.0	0.01
CAP.90.40.3	512452.0	512509.0	0.01
CAP.90.50.1	479683.0	479768.0	0.02
CAP.90.50.2	445030.0	445105.0	0.02
CAP.90.50.3	495052.5	495054.5	0.00
CAP.90.60.1	385430.5	385456.5	0.01
CAP.90.60.2	344775.0	344784.0	0.00
CAP.90.60.3	411205.0	411284.0	0.02
Sko64.05	250870.5	250924.5	0.02
AKV.70.05	2109745.5	2110528.5	0.04

6 Conclusões

Este trabalho abordou o *Corridor Allocation Problem* (CAP), problema de *layout* cujo objetivo é a alocação física de facilidades em um corredor de dois lados, a fim de reduzir o custo de comunicação. Foi desenvolvida uma Hiper-heurística com Programação Genética Gramatical (HPGG) com o objetivo de desenvolver heurísticas capazes de atingir melhores resultados para o problema.

A HPGG proposta foi capaz de produzir heurísticas eficientes para o CAP. Duas estratégias foram propostas: utilizando uma única instância durante a avaliação do GP e treinando cada indivíduo em um conjunto de instâncias.

Foi observado que a variante A_{Geral} da HPGG, que evoluiu utilizando informações de dez instâncias, obteve a melhor média de resultados, tendo o melhor desempenho geral, sendo a segunda abordagem mais confiável. Com base nessas informações, é possível concluir que a exploração do problema sob a perspectiva de suas diferentes características permite o desenvolvimento de heurísticas mais robustas.

Por outro lado, uma das variantes com uma única instância também obteve bons resultados. Especificamente, observamos que o uso das instâncias Sko42.01 e Sko42.03 durante a evolução permite a criação de boas heurísticas. Uma única instância de treino, se escolhida adequadamente, se destaca pelo fato de demandar apenas uma fração de tempo do método que utiliza múltiplas instâncias.

Para trabalhos futuros, considera-se relevante a aplicação do tempo de processamento para compor a aptidão do indivíduo em uma versão multiobjetivo da hiper-heurística proposta aqui. Também é pertinente o uso de novas heurísticas para compor a gramática, diversificando e enriquecendo a geração de indivíduos com uma variedade de operações mais complexas, a fim de abranger uma maior variedade de vizinhanças. Na etapa de seleção, é válido considerar a aplicação de técnicas aprimoradas que possibilitem a seleção de progenitores mais qualificados, em contraste com métodos totalmente aleatórios. Por fim, vale ressaltar que, embora os resultados obtidos não tenham sido capazes de superar aqueles presentes na literatura, a heurística sugerida pela variante A_{Geral} se

assemelha a um *Iterated Greedy*, sendo potencial objeto de estudo.

A Descrição das instâncias

As tabelas a seguir apresentam as instâncias de treino da seguinte forma: A parte superior contém o comprimento das facilidades, cuja coluna i se refere a l_i. A parte inferior pode ser entendida como uma matriz $F_{n,n}$ representando os coeficientes de demanda $f_{i,j}$.

Tabela A.1: Características da Instância S9

Comprimentos (l_i)

2	8	9	7	3	4	6	8	9

Coeficiente de Demanda ($f_{i,j}$)

0	0	2	8	7	4	0	1	6
0	0	8	0	2	7	4	4	6
2	8	0	2	7	8	0	2	6
8	0	2	0	5	0	8	8	6
7	2	7	5	0	5	4	7	6
4	7	8	0	5	0	8	2	6
0	4	0	8	4	8	0	4	6
1	4	2	8	7	2	4	0	6
6	6	6	6	6	6	6	6	0

Tabela A.2: Características da Instância S10

Comprimentos (l_i)

6	3	9	4	2	6	8	9	6	7

Coeficiente de Demanda ($f_{i,j}$)

0	0	9	5	1	4	5	4	7	0
0	0	4	5	2	7	2	9	0	7
9	4	0	0	7	2	5	0	9	4
5	5	0	0	7	9	0	5	2	5
1	2	7	7	0	0	4	7	2	12
4	7	2	9	0	0	4	9	3	5
5	2	5	0	4	4	0	0	2	9
4	9	0	5	7	9	0	0	1	7
7	0	9	2	2	3	2	1	0	0
0	7	4	5	12	5	9	7	0	0

Tabela A.3: Características da Instância S11

Comprimentos (l_i)

3	9	3	7	3	7	5	9	6	5	10

Coeficiente de Demanda ($f_{i,j}$)

0	20	2	8	0	9	5	7	0	20	3
20	0	8	9	13	17	16	1	8	6	7
2	8	0	18	0	10	4	18	5	8	0
8	9	18	0	6	16	10	4	2	14	6
0	13	0	6	0	6	0	11	0	8	2
9	17	10	16	6	0	6	13	2	7	18
5	16	4	10	0	6	0	1	11	15	7
7	1	18	4	11	13	1	0	1	7	2
0	8	5	2	0	2	11	1	0	12	0
20	6	8	14	8	7	15	7	12	0	3
3	7	0	6	2	18	7	2	0	3	0

Tabela A.4: Características da Instância Am15

Comprimentos (l_i)

20	3	9	3	7	3	7	5	9	6	5	3	9	3	7

Coeficiente de Demanda ($f_{i,j}$)

0	10	0	5	1	0	1	2	2	2	2	0	4	0	0
10	0	1	3	2	2	2	3	2	0	2	0	10	5	0
0	1	0	10	2	0	2	5	4	5	2	2	5	5	5
5	3	10	0	1	1	5	0	0	2	1	0	2	5	0
1	2	2	1	0	3	5	5	5	1	0	3	0	5	5
0	2	0	1	3	0	2	2	1	5	0	0	2	5	10
1	2	2	5	5	2	0	6	0	1	5	5	5	1	0
2	3	5	0	5	2	6	0	5	2	10	0	5	0	0
2	2	4	0	5	1	0	5	0	0	10	5	10	0	2
2	0	5	2	1	5	1	2	0	0	0	4	0	0	5
2	2	2	1	0	0	5	10	10	0	0	5	0	5	0
0	0	2	0	3	0	5	0	5	4	5	0	3	3	0
4	10	5	2	0	2	5	5	10	0	0	3	0	10	2
0	5	5	5	5	5	1	0	0	0	5	3	10	0	4
0	0	5	0	5	10	0	0	2	5	0	0	2	4	0

Tabela A.5: Características da Instância N25.1

Comprimentos (l_i)

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Coeficiente de Demanda ($f_{i,j}$)

0	3	2	0	0	10	5	0	5	2	0	0	2	0	5	3	0	1	10	0	2	1	1	1	0
3	0	4	0	10	0	0	2	2	1	5	0	0	0	0	0	1	6	1	0	2	2	5	1	10
2	4	0	3	4	5	5	5	1	4	0	4	0	4	0	3	2	5	5	2	0	0	3	1	0
0	0	3	0	0	0	2	2	0	6	2	5	2	5	1	1	1	2	2	4	2	0	2	2	5
0	10	4	0	0	2	0	0	0	0	0	0	0	0	2	0	0	2	0	5	0	2	1	0	2
10	0	5	0	2	0	10	10	5	10	6	0	0	10	2	10	1	5	5	2	5	0	2	0	1
5	0	5	2	0	10	0	1	3	5	0	0	2	4	5	10	6	0	5	5	5	0	5	5	0
0	2	5	2	0	10	1	0	10	2	5	2	0	3	0	0	0	4	0	5	0	5	2	2	5
5	2	1	0	0	5	3	10	0	5	6	0	1	5	5	5	2	3	5	0	2	10	10	1	5
2	1	4	6	0	10	5	2	5	0	0	1	2	1	0	0	0	0	6	6	4	5	3	2	2
0	5	0	2	0	6	0	5	6	0	0	2	0	4	2	1	0	6	2	1	5	0	0	1	5
0	0	4	5	0	0	0	2	0	1	2	0	2	1	0	3	10	0	0	4	0	0	4	2	5
2	0	0	2	0	0	2	0	1	2	0	2	0	4	5	0	1	0	5	0	0	0	5	1	1
0	0	4	5	0	10	4	3	5	1	4	1	4	0	0	0	2	2	0	2	5	0	5	2	5
5	0	0	1	2	2	5	0	5	0	2	0	5	0	0	2	0	0	0	6	3	5	0	0	5
3	0	3	1	0	10	10	0	5	0	1	3	0	0	2	0	0	5	5	1	5	2	1	2	10
0	1	2	1	0	1	6	0	2	0	0	10	1	2	0	0	0	5	2	1	1	5	6	5	5
1	6	5	2	2	5	0	4	3	0	6	0	0	2	0	5	5	0	4	0	0	0	0	5	0
10	1	5	2	0	5	5	0	5	6	2	0	5	0	0	5	2	4	0	5	4	4	5	0	2
0	0	2	4	5	2	5	5	0	6	1	4	0	2	6	1	1	0	5	0	4	4	1	0	2
2	2	0	2	0	5	5	0	2	4	5	0	0	5	3	5	1	0	4	4	0	1	0	10	1
1	2	0	0	2	0	0	5	10	5	0	0	0	0	5	2	5	0	4	4	1	0	0	0	0
1	5	3	2	1	2	5	2	10	3	0	4	5	5	0	1	6	0	5	1	0	0	0	0	0
1	1	1	2	0	0	5	2	1	2	1	2	1	2	0	2	5	5	0	0	10	0	0	0	2
0	10	0	5	2	1	0	5	5	2	5	5	1	5	5	10	5	0	2	2	1	0	0	2	0

Tabela A.6: Características da Instância Sko.42.1.

Comprimentos (l_i)

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Coeficiente de Demanda (l_i)

0	2	10	5	4	1	5	6	5	0	2	0	0	1	2	5	0	5	0	2	5	5	0	10	0	1	4	2	0	0	0	2	0	5	5	0	10	10	0	0	0	2
2	0	1	0	5	1	0	0	0	2	1	0	5	10	0	1	0	10	10	0	2	1	1	0	0	3	5	10	0	5	0	2	1	1	3	10	5	0	10	0	1	4
10	1	0	0	1	5	10	0	0	1	0	5	0	3	2	5	1	5	0	1	1	1	5	1	2	2	0	1	1	5	2	6	1	1	0	10	5	10	5	5	2	1
5	0	0	0	0	1	0	0	10	2	1	2	5	5	5	5	0	2	2	6	5	5	0	5	2	1	0	0	1	2	3	2	10	5	4	0	0	2	10	3	0	5
4	5	1	0	0	1	6	1	1	0	0	6	0	0	2	2	2	3	5	0	2	2	2	0	5	5	5	0	1	5	0	4	2	2	1	5	6	2	2	0	5	5
1	1	5	1	1	0	0	5	1	5	1	0	0	0	1	0	0	0	1	0	0	1	2	5	5	1	2	1	10	0	0	0	2	5	0	10	5	4	5	5	2	5
5	0	10	0	6	0	0	0	5	4	10	10	2	0	2	5	10	0	4	5	6	4	0	2	1	2	1	2	2	0	2	3	2	5	0	0	0	0	1	1	6	1
6	0	0	0	1	5	0	0	0	1	5	2	0	0	2	2	0	2	0	0	4	0	0	2	5	3	2	2	5	2	2	2	5	0	2	5	2	2	0	5	1	3
5	0	0	10	1	1	5	0	0	5	0	5	4	2	10	5	0	2	2	1	2	0	0	6	1	5	0	0	0	0	5	10	10	3	5	1	2	0	0	10	1	6
0	2	1	2	0	5	4	1	5	0	1	2	5	3	2	5	5	5	10	0	0	1	2	4	0	0	5	0	3	2	5	0	0	1	5	2	0	1	5	2	0	10
2	1	0	1	0	1	10	5	0	1	0	5	2	5	0	2	1	2	0	2	0	5	2	5	1	4	0	1	10	3	5	5	0	5	2	0	0	0	5	0	0	5
0	0	5	2	6	0	10	2	5	2	5	0	1	0	0	0	10	2	4	5	0	0	0	0	1	10	5	1	1	5	5	2	1	5	10	5	3	5	5	1	5	5
0	5	0	5	0	0	2	0	4	5	2	1	0	0	0	0	3	0	5	6	0	5	2	1	1	5	2	1	0	0	10	5	5	2	2	5	1	1	0	1	0	0
1	10	3	5	0	0	0	0	2	3	5	0	0	0	10	2	5	1	5	5	2	2	0	0	1	2	0	5	2	4	5	10	4	0	10	2	10	1	2	5	4	1
2	0	2	5	2	1	2	2	10	2	0	0	0	10	0	0	6	0	1	2	0	2	0	0	0	5	5	2	5	0	5	1	5	1	5	0	5	5	2	10	5	5
5	1	5	5	2	0	5	2	5	5	2	0	0	2	0	0	2	0	5	5	2	0	4	0	0	0	0	0	2	2	0	0	5	2	0	5	1	0	5	0	2	0
0	0	1	0	2	0	10	0	0	5	1	10	3	5	6	2	0	0	0	5	1	1	0	3	5	0	2	1	5	1	5	6	0	3	5	0	5	0	0	5	5	10
5	10	5	2	3	0	0	2	2	5	2	2	0	1	0	0	0	0	5	5	0	0	2	2	6	10	0	2	0	2	2	10	2	0	5	1	0	6	3	4	5	5
0	10	0	2	5	1	4	0	2	10	0	4	5	5	1	5	0	5	0	1	0	0	0	3	2	10	4	5	5	2	5	1	1	2	5	0	0	10	3	2	1	10
2	0	1	6	0	0	5	0	1	0	2	5	6	5	2	5	5	5	1	0	3	1	2	0	2	10	10	5	2	1	2	1	0	10	2	5	0	5	0	4	0	2
5	2	1	5	2	0	6	4	2	0	0	0	0	2	0	2	1	0	0	3	0	0	2	0	4	4	1	1	0	5	1	5	10	1	0	0	5	10	10	4	5	2
5	1	1	5	2	1	4	0	0	1	5	0	5	2	2	0	1	0	0	1	0	0	5	5	2	5	2	10	0	2	0	2	4	0	2	5	5	2	2	2	5	0
0	1	5	0	2	2	0	0	0	2	2	0	2	0	0	4	0	2	0	2	2	5	0	2	0	2	0	2	0	5	2	0	2	1	5	10	6	1	5	2	0	0
10	0	1	5	0	5	2	2	6	4	5	0	1	0	0	0	3	2	3	0	0	5	2	0	2	10	0	2	0	0	5	3	5	2	0	10	10	3	0	2	1	0
0	0	2	2	5	5	1	5	1	0	1	1	1	1	0	0	5	6	2	2	4	2	0	2	0	1	4	0	5	4	1	0	2	2	1	1	0	4	0	3	5	5
1	3	2	1	5	1	2	3	5	0	4	10	5	2	5	0	0	10	10	10	4	5	2	10	1	0	2	10	5	0	1	5	0	0	10	2	5	0	0	0	0	1
4	5	0	0	5	2	1	2	0	5	0	5	2	0	5	0	2	0	4	10	1	2	0	0	4	2	0	2	0	2	3	0	5	0	3	2	0	2	10	0	1	2
2	10	1	0	0	1	2	2	0	0	1	1	1	5	2	0	1	2	5	5	1	10	2	2	0	10	2	0	3	2	5	0	10	2	0	5	5	2	1	0	0	0
0	0	1	1	1	10	2	5	0	3	10	1	0	2	5	2	5	0	5	2	0	0	0	0	5	5	0	3	0	0	0	0	3	6	5	0	5	2	1	5	0	2
0	5	5	2	5	0	0	2	0	2	3	5	0	4	0	2	1	2	2	1	5	2	5	0	4	0	2	2	0	0	1	0	5	4	10	5	1	0	10	5	0	3
0	0	2	3	0	0	2	2	5	5	5	5	10	5	5	0	5	2	5	2	1	0	2	5	1	1	3	5	0	1	0	2	0	5	2	0	2	0	10	3	0	0
2	2	6	2	4	0	3	2	10	0	5	2	5	10	1	0	6	10	1	1	5	2	0	3	0	5	0	0	0	0	2	0	2	4	1	2	0	0	2	2	0	6
0	1	1	10	2	2	2	5	10	0	0	1	5	4	5	5	0	2	1	0	10	4	2	5	2	0	5	10	3	5	0	2	0	2	5	0	0	0	2	2	0	0
5	1	1	5	2	5	5	0	3	1	5	5	2	0	1	2	3	0	2	10	1	0	1	2	2	0	0	2	6	4	5	4	2	0	2	1	0	10	0	0	1	10
5	3	0	4	1	0	0	2	5	5	2	10	2	10	5	0	5	5	5	2	0	2	5	0	1	10	3	0	5	10	2	1	5	2	0	5	2	0	2	1	0	0
0	10	10	0	5	10	0	5	1	2	0	5	5	2	0	5	0	1	0	5	0	5	10	10	1	2	2	5	0	5	0	2	0	1	5	0	0	2	5	0	3	5
10	5	5	0	6	5	0	2	2	0	0	3	1	10	5	1	5	0	0	0	5	5	6	10	0	5	0	5	5	1	2	0	0	0	2	0	0	10	10	5	0	1
10	0	10	2	2	4	0	2	0	1	0	5	1	1	5	0	0	6	10	5	10	2	1	3	4	0	2	2	2	0	0	0	0	10	0	2	10	0	2	10	0	0
0	10	5	10	2	5	1	0	0	5	5	5	0	2	2	5	0	3	3	0	10	2	5	0	0	0	10	1	1	10	10	2	2	0	2	5	10	2	0	1	0	4
0	0	5	3	0	5	1	5	10	2	0	1	1	5	10	0	5	4	2	4	4	2	2	2	3	0	0	0	5	5	3	2	2	0	1	0	5	10	1	0	2	1
0	1	2	0	5	2	6	1	1	0	0	5	0	4	5	2	5	5	1	0	5	5	0	1	5	0	1	0	0	0	0	0	0	1	0	3	0	0	0	2	0	5
2	4	1	5	5	5	1	3	6	10	5	5	0	1	5	0	10	5	10	2	2	0	0	0	5	1	2	0	2	3	0	6	0	10	0	5	1	0	4	1	5	0

Tabela A.7: Características da Instância Sko.42.2.

Comprimentos (l_i)

21	17	10	11	5	14	7	20	16	9	16	6	10	20	15	5	18	14	4	5	13	14	8	13	10	2	2	7	1	9	15	3	2	13	13	1	1	5	13	2	8	14

Coeficiente de Demanda (l_i)

0	2	10	5	4	1	5	6	5	0	2	0	0	1	2	5	0	5	0	2	5	5	0	10	0	1	4	2	0	0	0	2	0	5	5	0	10	10	0	0	0	2
2	0	1	0	5	1	0	0	0	2	1	0	5	10	0	1	0	10	10	0	2	1	1	0	0	3	5	10	0	5	0	2	1	1	3	10	5	0	10	0	1	4
10	1	0	0	1	5	10	0	0	1	0	5	0	3	2	5	1	5	0	1	1	1	5	1	2	2	0	1	1	5	2	6	1	1	0	10	5	10	5	5	2	1
5	0	0	0	0	1	0	0	10	2	1	2	5	5	5	5	0	2	2	6	5	5	0	5	2	1	0	0	1	2	3	2	10	5	4	0	0	2	10	3	0	5
4	5	1	0	0	1	6	1	1	0	0	6	0	0	2	2	2	3	5	0	2	2	2	0	5	5	5	0	1	5	0	4	2	2	1	5	6	2	2	0	5	5
1	1	5	1	1	0	0	5	1	5	1	0	0	0	1	0	0	0	1	0	0	1	2	5	5	1	2	1	10	0	0	0	2	5	0	10	5	4	5	5	2	5
5	0	10	0	6	0	0	0	5	4	10	10	2	0	2	5	10	0	4	5	6	4	0	2	1	2	1	2	2	0	2	3	2	5	0	0	0	0	1	1	6	1
6	0	0	0	1	5	0	0	0	1	5	2	0	0	2	2	0	2	0	0	4	0	0	2	5	3	2	2	5	2	2	2	5	0	2	5	2	2	0	5	1	3
5	0	0	10	1	1	5	0	0	5	0	5	4	2	10	5	0	2	2	1	2	0	0	6	1	5	0	0	0	0	5	10	10	3	5	1	2	0	0	10	1	6
0	2	1	2	0	5	4	1	5	0	1	2	5	3	2	5	5	5	10	0	0	1	2	4	0	0	5	0	3	2	5	0	0	1	5	2	0	1	5	2	0	10
2	1	0	1	0	1	10	5	0	1	0	5	2	5	0	2	1	2	0	2	0	5	2	5	1	4	0	1	10	3	5	5	0	5	2	0	0	0	5	0	0	5
0	0	5	2	6	0	10	2	5	2	5	0	1	0	0	0	10	2	4	5	0	0	0	0	1	10	5	1	1	5	5	2	1	5	10	5	3	5	5	1	5	5
0	5	0	5	0	0	2	0	4	5	2	1	0	0	0	0	3	0	5	6	0	5	2	1	1	5	2	1	0	0	10	5	5	2	2	5	1	1	0	1	0	0
1	10	3	5	0	0	0	0	2	3	5	0	0	0	10	2	5	1	5	5	2	2	0	0	1	2	0	5	2	4	5	10	4	0	10	2	10	1	2	5	4	1
2	0	2	5	2	1	2	2	10	2	0	0	0	10	0	0	6	0	1	2	0	2	0	0	0	5	5	2	5	0	5	1	5	1	5	0	5	5	2	10	5	5
5	1	5	5	2	0	5	2	5	5	2	0	0	2	0	0	2	0	5	5	2	0	4	0	0	0	0	0	2	2	0	0	5	2	0	5	1	0	5	0	2	0
0	0	1	0	2	0	10	0	0	5	1	10	3	5	6	2	0	0	0	5	1	1	0	3	5	0	2	1	5	1	5	6	0	3	5	0	5	0	0	5	5	10
5	10	5	2	3	0	0	2	2	5	2	2	0	1	0	0	0	0	5	5	0	0	2	2	6	10	0	2	0	2	2	10	2	0	5	1	0	6	3	4	5	5
0	10	0	2	5	1	4	0	2	10	0	4	5	5	1	5	0	5	0	1	0	0	0	3	2	10	4	5	5	2	5	1	1	2	5	0	0	10	3	2	1	10
2	0	1	6	0	0	5	0	1	0	2	5	6	5	2	5	5	5	1	0	3	1	2	0	2	10	10	5	2	1	2	1	0	10	2	5	0	5	0	4	0	2
5	2	1	5	2	0	6	4	2	0	0	0	0	2	0	2	1	0	0	3	0	0	2	0	4	4	1	1	0	5	1	5	10	1	0	0	5	10	10	4	5	2
5	1	1	5	2	1	4	0	0	1	5	0	5	2	2	0	1	0	0	1	0	0	5	5	2	5	2	10	0	2	0	2	4	0	2	5	5	2	2	2	5	0
0	1	5	0	2	2	0	0	0	2	2	0	2	0	0	4	0	2	0	2	2	5	0	2	0	2	0	2	0	5	2	0	2	1	5	10	6	1	5	2	0	0
10	0	1	5	0	5	2	2	6	4	5	0	1	0	0	0	3	2	3	0	0	5	2	0	2	10	0	2	0	0	5	3	5	2	0	10	10	3	0	2	1	0
0	0	2	2	5	5	1	5	1	0	1	1	1	1	0	0	5	6	2	2	4	2	0	2	0	1	4	0	5	4	1	0	2	2	1	1	0	4	0	3	5	5
1	3	2	1	5	1	2	3	5	0	4	10	5	2	5	0	0	10	10	10	4	5	2	10	1	0	2	10	5	0	1	5	0	0	10	2	5	0	0	0	0	1
4	5	0	0	5	2	1	2	0	5	0	5	2	0	5	0	2	0	4	10	1	2	0	0	4	2	0	2	0	2	3	0	5	0	3	2	0	2	10	0	1	2
2	10	1	0	0	1	2	2	0	0	1	1	1	5	2	0	1	2	5	5	1	10	2	2	0	10	2	0	3	2	5	0	10	2	0	5	5	2	1	0	0	0
0	0	1	1	1	10	2	5	0	3	10	1	0	2	5	2	5	0	5	2	0	0	0	0	5	5	0	3	0	0	0	0	3	6	5	0	5	2	1	5	0	2
0	5	5	2	5	0	0	2	0	2	3	5	0	4	0	2	1	2	2	1	5	2	5	0	4	0	2	2	0	0	1	0	5	4	10	5	1	0	10	5	0	3
0	0	2	3	0	0	2	2	5	5	5	5	10	5	5	0	5	2	5	2	1	0	2	5	1	1	3	5	0	1	0	2	0	5	2	0	2	0	10	3	0	0
2	2	6	2	4	0	3	2	10	0	5	2	5	10	1	0	6	10	1	1	5	2	0	3	0	5	0	0	0	0	2	0	2	4	1	2	0	0	2	2	0	6
0	1	1	10	2	2	2	5	10	0	0	1	5	4	5	5	0	2	1	0	10	4	2	5	2	0	5	10	3	5	0	2	0	2	5	0	0	0	2	2	0	0
5	1	1	5	2	5	5	0	3	1	5	5	2	0	1	2	3	0	2	10	1	0	1	2	2	0	0	2	6	4	5	4	2	0	2	1	0	10	0	0	1	10
5	3	0	4	1	0	0	2	5	5	2	10	2	10	5	0	5	5	5	2	0	2	5	0	1	10	3	0	5	10	2	1	5	2	0	5	2	0	2	1	0	0
0	10	10	0	5	10	0	5	1	2	0	5	5	2	0	5	0	1	0	5	0	5	10	10	1	2	2	5	0	5	0	2	0	1	5	0	0	2	5	0	3	5
10	5	5	0	6	5	0	2	2	0	0	3	1	10	5	1	5	0	0	0	5	5	6	10	0	5	0	5	5	1	2	0	0	0	2	0	0	10	10	5	0	1
10	0	10	2	2	4	0	2	0	1	0	5	1	1	5	0	0	6	10	5	10	2	1	3	4	0	2	2	2	0	0	0	0	10	0	2	10	0	2	10	0	0
0	10	5	10	2	5	1	0	0	5	5	5	0	2	2	5	0	3	3	0	10	2	5	0	0	0	10	1	1	10	10	2	2	0	2	5	10	2	0	1	0	4
0	0	5	3	0	5	1	5	10	2	0	1	1	5	10	0	5	4	2	4	4	2	2	2	3	0	0	0	5	5	3	2	2	0	1	0	5	10	1	0	2	1
0	1	2	0	5	2	6	1	1	0	0	5	0	4	5	2	5	5	1	0	5	5	0	1	5	0	1	0	0	0	0	0	0	1	0	3	0	0	0	2	0	5
2	4	1	5	5	5	1	3	6	10	5	5	0	1	5	0	10	5	10	2	2	0	0	0	5	1	2	0	2	3	0	6	0	10	0	5	1	0	4	1	5	0

Tabela A.8: Características da Instância Sko.42.3.

Comprimentos (l_i)

12	11	2	8	8	6	3	11	11	12	8	9	3	8	12	14	5	5	14	4	13	15	4	5	2	2	11	4	14	4	4	16	8	6	6	6	7	10	3	2	8	14

Coeficiente de Demanda (l_i)

0	2	10	5	4	1	5	6	5	0	2	0	0	1	2	5	0	5	0	2	5	5	0	10	0	1	4	2	0	0	0	2	0	5	5	0	10	10	0	0	0	2
2	0	1	0	5	1	0	0	0	2	1	0	5	10	0	1	0	10	10	0	2	1	1	0	0	3	5	10	0	5	0	2	1	1	3	10	5	0	10	0	1	4
10	1	0	0	1	5	10	0	0	1	0	5	0	3	2	5	1	5	0	1	1	1	5	1	2	2	0	1	1	5	2	6	1	1	0	10	5	10	5	5	2	1
5	0	0	0	0	1	0	0	10	2	1	2	5	5	5	5	0	2	2	6	5	5	0	5	2	1	0	0	1	2	3	2	10	5	4	0	0	2	10	3	0	5
4	5	1	0	0	1	6	1	1	0	0	6	0	0	2	2	2	3	5	0	2	2	2	0	5	5	5	0	1	5	0	4	2	2	1	5	6	2	2	0	5	5
1	1	5	1	1	0	0	5	1	5	1	0	0	0	1	0	0	0	1	0	0	1	2	5	5	1	2	1	10	0	0	0	2	5	0	10	5	4	5	5	2	5
5	0	10	0	6	0	0	0	5	4	10	10	2	0	2	5	10	0	4	5	6	4	0	2	1	2	1	2	2	0	2	3	2	5	0	0	0	0	1	1	6	1
6	0	0	0	1	5	0	0	0	1	5	2	0	0	2	2	0	2	0	0	4	0	0	2	5	3	2	2	5	2	2	2	5	0	2	5	2	2	0	5	1	3
5	0	0	10	1	1	5	0	0	5	0	5	4	2	10	5	0	2	2	1	2	0	0	6	1	5	0	0	0	0	5	10	10	3	5	1	2	0	0	10	1	6
0	2	1	2	0	5	4	1	5	0	1	2	5	3	2	5	5	5	10	0	0	1	2	4	0	0	5	0	3	2	5	0	0	1	5	2	0	1	5	2	0	10
2	1	0	1	0	1	10	5	0	1	0	5	2	5	0	2	1	2	0	2	0	5	2	5	1	4	0	1	10	3	5	5	0	5	2	0	0	0	5	0	0	5
0	0	5	2	6	0	10	2	5	2	5	0	1	0	0	0	10	2	4	5	0	0	0	0	1	10	5	1	1	5	5	2	1	5	10	5	3	5	5	1	5	5
0	5	0	5	0	0	2	0	4	5	2	1	0	0	0	0	3	0	5	6	0	5	2	1	1	5	2	1	0	0	10	5	5	2	2	5	1	1	0	1	0	0
1	10	3	5	0	0	0	0	2	3	5	0	0	0	10	2	5	1	5	5	2	2	0	0	1	2	0	5	2	4	5	10	4	0	10	2	10	1	2	5	4	1
2	0	2	5	2	1	2	2	10	2	0	0	0	10	0	0	6	0	1	2	0	2	0	0	0	5	5	2	5	0	5	1	5	1	5	0	5	5	2	10	5	5
5	1	5	5	2	0	5	2	5	5	2	0	0	2	0	0	2	0	5	5	2	0	4	0	0	0	0	0	2	2	0	0	5	2	0	5	1	0	5	0	2	0
0	0	1	0	2	0	10	0	0	5	1	10	3	5	6	2	0	0	0	5	1	1	0	3	5	0	2	1	5	1	5	6	0	3	5	0	5	0	0	5	5	10
5	10	5	2	3	0	0	2	2	5	2	2	0	1	0	0	0	0	5	5	0	0	2	2	6	10	0	2	0	2	2	10	2	0	5	1	0	6	3	4	5	5
0	10	0	2	5	1	4	0	2	10	0	4	5	5	1	5	0	5	0	1	0	0	0	3	2	10	4	5	5	2	5	1	1	2	5	0	0	10	3	2	1	10
2	0	1	6	0	0	5	0	1	0	2	5	6	5	2	5	5	5	1	0	3	1	2	0	2	10	10	5	2	1	2	1	0	10	2	5	0	5	0	4	0	2
5	2	1	5	2	0	6	4	2	0	0	0	0	2	0	2	1	0	0	3	0	0	2	0	4	4	1	1	0	5	1	5	10	1	0	0	5	10	10	4	5	2
5	1	1	5	2	1	4	0	0	1	5	0	5	2	2	0	1	0	0	1	0	0	5	5	2	5	2	10	0	2	0	2	4	0	2	5	5	2	2	2	5	0
0	1	5	0	2	2	0	0	0	2	2	0	2	0	0	4	0	2	0	2	2	5	0	2	0	2	0	2	0	5	2	0	2	1	5	10	6	1	5	2	0	0
10	0	1	5	0	5	2	2	6	4	5	0	1	0	0	0	3	2	3	0	0	5	2	0	2	10	0	2	0	0	5	3	5	2	0	10	10	3	0	2	1	0
0	0	2	2	5	5	1	5	1	0	1	1	1	1	0	0	5	6	2	2	4	2	0	2	0	1	4	0	5	4	1	0	2	2	1	1	0	4	0	3	5	5
1	3	2	1	5	1	2	3	5	0	4	10	5	2	5	0	0	10	10	10	4	5	2	10	1	0	2	10	5	0	1	5	0	0	10	2	5	0	0	0	0	1
4	5	0	0	5	2	1	2	0	5	0	5	2	0	5	0	2	0	4	10	1	2	0	0	4	2	0	2	0	2	3	0	5	0	3	2	0	2	10	0	1	2
2	10	1	0	0	1	2	2	0	0	1	1	1	5	2	0	1	2	5	5	1	10	2	2	0	10	2	0	3	2	5	0	10	2	0	5	5	2	1	0	0	0
0	0	1	1	1	10	2	5	0	3	10	1	0	2	5	2	5	0	5	2	0	0	0	0	5	5	0	3	0	0	0	0	3	6	5	0	5	2	1	5	0	2
0	5	5	2	5	0	0	2	0	2	3	5	0	4	0	2	1	2	2	1	5	2	5	0	4	0	2	2	0	0	1	0	5	4	10	5	1	0	10	5	0	3
0	0	2	3	0	0	2	2	5	5	5	5	10	5	5	0	5	2	5	2	1	0	2	5	1	1	3	5	0	1	0	2	0	5	2	0	2	0	10	3	0	0
2	2	6	2	4	0	3	2	10	0	5	2	5	10	1	0	6	10	1	1	5	2	0	3	0	5	0	0	0	0	2	0	2	4	1	2	0	0	2	2	0	6
0	1	1	10	2	2	2	5	10	0	0	1	5	4	5	5	0	2	1	0	10	4	2	5	2	0	5	10	3	5	0	2	0	2	5	0	0	0	2	2	0	0
5	1	1	5	2	5	5	0	3	1	5	5	2	0	1	2	3	0	2	10	1	0	1	2	2	0	0	2	6	4	5	4	2	0	2	1	0	10	0	0	1	10
5	3	0	4	1	0	0	2	5	5	2	10	2	10	5	0	5	5	5	2	0	2	5	0	1	10	3	0	5	10	2	1	5	2	0	5	2	0	2	1	0	0
0	10	10	0	5	10	0	5	1	2	0	5	5	2	0	5	0	1	0	5	0	5	10	10	1	2	2	5	0	5	0	2	0	1	5	0	0	2	5	0	3	5
10	5	5	0	6	5	0	2	2	0	0	3	1	10	5	1	5	0	0	0	5	5	6	10	0	5	0	5	5	1	2	0	0	0	2	0	0	10	10	5	0	1
10	0	10	2	2	4	0	2	0	1	0	5	1	1	5	0	0	6	10	5	10	2	1	3	4	0	2	2	2	0	0	0	0	10	0	2	10	0	2	10	0	0
0	10	5	10	2	5	1	0	0	5	5	5	0	2	2	5	0	3	3	0	10	2	5	0	0	0	10	1	1	10	10	2	2	0	2	5	10	2	0	1	0	4
0	0	5	3	0	5	1	5	10	2	0	1	1	5	10	0	5	4	2	4	4	2	2	2	3	0	0	0	5	5	3	2	2	0	1	0	5	10	1	0	2	1
0	1	2	0	5	2	6	1	1	0	0	5	0	4	5	2	5	5	1	0	5	5	0	1	5	0	1	0	0	0	0	0	0	1	0	3	0	0	0	2	0	5
2	4	1	5	5	5	1	3	6	10	5	5	0	1	5	0	10	5	10	2	2	0	0	0	5	1	2	0	2	3	0	6	0	10	0	5	1	0	4	1	5	0

Tabela A.9: Características da Instância Sko.42.4.

Comprimentos (l_i)

10	4	3	10	3	7	11	8	10	1	2	9	5	10	8	8	4	3	3	3	5	10	6	9	6	6	6	5	10	1	4	1	8	8	11	7	5	3	7	8	5	1

Coeficiente de Demanda (l_i)

0	2	10	5	4	1	5	6	5	0	2	0	0	1	2	5	0	5	0	2	5	5	0	10	0	1	4	2	0	0	0	2	0	5	5	0	10	10	0	0	0	2
2	0	1	0	5	1	0	0	0	2	1	0	5	10	0	1	0	10	10	0	2	1	1	0	0	3	5	10	0	5	0	2	1	1	3	10	5	0	10	0	1	4
10	1	0	0	1	5	10	0	0	1	0	5	0	3	2	5	1	5	0	1	1	1	5	1	2	2	0	1	1	5	2	6	1	1	0	10	5	10	5	5	2	1
5	0	0	0	0	1	0	0	10	2	1	2	5	5	5	5	0	2	2	6	5	5	0	5	2	1	0	0	1	2	3	2	10	5	4	0	0	2	10	3	0	5
4	5	1	0	0	1	6	1	1	0	0	6	0	0	2	2	2	3	5	0	2	2	2	0	5	5	5	0	1	5	0	4	2	2	1	5	6	2	2	0	5	5
1	1	5	1	1	0	0	5	1	5	1	0	0	0	1	0	0	0	1	0	0	1	2	5	5	1	2	1	10	0	0	0	2	5	0	10	5	4	5	5	2	5
5	0	10	0	6	0	0	0	5	4	10	10	2	0	2	5	10	0	4	5	6	4	0	2	1	2	1	2	2	0	2	3	2	5	0	0	0	0	1	1	6	1
6	0	0	0	1	5	0	0	0	1	5	2	0	0	2	2	0	2	0	0	4	0	0	2	5	3	2	2	5	2	2	2	5	0	2	5	2	2	0	5	1	3
5	0	0	10	1	1	5	0	0	5	0	5	4	2	10	5	0	2	2	1	2	0	0	6	1	5	0	0	0	0	5	10	10	3	5	1	2	0	0	10	1	6
0	2	1	2	0	5	4	1	5	0	1	2	5	3	2	5	5	5	10	0	0	1	2	4	0	0	5	0	3	2	5	0	0	1	5	2	0	1	5	2	0	10
2	1	0	1	0	1	10	5	0	1	0	5	2	5	0	2	1	2	0	2	0	5	2	5	1	4	0	1	10	3	5	5	0	5	2	0	0	0	5	0	0	5
0	0	5	2	6	0	10	2	5	2	5	0	1	0	0	0	10	2	4	5	0	0	0	0	1	10	5	1	1	5	5	2	1	5	10	5	3	5	5	1	5	5
0	5	0	5	0	0	2	0	4	5	2	1	0	0	0	0	3	0	5	6	0	5	2	1	1	5	2	1	0	0	10	5	5	2	2	5	1	1	0	1	0	0
1	10	3	5	0	0	0	0	2	3	5	0	0	0	10	2	5	1	5	5	2	2	0	0	1	2	0	5	2	4	5	10	4	0	10	2	10	1	2	5	4	1
2	0	2	5	2	1	2	2	10	2	0	0	0	10	0	0	6	0	1	2	0	2	0	0	0	5	5	2	5	0	5	1	5	1	5	0	5	5	2	10	5	5
5	1	5	5	2	0	5	2	5	5	2	0	0	2	0	0	2	0	5	5	2	0	4	0	0	0	0	0	2	2	0	0	5	2	0	5	1	0	5	0	2	0
0	0	1	0	2	0	10	0	0	5	1	10	3	5	6	2	0	0	0	5	1	1	0	3	5	0	2	1	5	1	5	6	0	3	5	0	5	0	0	5	5	10
5	10	5	2	3	0	0	2	2	5	2	2	0	1	0	0	0	0	5	5	0	0	2	2	6	10	0	2	0	2	2	10	2	0	5	1	0	6	3	4	5	5
0	10	0	2	5	1	4	0	2	10	0	4	5	5	1	5	0	5	0	1	0	0	0	3	2	10	4	5	5	2	5	1	1	2	5	0	0	10	3	2	1	10
2	0	1	6	0	0	5	0	1	0	2	5	6	5	2	5	5	5	1	0	3	1	2	0	2	10	10	5	2	1	2	1	0	10	2	5	0	5	0	4	0	2
5	2	1	5	2	0	6	4	2	0	0	0	0	2	0	2	1	0	0	3	0	0	2	0	4	4	1	1	0	5	1	5	10	1	0	0	5	10	10	4	5	2
5	1	1	5	2	1	4	0	0	1	5	0	5	2	2	0	1	0	0	1	0	0	5	5	2	5	2	10	0	2	0	2	4	0	2	5	5	2	2	2	5	0
0	1	5	0	2	2	0	0	0	2	2	0	2	0	0	4	0	2	0	2	2	5	0	2	0	2	0	2	0	5	2	0	2	1	5	10	6	1	5	2	0	0
10	0	1	5	0	5	2	2	6	4	5	0	1	0	0	0	3	2	3	0	0	5	2	0	2	10	0	2	0	0	5	3	5	2	0	10	10	3	0	2	1	0
0	0	2	2	5	5	1	5	1	0	1	1	1	1	0	0	5	6	2	2	4	2	0	2	0	1	4	0	5	4	1	0	2	2	1	1	0	4	0	3	5	5
1	3	2	1	5	1	2	3	5	0	4	10	5	2	5	0	0	10	10	10	4	5	2	10	1	0	2	10	5	0	1	5	0	0	10	2	5	0	0	0	0	1
4	5	0	0	5	2	1	2	0	5	0	5	2	0	5	0	2	0	4	10	1	2	0	0	4	2	0	2	0	2	3	0	5	0	3	2	0	2	10	0	1	2
2	10	1	0	0	1	2	2	0	0	1	1	1	5	2	0	1	2	5	5	1	10	2	2	0	10	2	0	3	2	5	0	10	2	0	5	5	2	1	0	0	0
0	0	1	1	1	10	2	5	0	3	10	1	0	2	5	2	5	0	5	2	0	0	0	0	5	5	0	3	0	0	0	0	3	6	5	0	5	2	1	5	0	2
0	5	5	2	5	0	0	2	0	2	3	5	0	4	0	2	1	2	2	1	5	2	5	0	4	0	2	2	0	0	1	0	5	4	10	5	1	0	10	5	0	3
0	0	2	3	0	0	2	2	5	5	5	5	10	5	5	0	5	2	5	2	1	0	2	5	1	1	3	5	0	1	0	2	0	5	2	0	2	0	10	3	0	0
2	2	6	2	4	0	3	2	10	0	5	2	5	10	1	0	6	10	1	1	5	2	0	3	0	5	0	0	0	0	2	0	2	4	1	2	0	0	2	2	0	6
0	1	1	10	2	2	2	5	10	0	0	1	5	4	5	5	0	2	1	0	10	4	2	5	2	0	5	10	3	5	0	2	0	2	5	0	0	0	2	2	0	0
5	1	1	5	2	5	5	0	3	1	5	5	2	0	1	2	3	0	2	10	1	0	1	2	2	0	0	2	6	4	5	4	2	0	2	1	0	10	0	0	1	10
5	3	0	4	1	0	0	2	5	5	2	10	2	10	5	0	5	5	5	2	0	2	5	0	1	10	3	0	5	10	2	1	5	2	0	5	2	0	2	1	0	0
0	10	10	0	5	10	0	5	1	2	0	5	5	2	0	5	0	1	0	5	0	5	10	10	1	2	2	5	0	5	0	2	0	1	5	0	0	2	5	0	3	5
10	5	5	0	6	5	0	2	2	0	0	3	1	10	5	1	5	0	0	0	5	5	6	10	0	5	0	5	5	1	2	0	0	0	2	0	0	10	10	5	0	1
10	0	10	2	2	4	0	2	0	1	0	5	1	1	5	0	0	6	10	5	10	2	1	3	4	0	2	2	2	0	0	0	0	10	0	2	10	0	2	10	0	0
0	10	5	10	2	5	1	0	0	5	5	5	0	2	2	5	0	3	3	0	10	2	5	0	0	0	10	1	1	10	10	2	2	0	2	5	10	2	0	1	0	4
0	0	5	3	0	5	1	5	10	2	0	1	1	5	10	0	5	4	2	4	4	2	2	2	3	0	0	0	5	5	3	2	2	0	1	0	5	10	1	0	2	1
0	1	2	0	5	2	6	1	1	0	0	5	0	4	5	2	5	5	1	0	5	5	0	1	5	0	1	0	0	0	0	0	0	1	0	3	0	0	0	2	0	5
2	4	1	5	5	5	1	3	6	10	5	5	0	1	5	0	10	5	10	2	2	0	0	0	5	1	2	0	2	3	0	6	0	10	0	5	1	0	4	1	5	0

Tabela A.10: Características da Instância Sko.49.1.

Comprimentos (l_i)

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Coeficiente de Demanda (l_i)

0	5	2	1	1	5	6	0	2	1	0	5	2	2	6	0	1	0	5	0	0	2	0	2	0	2	5	2	3	0	0	2	5	2	5	3	0	5	0	0	5	5	5	5	3	0	1	0	2
5	0	1	1	0	10	0	5	5	3	5	1	0	5	2	5	1	0	1	0	0	0	5	0	0	0	0	0	5	5	10	2	0	5	5	0	10	5	2	2	1	0	2	5	1	5	0	10	1
2	1	0	0	5	2	0	0	0	0	1	10	2	5	10	0	1	1	1	2	3	2	10	1	1	2	5	3	0	10	10	0	0	0	3	0	1	5	0	5	0	6	2	3	0	0	1	2	0
1	1	0	0	2	0	0	3	4	0	1	5	5	2	5	5	10	2	5	2	2	3	0	10	1	3	2	0	2	0	0	0	6	0	2	0	4	2	0	1	5	2	0	1	2	0	1	1	0
1	0	5	2	0	5	1	5	1	6	1	0	2	0	5	10	5	0	2	0	3	2	2	2	1	5	5	5	4	0	5	3	10	0	2	2	4	2	5	2	2	3	0	1	2	0	0	5	0
5	10	2	0	5	0	5	0	1	10	3	2	1	0	2	1	0	2	1	2	0	1	1	5	10	0	2	0	1	0	10	1	5	10	0	5	10	2	5	5	5	0	10	5	5	0	0	5	1
6	0	0	0	1	5	0	0	6	2	5	2	2	2	1	0	4	5	4	0	0	0	0	2	2	4	1	0	5	6	0	10	10	1	1	5	0	0	1	5	5	10	3	1	0	6	0	5	1
0	5	0	3	5	0	0	0	0	4	2	1	2	0	5	5	2	5	0	0	5	2	10	2	5	0	0	0	0	5	5	5	2	1	0	2	1	10	5	2	5	10	0	10	0	0	5	4	1
2	5	0	4	1	1	6	0	0	10	0	10	10	5	10	0	0	2	0	4	2	5	5	10	0	2	2	0	3	3	1	1	1	10	5	4	5	5	2	0	5	2	0	5	3	0	1	1	1
1	3	0	0	6	10	2	4	10	0	0	3	2	5	10	1	5	1	1	0	2	3	2	1	3	1	5	2	4	0	2	5	3	10	10	1	5	2	0	1	10	0	5	0	2	2	0	3	0
0	5	1	1	1	3	5	2	0	0	0	5	0	6	3	5	0	5	5	2	0	6	4	0	10	2	1	2	5	4	0	0	4	2	1	5	0	4	5	10	5	1	5	1	2	0	0	2	2
5	1	10	5	0	2	2	1	10	3	5	0	0	2	2	2	3	2	0	0	0	2	2	0	1	2	1	1	10	1	5	0	0	2	0	5	0	1	4	10	0	2	5	5	5	2	2	2	0
2	0	2	5	2	1	2	2	10	2	0	0	0	5	3	3	0	0	5	2	0	0	0	1	0	1	3	2	2	0	5	0	2	0	5	5	1	0	1	5	5	5	5	0	1	0	6	0	0
2	5	5	2	0	0	2	0	5	5	6	2	5	0	0	0	1	6	2	2	0	0	2	2	2	5	5	0	1	0	2	5	2	2	2	1	5	5	1	0	10	4	5	0	2	5	5	1	0
6	2	10	5	5	2	1	5	10	10	3	2	3	0	0	5	2	4	2	5	2	1	4	2	5	2	0	0	5	2	0	5	2	5	0	3	0	5	0	0	2	5	0	0	2	0	4	0	0
0	5	0	5	10	1	0	5	0	1	5	2	3	0	5	0	0	2	0	0	10	5	5	0	2	5	1	0	1	5	5	6	0	10	1	6	0	0	5	5	1	5	5	2	5	0	0	2	2
1	1	1	10	5	0	4	2	0	5	0	3	0	1	2	0	0	0	1	0	1	5	0	0	5	0	2	2	1	5	0	10	0	2	2	10	0	3	3	0	3	0	5	10	0	0	2	2	1
0	0	1	2	0	2	5	5	2	1	5	2	0	6	4	2	0	0	0	10	0	5	0	1	0	4	0	2	5	2	1	0	0	10	0	5	0	2	2	0	6	10	0	5	5	5	0	5	3
5	1	1	5	2	1	4	0	0	1	5	0	5	2	2	0	1	0	0	6	6	2	5	1	5	0	1	0	2	10	0	5	0	1	5	5	5	0	0	0	0	6	2	5	0	1	0	0	0
0	0	2	2	0	2	0	0	4	0	2	0	2	2	5	0	0	10	6	0	5	0	2	1	2	2	10	10	5	2	5	0	10	4	2	0	0	0	0	0	2	5	10	4	2	5	5	0	0
0	0	3	2	3	0	0	5	2	2	0	0	0	0	2	10	1	0	6	5	0	4	1	5	0	2	5	4	5	10	0	5	0	0	6	1	2	0	5	0	2	3	2	0	6	0	0	1	0
2	0	2	3	2	1	0	2	5	3	6	2	0	0	1	5	5	5	2	0	4	0	2	2	0	0	5	0	0	5	5	5	2	0	1	5	5	0	0	10	0	2	0	4	5	1	1	1	5
0	5	10	0	2	1	0	10	5	2	4	2	0	2	4	5	0	0	5	2	1	2	0	0	0	2	10	1	3	0	10	1	10	0	1	0	0	2	2	0	0	0	4	2	1	2	0	10	0
2	0	1	10	2	5	2	2	10	1	0	0	1	2	2	0	0	1	1	1	5	2	0	0	0	10	0	1	1	5	10	2	2	5	0	10	2	2	2	2	3	1	0	1	2	0	0	5	2
0	0	1	1	1	10	2	5	0	3	10	1	0	2	5	2	5	0	5	2	0	0	0	0	0	10	0	10	5	5	1	0	1	3	0	5	0	0	0	5	2	5	0	1	5	5	3	2	0
2	0	2	3	5	0	4	0	2	1	2	2	1	5	2	5	0	4	0	2	2	0	2	10	10	0	0	2	5	5	2	0	5	0	5	10	2	0	5	1	2	0	1	2	2	0	5	5	10
5	0	5	2	5	2	1	0	2	5	1	1	3	5	0	1	2	0	1	10	5	5	10	0	0	0	0	2	2	2	2	0	3	5	0	2	2	10	2	5	5	4	5	0	5	2	0	3	5
2	0	3	0	5	0	0	0	0	2	2	1	2	0	0	0	2	2	0	10	4	0	1	1	10	2	2	0	0	0	5	2	5	1	4	2	0	6	5	0	2	5	1	0	0	10	0	1	2
3	5	0	2	4	1	5	0	3	4	5	10	2	1	5	1	1	5	2	5	5	0	3	1	5	5	2	0	0	2	0	0	0	5	5	0	2	5	5	1	10	0	1	1	5	10	2	3	3
0	5	10	0	0	0	6	5	3	0	4	1	0	0	2	5	5	2	10	2	10	5	0	5	5	5	2	0	2	0	5	5	10	10	2	0	3	0	0	0	2	2	1	3	1	1	0	3	0
0	10	10	0	5	10	0	5	1	2	0	5	5	2	0	5	0	1	0	5	0	5	10	10	1	2	2	5	0	5	0	2	5	2	5	0	0	5	0	5	5	0	2	5	0	0	4	0	10
2	2	0	0	3	1	10	5	1	5	0	0	0	5	5	6	10	0	5	0	5	5	1	2	0	0	0	2	0	5	2	0	10	0	5	5	1	6	0	2	5	2	0	0	0	0	0	0	2
5	0	0	6	10	5	10	2	1	3	4	0	2	2	2	0	0	0	0	10	0	2	10	2	1	5	3	5	0	10	5	10	0	1	1	1	10	2	1	1	1	2	0	5	5	1	0	0	10
2	5	0	0	0	10	1	1	10	10	2	2	0	2	5	10	2	10	1	4	0	0	0	5	3	0	5	1	5	10	2	0	1	0	0	2	1	6	0	4	0	5	0	0	2	5	1	5	2
5	5	3	2	2	0	1	0	5	10	1	0	5	2	0	1	2	0	5	2	6	1	1	0	0	5	0	4	5	2	5	5	1	0	0	0	2	2	3	5	10	5	0	0	2	5	0	10	4
3	0	0	0	2	5	5	2	4	1	5	5	5	1	3	6	10	5	5	0	1	5	0	10	5	10	2	2	0	0	0	5	1	2	0	0	0	0	0	5	0	10	4	0	1	2	10	0	2
0	10	1	4	4	10	0	1	5	5	0	0	1	5	0	0	0	0	5	0	2	5	0	2	0	2	2	0	2	3	0	1	10	1	2	0	0	0	2	0	2	4	10	0	10	0	0	0	1
5	5	5	2	2	2	0	10	5	2	4	1	0	5	5	0	3	2	0	0	0	0	2	2	0	0	10	6	5	0	5	6	2	6	2	0	0	0	5	0	0	5	5	5	10	3	5	3	0
0	2	0	0	5	5	1	5	2	0	5	4	1	1	0	5	3	2	0	0	5	0	2	2	0	5	2	5	5	0	0	0	1	0	3	0	2	5	0	0	0	0	2	5	2	2	5	2	0
0	2	5	1	2	5	5	2	0	1	10	10	5	0	0	5	0	0	0	0	0	10	0	2	5	1	5	0	1	0	5	2	1	4	5	5	0	0	0	0	0	1	2	2	0	4	0	5	5
5	1	0	5	2	5	5	5	5	10	5	0	5	10	2	1	3	6	0	2	2	0	0	3	2	2	5	2	10	2	5	5	1	0	10	0	2	0	0	0	0	10	0	1	10	1	10	2	2
5	0	6	2	3	0	10	10	2	0	1	2	5	4	5	5	0	10	6	5	3	2	0	1	5	0	4	5	0	2	0	2	2	5	5	10	4	5	0	1	10	0	0	0	2	2	5	0	0
5	2	2	0	0	10	3	0	0	5	5	5	5	5	0	5	5	0	2	10	2	0	4	0	0	1	5	1	1	1	2	0	0	0	0	4	10	5	2	2	0	0	0	0	5	3	1	5	5
5	5	3	1	1	5	1	10	5	0	1	5	0	0	0	2	10	5	5	4	0	4	2	1	1	2	0	0	1	3	5	0	5	0	0	0	0	5	5	2	1	0	0	0	0	1	1	2	1
3	1	0	2	2	5	0	0	3	2	2	5	1	2	2	5	0	5	0	2	6	5	1	2	5	2	5	0	5	1	0	0	5	2	2	1	10	10	2	0	10	2	5	0	0	0	5	1	0
0	5	0	0	0	0	6	0	0	2	0	2	0	5	0	0	0	5	1	5	0	1	2	0	5	0	2	10	10	1	0	0	1	5	5	2	0	3	2	4	1	2	3	1	0	0	2	0	2
1	0	1	1	0	0	0	5	1	0	0	2	6	5	4	0	2	0	0	5	0	1	0	0	3	5	0	0	2	0	4	0	0	1	0	10	0	5	5	0	10	5	1	1	5	2	0	0	2
0	10	2	1	5	5	5	4	1	3	2	2	0	1	0	2	2	5	0	0	1	1	10	5	2	5	3	1	3	3	0	0	0	5	10	0	0	3	2	5	2	0	5	2	1	0	0	0	10
2	1	0	0	0	1	1	1	1	0	2	0	0	0	0	2	1	3	0	0	0	5	0	2	0	10	5	2	3	0	10	2	10	2	4	2	1	0	0	5	2	0	5	1	0	2	2	10	0

B Descrição dos Agentes

Com base nos componentes disponíveis na gramática, os melhores agentes de cada variante são apresentados a seguir. Para aquelas variantes cujos melhores indivíduos possuem a mesma aptidão, a escolha daquele que a representará foi feita aleatoriamente.

B.1 A_{S9}

```
1
2  void execute(I){
3      S = randomlyGenerate(I); Sb=S;
4      for(i={1,...,50}){
5          S.2-opt(); S.swap(0.65,0.50);
6          S.oppSidesSwap(); S.reconstrucion(0.15);
7          S.oppSidesSwap();
8          if (F(S)<F(Sb)){Sb=S;}
9      }
10 }
```

Figura B.1: Melhor heurística gerada pela variante A_{S9}.

B.2 A_{S10}

```
1
2  void execute(I){
3      S = randomlyGenerate(I); Sb=S;
4      for(i={1,...,50}){
5          S.reconstrucion(0.10);
6          S.shake(0.10); S.shake(0.55);
7          S.neighborSwap(); S.neighborSwap();
8          S.reconstrucion(0.00); S.shake(0.45);
9          S.2-opt(0.45); S.shake(0.45);
10         S.oppSidesSwap(); S.reconstrucion(0.45);
11         S.nonNeighborSwap();  S.oppSidesSwap();
12         S.oppSidesSwap(); S.shake(0.10);
13         S.2-opt(0.45); S.rvnd(0.45);
14         S.2-opt(0.45);
15         if (F(S)<F(Sb)){Sb=S;}
16     }
17 }
```

Figura B.2: Melhor heurística gerada pela variante A_{S10}.

B.3 A_{S11}

```
1
2  void execute(I){
3      S = randomlyGenerate(I); Sb=S;
4      for(i={1,...,50}){
5          S.repart(+0.70);
6          S.rec(0.30);
7          S.rvnd();
8          if (F(S)<F(Sb)){Sb=S;}
9      }
10 }
```

Figura B.3: Melhor heurística gerada pela variante A_{S11}.

B.4 A_{Am15}

```
1
2  void execute(I){
3      S = randomlyGenerate(I); Sb=S;
4      for(i={1,...,50}){
5          S.neighborSwap(); S.insertion();
6          S.2-opt(); S.rvnd();
7          S.swap(0.40,0.10);
8          S.nonNeighborSwap();
9          S.rvnd(); S.swap(0.70,0.70);
10         S.2-opt(); S.rvnd();
11         if (F(S)<F(Sb)){Sb=S;}
12     }
13 }
```

Figura B.4: Melhor heurística gerada pela variante A_{Am15}.

B.5 $A_{N25.1}$

```
1
2  void execute(I){
3      S = randomlyGenerate(I); Sb=S;
4      for(i={1,...,50}){
5          S.shake(0.60);
6          S.repart(-0.70);
7          S.reconstrucion(0.95);
8          S.oppSidesSwap();
9          S.rvnd();
10         if (F(S)<F(Sb)){Sb=S;}
11 }
```

Figura B.5: Melhor heurística gerada pela variante $A_{N25.1}$.

B.6 $A_{Sko42.1}$

```
1
2  void execute(I){
3      S = randomlyGenerate(I); Sb=S;
4      for(i={1,...,50}){
5          S.swap(0.85,0.20);
6          S.repart(+0.15);
7          S.rvnd();
8          S.rvnd();
9          S.neighborSwap();
10         S.repart(+0.20);
11         S.rvnd();
12         S.swap(0.45,0.15);
13         S.rvnd();
14         S.rvnd();
15         if (F(S)<F(Sb)){Sb=S;}
16 }
```

Figura B.6: Melhor heurística gerada pela variante $A_{Sko42.1}$.

B.7 $A_{Sko42.2}$

```
1
2  void execute(I){
3      S = randomlyGenerate(I); Sb=S;
4      for(i={1,...,50}){
5          S.repart(-0.50);
6          S.shake(-0.95);
7          S.2-opt(-0.95);
8          S.oppSidesSwap();
9          S.reconstruction(0.35);
10         S.rvnd();
11         S.2-opt(-0.95);
12         S.2-opt(-0.95);
13         if (F(S)<F(Sb)){Sb=S;}
14 }
```

Figura B.7: Melhor heurística gerada pela variante $A_{Sko42.2}$.

B.8 $A_{Sko42.3}$

```

void execute(I){
    S = randomlyGenerate(I); Sb=S;
    for(i={1,...,50}){
        S.rvnd();
        S.oppSidesSwap();
        S.shake(0.35);
        S.rvnd();
        S.nonNeighborSwap();
        S.shake(0.35);
        S.oppSidesSwap();
        S.2-opt(-0.95);
        S.swap(0.50,0.80);
        S.shake(0.35);
        S.neighborSwap();
        S.neighborSwap();
        S.rvnd();
        S.oppSidesSwap();
        if (F(S)<F(Sb)){Sb=S;}
}
```

Figura B.8: Melhor heurística gerada pela variante $A_{Sko42.3}$.

B.9 $A_{Sko42.4}$

```

void execute(I){
    S = randomlyGenerate(I); Sb=S;
    for(i={1,...,50}){
        S.oppSidesSwap();
        S.shake(0.90);
        S.shake(0.70);
        S.rvnd();
        if (F(S)<F(Sb)){Sb=S;}
}
```

Figura B.9: Melhor heurística gerada pela variante $A_{Sko42.4}$.

B.10 $A_{Sko49.1}$

```
1
2  void execute(I){
3      S = randomlyGenerate(I); Sb=S;
4      for(i={1,...,50}){
5          S.reconstruction(0.40);
6          S.rvnd();
7          S.rvnd();
8          S.2-opt();
9          if (F(S)<F(Sb)){Sb=S;}
10 }
```

Figura B.10: Melhor heurística gerada pela variante $A_{Sko49.1}$.

B.11 A_{Geral}

```
1
2  void execute(I){
3      S = randomlyGenerate(I); Sb=S;
4      for(i={1,...,50}){
5          S.reconstruction(0.40);
6          S.rvnd();
7          S.2-opt();
8          S.insertion();
9          S.reconstrucion(0.65);
10         S.rvnd();
11         S.insertion();
12         S.reconstruction(0.50);
13         S.rvnd();
14         S.swap(0.40,0.25);
15         S.rvnd();
16         if (F(S)<F(Sb)){Sb=S;}
17     }
18 }
```

Figura B.11: Melhor heurística gerada pela variante Geral.

Bibliografia

AHONEN, H.; ALVARENGA, A. G. de; AMARAL, A. R. S. Simulated annealing and tabu search approaches for the corridor allocation problem. *European Journal of Operational Research*, Elsevier, v. 232, n. 1, p. 221–233, 2014.

ALVARENGA, A. Gomes de et al. Metaheuristic methods for a class of the facility layout problem. *Journal of intelligent manufacturing*, Springer, v. 11, n. 4, p. 421–430, 2000.

AMARAL, A. R. On the exact solution of a facility layout problem. *European Journal of operational research*, Elsevier, v. 173, n. 2, p. 508–518, 2006.

AMARAL, A. R. An exact approach to the one-dimensional facility layout problem. *Operations Research*, INFORMS, v. 56, n. 4, p. 1026–1033, 2008.

AMARAL, A. R. The corridor allocation problem. *Computers & Operations Research*, Elsevier, v. 39, n. 12, p. 3325–3330, 2012.

AMARAL, A. R. Optimal solutions for the double row layout problem. *Optimization Letters*, Springer, v. 7, n. 2, p. 407–413, 2013.

AMARAL, A. R. A heuristic approach for the double row layout problem. *Annals of Operations Research*, Springer, p. 1–36, 2020.

AMARAL, A. R. A mixed-integer programming formulation of the double row layout problem based on a linear extension of a partial order. *Optimization Letters*, Springer, v. 15, n. 4, p. 1407–1423, 2021.

ANJOS, M. F.; KENNINGS, A.; VANNELLI, A. A semidefinite optimization approach for the single-row layout problem with unequal dimensions. *Discrete Optimization*, Elsevier, v. 2, n. 2, p. 113–122, 2005.

ANJOS, M. F.; VANNELLI, A. Computing globally optimal solutions for single-row layout problems using semidefinite programming and cutting planes. *INFORMS Journal on Computing*, INFORMS, v. 20, n. 4, p. 611–617, 2008.

ANJOS, M. F.; VIEIRA, M. V. Mathematical optimization approaches for facility layout problems: The state-of-the-art and future research directions. *European Journal of Operational Research*, Elsevier, v. 261, n. 1, p. 1–16, 2017.

ANJOS, M. F.; YEN, G. Provably near-optimal solutions for very large single-row facility layout problems. *Optimization Methods & Software*, Taylor & Francis, v. 24, n. 4-5, p. 805–817, 2009.

BARBOSA, H. J. C.; BERNARDINO, H. S.; BARRETO, A. M. S. Using performance profiles to analyze the results of the 2006 CEC constrained optimization competition. In: IEEE. *Evolutionary Computation (CEC), 2010 IEEE Congress on*. [S.l.], 2010. p. 1–8.

BURKE, E. K. et al. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, Springer, v. 64, n. 12, p. 1695–1724, 2013.

CELLIN, B. D. P. *Métodos para Resolução eficiente de Problemas de Layout*. 73 p. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, Vitória, 2017.

CHAE, J.; REGAN, A. C. A mixed integer programming model for a double row layout problem. *Computers & Industrial Engineering*, Elsevier, v. 140, p. 106244, 2020.

CHUNG, J.; TANCHOCO, J. The double row layout problem. *International Journal of Production Research*, Taylor & Francis, v. 48, n. 3, p. 709–727, 2010.

CORREA, R. F. et al. A grammar-based genetic programming hyper-heuristic for corridor allocation problem. In: SPRINGER. *Intelligent Systems: 11th Brazilian Conference, BRACIS 2022, Campinas, Brazil, November 28–December 1, 2022, Proceedings, Part I*. [S.l.], 2022. p. 504–519.

COWLING, P.; KENDALL, G.; SOUBEIGA, E. A hyperheuristic approach to scheduling a sales summit. In: SPRINGER. *Practice and Theory of Automated Timetabling III: Third International Conference, PATAT 2000 Konstanz, Germany, August 16–18, 2000 Selected Papers 3*. [S.l.], 2001. p. 176–190.

CRAVO, G. L.; AMARAL, A. R. S. A grasp algorithm for solving large-scale single row facility layout problems. *Computers & Operations Research*, Elsevier, v. 106, p. 49–61, 2019.

DATTA, D.; AMARAL, A. R.; FIGUEIRA, J. R. Single row facility layout problem using a permutation-based genetic algorithm. *European Journal of Operational Research*, Elsevier, v. 213, n. 2, p. 388–394, 2011.

DOLAN, E. D.; MORÉ, J. J. Benchmarking optimization software with performance profiles. *Math. Program.*, v. 91, n. 2, p. 201–213, Jan 2002.

EL-BAZ, M. A. A genetic algorithm for facility layout problems of different manufacturing environments. *Computers & Industrial Engineering*, v. 47, n. 2, p. 233–246, 2004. ISSN 0360-8352. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360835204001159>.

FESTA, P. A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. In: IEEE. *2014 16th International Conference on Transparent Optical Networks (ICTON)*. [S.l.], 2014. p. 1–20.

FICKO, M.; BREZOCNIK, M.; BALIC, J. Designing the layout of single- and multiple-rows flexible manufacturing system by genetic algorithms. *Journal of Materials Processing Technology*, v. 157-158, p. 150–158, 2004. ISSN 0924-0136. Achievements in Mechanical and Materials Engineering Conference. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0924013604010544>.

FICKO, M. et al. Intelligent design of an unconstrained layout for a flexible manufacturing system. *Neurocomputing*, v. 73, n. 4, p. 639–647, 2010. ISSN 0925-2312. Bayesian Networks / Design and Application of Neural Networks and Intelligent Learning Systems (KES 2008 / Bio-inspired Computing: Theories and Applications (BIC-TA 2007). Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231209004056>.

FISCHER, A.; FISCHER, F.; HUNGERLÄNDER, P. New exact approaches to row layout problems. *Mathematical Programming Computation*, Springer, v. 11, n. 4, p. 703–754, 2019.

FREITAS, J. M. de; SOUZA, F. R. de; BERNARDINO, H. S. Evolving controllers for mario ai using grammar-based genetic programming. In: IEEE. *2018 IEEE Congress on Evolutionary Computation (CEC)*. [S.l.], 2018. p. 1–8.

GHOSH, D.; KOTHARI, R. Population heuristics for the corridor allocation problem. 2012.

GOGNA, A.; TAYAL, A. Metaheuristics: review and application. *Journal of Experimental & Theoretical Artificial Intelligence*, Taylor & Francis, v. 25, n. 4, p. 503–526, 2013.

GUAN, J.; LIN, G. Hybridizing variable neighborhood search with ant colony optimization for solving the single row facility layout problem. *European Journal of Operational Research*, Elsevier, v. 248, n. 3, p. 899–909, 2016.

HASSAN, M. M. Machine layout problem in modern manufacturing facilities. *The International Journal of Production Research*, Taylor & Francis, v. 32, n. 11, p. 2559–2584, 1994.

HERAGU, S. S.; ALFA, A. S. Experimental analysis of simulated annealing based algorithms for the layout problem. *European Journal of Operational Research*, Elsevier, v. 57, n. 2, p. 190–202, 1992.

HERAGU, S. S.; ALFA, A. S. Experimental analysis of simulated annealing based algorithms for the layout problem. *European Journal of Operational Research*, v. 57, n. 2, p. 190–202, 1992. ISSN 0377-2217. Facility Layout. Disponível em: <https://www.sciencedirect.com/science/article/pii/0377221792900428>.

HERAGU, S. S.; KUSIAK, A. Efficient models for the facility layout problem. *European Journal of Operational Research*, Elsevier, v. 53, n. 1, p. 1–13, 1991.

HERRÁN, A.; COLMENAR, J. M.; DUARTE, A. An efficient variable neighborhood search for the space-free multi-row facility layout problem. *European Journal of Operational Research*, Elsevier, v. 295, n. 3, p. 893–907, 2021.

HOPCROFT, J. E.; ULLMAN, J. D.; MOTWANI, R. Introdução à teoria de autômatos, linguagens e computação. *Rio de Janeiro: Campus*, 2002.

KARP, R. M.; HELD, M. Finite-state processes and dynamic programming. *SIAM Journal on Applied Mathematics*, SIAM, v. 15, n. 3, p. 693–718, 1967.

KOTHARI, R.; GHOSH, D. Tabu search for the single row facility layout problem using exhaustive 2-opt and insertion neighborhoods. *European Journal of Operational Research*, Elsevier, v. 224, n. 1, p. 93–100, 2013.

KOTHARI, R.; GHOSH, D. A scatter search algorithm for the single row facility layout problem. *Journal of Heuristics*, Springer, v. 20, n. 2, p. 125–142, 2014.

KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. [S.l.]: The MIT Press, 1992.

KUMAR, R. S. et al. Scatter search algorithm for single row layout problem in fms. *Advances in Production Engineering & Management*, v. 3, n. 4, p. 193–204, 2008.

KUSIAK, A.; HERAGU, S. S. The facility layout problem. *European Journal of operational research*, Elsevier, v. 29, n. 3, p. 229–251, 1987.

LEE, Y. H.; LEE, M. H. A shape-based block layout approach to facility layout problems using hybrid genetic algorithm. *Computers & Industrial Engineering*, v. 42, n. 2, p. 237–248, 2002. ISSN 0360-8352. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360835202000189>.

LOIOLA, E. M. et al. A survey for the quadratic assignment problem. *European journal of operational research*, Elsevier, v. 176, n. 2, p. 657–690, 2007.

LOZANO, M.; MOLINA, D.; GARCíA-MARTíNEZ, C. Iterated greedy for the maximum diversity problem. *European Journal of Operational Research*, v. 214, n. 1, p. 31–38, 2011. ISSN 0377-2217.

MCKIGHT, P. E.; NAJAB, J. Kruskal-wallis test. *The corsini encyclopedia of psychology*, Wiley Online Library, p. 1–1, 2010.

MIAO, Z.; XU, K.-l. Research of multi-rows facility layout based on hybrid algorithm. In: IEEE. *2009 International Conference on Information Management, Innovation Management and Industrial Engineering*. [S.l.], 2009. v. 2, p. 553–556.

OZCELIK, F. A hybrid genetic algorithm for the single row layout problem. *International Journal of Production Research*, Taylor & Francis, v. 50, n. 20, p. 5872–5886, 2012.

PALUBECKIS, G. Fast local search for single row facility layout. *European Journal of Operational Research*, Elsevier, v. 246, n. 3, p. 800–814, 2015.

PALUBECKIS, G. Single row facility layout using multi-start simulated annealing. *Computers & Industrial Engineering*, Elsevier, v. 103, p. 1–16, 2017.

SAMARGHANDI, H.; ESHGHI, K. An efficient tabu algorithm for the single row facility layout problem. *European Journal of Operational Research*, Elsevier, v. 205, n. 1, p. 98–105, 2010.

SAMARGHANDI, H.; TAABAYAN, P.; JAHANTIGH, F. F. A particle swarm optimization for the single row facility layout problem. *Computers & Industrial Engineering*, Elsevier, v. 58, n. 4, p. 529–534, 2010.

SANTANA, C. A.; SANTOS, A. G. dos. Uma heurística baseada na busca local iterada para o problema de alocação de corredor. 2016.

SIMMONS, D. M. One-dimensional space allocation: an ordering algorithm. *Operations Research*, INFORMS, v. 17, n. 5, p. 812–826, 1969.

SOLIMANPUR, M.; VRAT, P.; SHANKAR, R. An ant algorithm for the single row layout problem in flexible manufacturing systems. *Computers & Operations Research*, Elsevier, v. 32, n. 3, p. 583–598, 2005.

TALBI, E.-G. *Metaheuristics: from design to implementation*. [S.l.]: John Wiley & Sons, 2009.

TANIYAMA TAVARES, L. Proposta de um algoritmo hill-climbing para solucionar o problema de arranjo físico com múltiplos corredores. *Congresso Brasileiro de Engenharia de Produção*, 2018.

WHIGHAM, P. A. Grammatically-based genetic programming. In: ROSCA, J. P. (Ed.). *Proc. of the Workshop on Genetic Programming: From Theory to Real-World Applications*. Tahoe City, USA: University of Rochester, 1995. p. 33–41.