UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
UNIVERSIDADE FEDERAL RURAL DO SEMIÁRIDO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ALEXSANDRO TRINDADE SALES DA SILVA

DESENVOLVIMENTO DE UM FRAMEWORK PARA UTILIZAÇÃO DO GR-LEARNING EM PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA

MOSSORÓ - RN
2016

ALEXSANDRO TRINDADE SALES DA SILVA

DESENVOLVIMENTO DE UM FRAMEWORK PARA UTILIZAÇÃO DO GR-LEARNING EM PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação - associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semiárido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Francisco Chagas de Lima Júnior

Co-orientador: Prof. Dr. Carlos Heitor Pereira Liberalino

MOSSORÓ - RN
2016

Catalogação da Publicação na Fonte.

Silva, Alexsandro Trindade Sales da
Desenvolvimento de um framework para utilização do gr-learning em problemas de otimização combinatória. / Alexsandro Trindade Sales da Silva. - Mossoró, RN, 2016.

99 p.
Orientador(a): Prof. Dr. Francisco Chagas de Lima Júnior

Dissertação (Mestrado em Ciência da Computação). Universidade do Estado do Rio Grande do Norte. Universidade Federal Rural do Semi-Árido.

1. *Framework. 2. M*etaheurística. 3. GRASP - Aprendizado por reforço. I. Lima Júnior, Francisco Chagas de. II. Universidade do Estado do Rio Grande do Norte . III.Título.

UERN/SIB/BC CDD 005

Bibliotecária: Jocelania Marinho Maia de Oliveira CRB 15 / 319

ALEXSANDRO TRINDADE SALES DA SILVA

DESENVOLVIMENTO DE UM FRAMEWORK PARA UTILIZAÇÃO DO GR-LEARNING EM PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: 20 /07 /2016.

BANCA EXAMINADORA

Dr. Francisco Chagas de Lima Júnior - UERN
Orientador

Dr. Carlos Heitor Pereira Liberalino - UERN
Co-orientador

Dr. Dario José Aloise - UERN
Membro Interno

Dr. Francisco Dantas de Medeiros Neto - UERN
Membro Interno

Dr. João Paulo de Souza Medeiros – DCT/UFRN
Membro Externo

Dedico este trabalho a minha mãe, esposa e filhos.

AGRADECIMENTOS

Agradeço ao meu Deus que mais uma vez honrou o trabalho de minhas mãos e me permitiu mais essa importante conquista em minha vida acadêmica.

Aos meus avós (*in memoriam*) que sempre me mostraram a honestidade, educação e o trabalho como opções para o crescimento pessoal e profissional.

A minha mãe por todo o esforço que fez, e por ter acreditado no sonho de uma criança que aos treze anos queria apenas fazer um curso de informática. A esta sou eternamente grata.

A minha amada esposa que desde o namoro teve que aprender a me dividir com os livros.

Aos meus orientadores Prof. Dr. Francisco Chagas de Lima Júnior e Prof. Dr. Carlos Heitor Pereira Liberalino, pela paciência e conhecimentos repassados durante as disciplinas ministradas e durante todo o desenvolvimento deste trabalho.

Aos professores Dario Aloise, Jéssica Neiva e Max Lopes pelas vezes que me indicaram para o programa com suas cartas de recomendação.

Ao meu amigo sempre prestativo Francisco Márcio de Oliveira, o qual tive o prazer de ser aluno, colega de trabalho e hoje amigo, que sempre me incentivou, indicou e acreditou que eu seria bem sucedido nesta caminhada.

A PEGGASUS, empresa onde trabalhei durante oito ótimos anos, e que sempre me apoiou e compreendeu meus momentos de ausência.

Aos amigos Rennê, Rafael e Clayton pela parceria firmada durante o período de curso das disciplinas.

A Universidade do Estado do Rio Grande do Norte e Universidade Federal Rural do Semiárido por permitirem a minha qualificação acadêmica.

“Mas buscais em primeiro lugar o reino de Deus, e a sua justiça, e todas estas coisas vos serão acrescentadas”. Mateus 6:33

RESUMO

A utilização de metaheurísticas para resolução de problemas de otimização combinatória pertencentes à classe NP-Difícil vem se tornando cada vez mais comum, e segundo Temponi (2007 apud RIBEIRO, 1996) uma metaheurística deve ser modelada de acordo com o problema que ela foi projetada para resolver. Isto na maioria vezes requer muitas alterações quando se tem que aplicar uma mesma metaheurística a diversos tipos de problemas de otimização combinatória. Neste trabalho foi proposto um *framework* para utilização de uma metaheurística híbrida proposta por Almeida (2014) que utilizou a metaheurística *GRASP* Reativo juntamente com uma técnica de aprendizagem por reforço (denominada *GR-Learning*). Especificamente, o algoritmo *Q-learning*, que foi utilizado para aprender com o passar das iterações qual valor para o parâmetro α (alfa) utilizar durante a fase de construção da *GRASP*. O *GR-Learning* foi utilizado para resolver o problema dos *p*-Centros aplicado a Segurança Pública na Cidade de Mossoró/RN. Para validar a eficácia do *framework* proposto o mesmo foi utilizado para resolver dois problemas clássicos de otimização combinatória: O Problema de Localização de Hubs (do inglês *Hub Location Problem* - *HLP*) e o Problema de Corte e Estoque – PCE (do inglês *Cutting Stock Problem* - *CSP*). Para validação dos resultados obtidos foram utilizadas instâncias com resultados já conhecidos na literatura e adicionalmente foi criada uma instância com dados do setor aeroviário Brasileiro. Os resultados obtidos mostraram que o *framework* proposto foi bastante competitivo quando comparado a outros resultados de diversos algoritmos já conhecidos na literatura, pois obteve o valor ótimo em quase todas as instâncias do *HLP* como também novos valores (melhores que os obtidos com outros algoritmos já conhecido na literatura) para algumas instâncias do *CSP*.

Palavras-chave: *Framework*, Metaheurística, GRASP, Aprendizado por Reforço.

ABSTRACT

The use of metaheuristics for solving combinatorial optimization problems belong to NP-Hard class is becoming increasingly common, and second Temponi (2007 apud RIBEIRO, 1996) a metaheurist should be modeled according to the problem she was designed to solve. This most often requires many changes when you have to apply the same metaheuristic to various types of combinatorial optimization problems. In this work we propose a framework for use of a hybrid metaheuristic proposed by Almeida (2014) who used the GRASP Reactive along with a reinforcement learning technique (called GR-learning). Specifically, the Q-learning algorithm that was used to learn over which the iterations value for the parameter α (alpha) used during the construction phase of GRASP. The GR-Learning was used to solve the problem of p-centers applied to Public Security in the city of Mossoró/RN. To validate the effectiveness of the framework proposed it was used to solve two classical problems of combinatorial optimization: The Hub Location Problem (HLP), and the Cutting Stock Problem (CSP). To validate the results obtained we used instances with results known in the literature and in addition has created an instance with data from the Brazilian airline industry. The results showed that the proposed framework was quite competitive when compared to other results of different algorithms known in the literature as got great value in almost all instances of HLP as well as new values (better than those obtained with other algorithms known in the literature) for some instances of CSP.

Key-words: Framework, Metaheuristic, GRASP, Reinforcement Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP – Australian Post

AR – Aprendizagem por Reforço

CAB – *Civil Aeronautics Board*

CSHLP – Capacitated Single Allocation Hub Location Problem

CSSATL – Clustering Search Simulated Annealing Tabu List

CSP – Cutting Stock Problem

DAC – Departamento de Aviação Comercial

DP – Dynamic Programming

FAA – US Federal Aviation Administration

FH – Faixa Horizontal

FLP – Facility Location Problem

FV – Faixa Vertical

GA – Genetic Algorithms

GRASP – Greedy Randomized Adaptive Search Procedure

HLP – Hub Location Problem

ICAIO – International Civil Aviation Organization

ILS – Iterated Local Search

NE – Naive Evolution

ODP – Open Dimensional Problem

PAX – Passageiros

PCB – Problema de Corte Bidimensional

PCE – Problema de Corte e Estoque

PCV – Problema do Caixeiro Viajante

PDM – Processo de Decisão de Markov

PPC – Problema dos p-Centros

PSO – Particle Swarm Optimiztion

R_{fh} – Resto da Faixa Horizontal

R_{fv} – Resto da Faixa Vertical

RPK – Revenue Passenger Kilometers

RL – Reinforcement Learning

SA – *Simulated Annealing*

SATL – Simulated Annealing Tabu List

SPAM – Surplus Plate Application Module

TSP – Traveling Salesman Problem

UHPS – Uncapacitated Hub Location Problem with Single Allocation

UMAHLP – Uncapacitaded Multiple Allocation Hub Location Problem

USAHLP – Uncapacitated Single Allocation Hub Location Problem

USApHMP – Uncapacitated Single Allocation p-Hub Median Problem

TAC – Turkish Air Cargo

VNS – Variable Neighboard Search

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO

O processo de tomada de decisões está presente no cotidiano de todos. Escolher uma alternativa dentre um conjunto de possíveis soluções a serem tomadas é uma tarefa simples, porém escolher a melhor solução possível dentre as soluções disponíveis passa a ser uma tarefa mais complexa. Ao processo de encontrar a melhor solução dentre um conjunto de possibilidades dá-se o nome de otimização (BECCENARI, 2012).

Uma maneira para encontrar a melhor solução seria verificar todas as soluções possíveis e escolher a de menor custo (para os problemas de minimização) ou escolher a de maior lucro ou de melhor aproveitamento (para problemas de maximização), porém em diversos problemas existentes no mundo real, essa técnica seria impraticável, mesmo que fosse utilizado para isso um computador de última geração.

Este trabalho aborda dois problemas clássicos da área de pesquisa operacional: O Problema de Localização de Hubs (do inglês *Hub Location Problem – HLP),* que é um caso especial do clássico Problema de Localização de Facilidades (do inglês *Facility Location Problem – FLP*), e o Problema de Corte e Estoque (do inglês *Cutting Stock Problem - CSP*), que devido à sua complexidade computacional, ambos estão inseridos na classe de problemas NP-Difícil. (KORTE; VYGEN, 2008)

Para a resolução desses tipos de problemas é muito comum à utilização de metaheurísticas, que fornecem uma boa solução em um tempo computacional viável, porém sem garantir que a solução encontrada seja a solução ótima, embora em alguns casos a solução ótima seja alcançada.

Dentre as diversas metaheurísticas existentes na literatura, a metaheurística *GRASP* (do inglês *Greedy Randomized Adaptive Search Procedure*), tem apresentado bons resultados na resolução de diversos problemas de otimização combinatória como, por exemplo: Problema do Caixeiro Viajante – PCV (do inglês *Traveling Salesman Problem - TSP*), *FLP, CSP* e diversos outros.

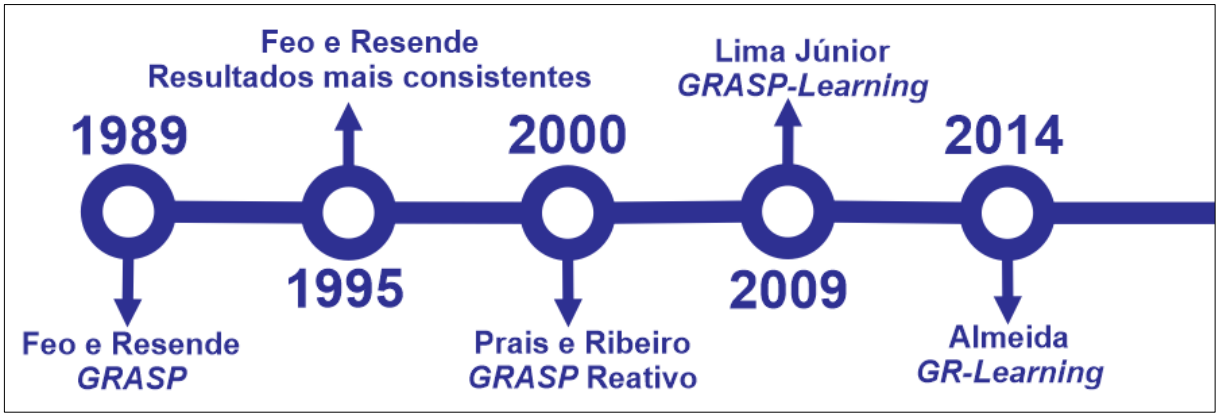


Figura 1 – Evolução da metaheurística *GRASP*.

Conforme pode ser visto na Figura 1, a *GRASP* foi proposta por Feo e Resende (1989), porém somente em Feo e Resende (1995) foram apresentados resultados mais consistentes. A metaheurística proposta, como também a sua evolução serão abordados no decorres desse trabalho.

A *GRASP* possui duas fases distintas, a primeira chamada de fase de construção e a segunda chamada de fase de busca local, ou fase de melhoria. Na primeira fase uma solução é criada e serve como ponto de partida para a segunda fase, onde é feita uma busca no espaço de soluções. Este procedimento se repete até que um critério de parada seja alcançado.

A *GRASP* possui um importante parâmetro chamado de α (alfa), este parâmetro tem seu valor compreendido entre [0, 1] e o mesmo é responsável por controlar a intensidade de gula ou aleatoriedade das soluções criadas na fase de construção. Para $\alpha = 0$ temos soluções gulosas, e para $\alpha = 1$ temos soluções semi-gulosas (LIMA JÚNIOR, 2009).

Prais e Ribeiro (2000) apresentaram uma versão melhorada do *GRASP*, o *GRASP* Reativo, em que diferentemente do *GRASP* tradicional as iterações não são feitas com apenas um α predeterminado, de acordo com as soluções obtidas e a partir de um conjunto de α predeterminados, os α que gerarem soluções mais distantes da melhor solução encontrada até o momento, recebem uma punição, e dessa forma os α com melhor desempenho tem uma maior probabilidade de serem escolhidos em futuras iterações, porém sem garantia que esses sejam escolhidos (ALMEIDA, 2014).

Versões híbridas da GRASP vem sendo utilizadas por diversos pesquisadores, Lima Júnior (2009) propôs a utilização da *GRASP* com Aprendizagem por Reforço – AR (do inglês *Reinforcement Learning - RL*). A AR possibilita um agente aprender a partir da iteração com o ambiente ao qual está inserido. Em Lima Júnior (2009) a fase de construção da *GRASP* foi

substituída por uma estratégia de geração de uma solução para o PCV a partir da matriz dos Q-valores, que representa o conhecimento adquirido do algoritmo *Q-learning*.

Em sua dissertação, Almeida (2014) propôs um algoritmo híbrido utilizando a *GRASP* Reativo juntamente com o *Q-learning* (*GR-Learning*), porém com uma diferença crucial, no método proposto o algoritmo original não sofre alterações, visto que o *Q-learning* é utilizado para aprender com o passar das iterações qual o melhor α deve ser utilizado. O método proposto foi utilizado para resolver o Problema dos *p*-Centros (PPC) aplicado à localização de pontos estratégicos para instalação de bases policiais na cidade de Mossoró/RN.

O presente trabalho apresenta o desenvolvimento de um *framework* para utilização do algoritmo *GR-Learning* proposto em Almeida (2014) para resolução de diversos problemas de otimização combinatória, sejam eles com intuito de minimizar os custos ou maximizar o lucro ou aproveitamento, de maneira que seja necessário apenas estender uma classe e sobrescrever dois métodos: `geraSolução`, que será responsável por gerar a solução inicial, e `buscaLocal`, que será utilizado durante a segunda fase da metaheurística para refinamento das solução obtida na fase anterior. Assim é desnecessário que o pesquisador altere o restante do algoritmo ou mesmo tenha que se aprofundar em como o algoritmo *Q-learning* funciona.

Para validação do *framework* proposto foram utilizadas instâncias com os valores ótimos já conhecidos na literatura. Para o *HLP* foi utilizado a instância CAB (*Civil Aeronautics Board*), introduzida na literatura por O´Kelly (1987), foi utilizada por outros pesquisadores para validação de métodos propostos para resolução do *HLP*. Foi criada uma segunda instância com base nos dados publicados em ANAC (2011). A publicação apresenta os dados estatísticos e econômicos referentes ao transporte aéreo brasileiro no ano de 2010.

Para validação do *CSP* foi abordada uma variação desse problema, especificamente, o Problema de Corte Bidimensional (PCB). Foram utilizados três conjuntos de instâncias também com valor ótimo conhecido na literatura. Essas instâncias são utilizadas para validações de métodos propostos para solucionar o *CSP*.

1.2. MOTIVAÇÃO

Durante a implementação de uma metaheurística é comum o código desenvolvido ficar fortemente acoplado ao problema para o qual foi projetado. Dessa maneira, quando é preciso utilizar a mesma implementação para resolver um problema diferente do qual foi planejado inicialmente, se faz necessário um grande esforço para adequar o código já desenvolvido.

Diante dessa dificuldade nasceu a ideia de desenvolver um framework para utilização do algoritmo *GR-Learning* em diversos problemas de otimização combinatória, não sendo necessário tanto esforço para adaptar o código já desenvolvido a cada novo problema aplicado.

A escolha de utilizar como validação do *framework* o problema de localização de hubs e o problema de corte e estoque se deu devido à aplicação em diversos setores da indústria, redes de telecomunicações, logística e transporte. Outro fator levado em consideração na escolha foi por serem problemas completamente distintos um do outro. Enquanto no problema de corte bidimensional temos que encontrar um padrão de corte que forneça o melhor aproveitamento das placas utilizadas (problemas de maximização), no problema de localização de hubs deve-se localizar e alocar hubs (*concentradores*) que sirvam de pontos de comutação e distribuição entre outros pontos da rede (*Spoke*) de maneira a minimizar o custo total de transporte da rede (problemas de minimização).

1.3. OBJETIVO GERAL

- Desenvolver um *framework* para utilização do *GR-Learning* em problemas de otimização combinatória.

1.4. OBJETIVOS ESPECÍFICOS

- Aplicar o *framework* desenvolvido a dois problemas clássicos da pesquisa operacional: o Problema de Corte Bidimensional e o Problema de Alocação de Hubs.
- Apresentar uma nova instância com dados referentes ao sistema aeroviário Brasileiro.
- Validar os resultados obtidos com o *framework* proposto.

1.5. ORGANIZAÇÃO DA DISSERTAÇÃO

O restante do presente trabalho está dividido desta forma:

- O Capítulo 2 apresenta a fundamentação teórica para o Problema de Localização de Hubs (*HLP*), Problema de Corte e Estoque (*CSP*), os modelos matemáticos abordados juntamente com a revisão da literatura.

- O Capítulo 3 traz uma breve explanação sobre as metaheurísticas *GRASP, GRASP* Reativo e *GRASP-2d*.

- O Capítulo 4 apresenta os conceitos de Aprendizagem por Reforço (AR) e os algoritmos híbridos *GRASP Learning* e *GR-Learning*.

- O Capítulo 5 apresenta o *framework* proposto.

- O Capítulo 6 apresentará as instâncias utilizadas, como também o detalhamento do processo de criação da instância BR2010 com dados do setor aeroviário brasileiro.

- O Capítulo 7 apresentará os resultados dos testes computacionais.

- O Capítulo 8 faz uma discussão geral sobre as contribuições desse trabalho juntamente com considerações finais e sugestões de trabalhos futuros.

- O Apêndice apresenta as Tabelas com os dados utilizados para criação da instância BR2010, e também a representação gráfica para algumas instâncias do PCB.

CAPÍTULO 2

REFERENCIAL TEÓRICO

Neste capítulo são apresentados os fundamentos acerca dos temas abordados no presente trabalho. As próximas seções trazem a definição e os principais conceitos sobre Problemas de Localização de Hubs (*HLP*) e Problema de Corte e Estoque (*CSP*).

2.1. PROBLEMAS DE LOCALIZAÇÃO DE HUBS

Os Problemas de Localização de Hubs são uma variação do clássico Problema de Localização de Facilidades, este foi introduzido na literatura por Alfred Weber em 1909 (OLIVEIRA, 2013), e segundo Ribeiro (2008), determinar a melhor localização para instalações de filiais em um horizonte de planejamento é um relevante desafio para manter a competitividade de uma empresa, e Oliveira (2013 apud SULE, 2001) listou diversas áreas em que o FLP pode ser aplicado:

a) Selecionar a localização para serviços de emergência, como hospitais e corpo de bombeiros, ou a distribuição de postos policiais no território;
b) Determinar a melhor localização para sala/mesa de ferramentas, máquinas, fontes de água, áreas de lavagem e salas de primeiros socorros, em uma fábrica;
c) Escolher a localização de armazéns e centros de distribuição;
d) Selecionar prestadores de serviços e atribuir trabalho adequado para cada um;
e) Escolher fornecedores e selecionar os itens para adquirir de cada um;
f) Escolher localização para departamentos de manutenção
g) A escolha de localização e capacidades de máquinas para atender às demandas de clientes distribuídos por uma determinada área;
h) Desenvolver um *layout* para oficina mecânica ou um painel de instrumentos;
i) Selecionar local para uma facilidade indesejável, tais como estações de coleta e tratamento de esgotos e aterros sanitários.

De acordo com Alumur e Kara (2008), o primeiro trabalho sobre o HLP surgiu em 1986 com O'Kelly, que mais tarde em 1987 apresentou a primeira formulação matemática

reconhecida para o HLP, o autor abordou uma rede de passageiros de companhias aéreas, introduzindo assim um conjunto de dados baseados em informações de companhias aéreas de passageiros de 25 cidades norte-americanas nos anos 70, avaliadas pelo Conselho de Aeronáutica Civil (CAB). Este mesmo conjunto de dados foi posteriormente utilizado por diversos outros trabalhos e são referenciados como conjunto de dados CAB.

2.1.1. Variações do *Hub Location Problem*

Diversas aplicações práticas em redes de transporte e de telecomunicações, como o transporte de pessoas, produtos, materiais ou de dados, ocorrem em redes deste tipo e existem diferentes versões do *HLP*, vejamos algumas:

Tabela 1 – Classificação para o *HLP* (ALUMUR; KARA, 2008).

Classificação	Descrição
Problema Não-Capacitado de Localização de Hubs (*USAHLP - Uncapacitated Single Allocation Hub Location Problem*)	Quando não existe restrição quanto ao fluxo que passa pelos nós, e cada nó de demanda não pode ser alocado a mais de um nó hub.
Problema Não-Capacitado de Localização de *p*-Hubs (*USApHMP - Uncapacitated Single Allocation p-Hub Median Problem*)	Quando número de hubs a serem alocados é fixo (por exemplo, igual a p).
Problema Não-Capacitado de Localização de Hubs com Alocação Múltipla (*UMAHLP - Uncapacitaded Multiple Allocation Hub Location Problem*)	Quando um nó de demanda pode ser alocado a mais de um nó hub.
Problema Capacitado de Localização de Hubs (*CSAHLP - Capacitated Single Allocation Hub Location Problem*)	Quando existe restrição quanto ao fluxo que passa pelos nós, e cada nó de demanda não pode ser alocado a mais de um nó hub.

2.1.2. O modelo *Hub and Spoke*

Segundo Almeida (2009), em diversos problemas de redes o fluxo entre os nós da rede não acontece de forma direta, mas por meio de nós especiais denominados hubs (concentrado-

res), nestes casos, diz-se que a rede é do tipo *hub and spoke*. Uma vez que os hubs são escolhidos o fluxo origem/destino é encaminhado através dos hubs e das arestas que os ligam.

Em 1978 o governo americano promoveu o ato de desregulamentação das empresas aéreas com o intuito de promover a concorrência e expandir o transporte aéreo nos EUA. O resultado dessa medida foi o surgimento de novas companhias e as chamadas companhias de baixo custo, o que forçou a mudança do modelo ponto a ponto para o modelo *hub and spoke* (ALMEIDA, 2012). Na Figura 2 é apresentada uma rede do tipo hub and spoke, os retângulos são os nós hub, e os círculos são os nós *Spoke*, e todo tráfego entre dois nós *Spoke* são roteados por um nó hub.

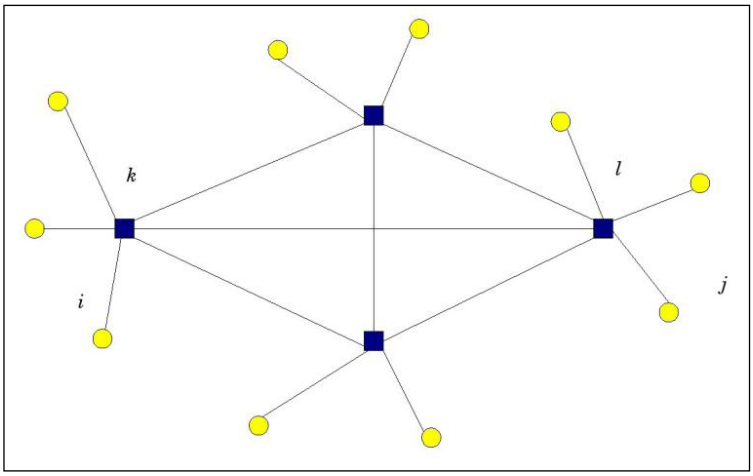


Figura 2 – Rede do tipo *Hub and Spoke* (ALMEIDA, 2009).

Temos em Soutelino (2006) definições de diversos autores para o sistema hub and spoke relacionados ao setor aeroviário.

Tabela 2 – Definições para o termo *Hub and Spoke* (SOUTELINO, 2006 adaptado).

Autor	Definição
Berry, Carnall & Spiller (1996), p.1	[No sistema *hub and spoke*], os passageiros mudam de avião no aeroporto hub no caminho para destino final.
Bootsma (1997), p.4	No sistema *hub and spoke*, a rede é desenhada de uma forma em que as rotas são concentradas em um número de facilidades nas conexões, chamadas hub. Destinos até o hub são chamados aros. [..]. Visando minimizar, as possíveis conexões, a companhia aérea operadora do hub, usualmente elaboram os horários dentro de um de-

	terminado tempo.
Button (1998), p. 20	No sistema *hub and spoke*, (…) as companhias aéreas geralmente usam um ou dois grandes aeroportos (…). Voos são planejados de maneira que permitem aos passageiros continuar a sua viagem através de conexões para mais destinos.
Burghouwt & Hakfoort (2001), p. 311	O sistema *hub and spoke* é uma combinação das ligações ponto a ponto com transferência de tráfego em um hub central.

Segundo Çiftçi (2015) um modelo *hub and spoke* possui vantagens significativas para as companhias aéreas como:

a) Consolidação do número de passageiros criando economia de escala;
b) Diminuição do número de rotas necessárias para conectar cada par origem/destino
c) Aumento da demanda por voos frequentes;
d) Concentração das atividades de pessoal, manutenção e operações;
e) Redução de custos.

No entanto a tarefa de encontrar a melhor localização de *hubs* e fazer a alocação dos nós aos *hubs* escolhidos possui uma natureza combinatória elevada, isto inclui o problema na classe de problemas NP-Difícil, e para que algoritmos exatos possam encontrar a solução ótima do problema precisaria de grande esforço computacional, e com os recursos computacionais disponíveis atualmente torna-se impraticável o uso de algoritmos exatos para instâncias de grandes dimensões (ALUMUR e KARA, 2008).

2.1.3. Modelo Matemático

O primeiro modelo matemático com função objetivo quadrático para o problema de localização não capacitado de *p*-hubs (*USApHMP*) foi elaborado por O'Kelly (1987). Mais tarde Campbell (1994) apresentou o primeiro modelo de Programação Linear Inteira, no entanto o modelo exigia um número muito grande de variáveis e restrições. Neste trabalho foi utilizado o modelo proposto em Ernst e Mohan (1996) que formularam o problema de localização de p-hubs como:

$$Min \sum_i \sum_k d_{ik} x_{ik} (\lambda E_i + \delta S_i) + \sum_i \sum_k \sum_l \beta d_{kl} y_{kl}^i \qquad (1.1)$$

$Sujeito\ a$:

$$\sum_k x_{kk} = p \quad (k = 1, \dots, n), n > p \qquad (1.2)$$

$$\sum_k x_{ik} = 1, \quad \forall i \in V \qquad (1.3)$$

$$y_{kl}^i \geq 0 \qquad \forall i, k, l \in V \qquad (1.4)$$

$$x_{ik} \in \{0,1\} \qquad \forall i, k \in V \qquad (1.5)$$

- No modelo apresentado à função objetivo é definida na Equação (1.1) que estabelece o custo total a ser minimizado referente à soma dos custos de coleta, transferência e distribuição de uma rede;
- A restrição descrita na Equação (1.2) garante que o número de *hubs* alocados seja igual a *p*;
- A restrição descrita na Equação (1.3) garante que um nó não *hub* somente poderá ser alocado a um único *hub*;
- A equação (1.4) determina que o fluxo entre *hubs* deve ser maior que zero;
- O domínio das variáveis é definido na Equação (1.5).
- n é $número\ de\ nós\ da\ rede$
- V é $o\ conjunto\ de\ nós\ da\ rede$
- p é $o\ número\ de\ hubs\ a\ serem\ localizados$
- d_{ij} é $a\ distância\ entre\ os\ nós\ i\ e\ j$
- w_{ij} é $a\ quantidade\ de\ fluxo\ que\ passa\ entre\ os\ nós\ i\ e\ j$
- $E_i = \sum_j w_{ij}\ representa\ o\ fluxo\ que\ chega\ ao\ nó\ i$
- $S_i = \sum_j w_{ji}\ representa\ o\ fluxo\ que\ sai\ do\ nó\ i$

- λ, β, δ *representam os custos de coleta, transferência e distribuição*
- y_{kl}^{i} *significa a quantidade de fluxo transferido entre os concentradores* k *e* l *originado a partir do nó* i
- $x_{ik} = 1$ *se o nó* i *está alocado ao concentrador* k ($x_{ik} = 0$ *caso contrário*)
- $x_{kk} = 1$ *o nó* k *é um concentrador caso contrário* $x_{kk} = 0$

O'Kelly (1992) apresentou um modelo para o problema de localização de hubs, no qual cada hub alocado possui um custo fixo associado. O problema foi formulado como:

$$Min \sum_{i}\sum_{k} d_{ik}x_{ik}(\lambda E_i + \delta S_i) + \sum_{i}\sum_{k}\sum_{l} \beta d_{kl} y_{kl}^{i} + \sum_{k} f_k x_{kk} \quad (1.6)$$

Sujeito a: (1.3), (1.4) e (1.5)

- A Equação (1.6) se diferencia da Equação (1.1) por acrescentar ao valor da função objetivo o custo fixo de alocação de cada *hub*, que no modelo é representado por f_k.

2.1.4. Revisão da Literatura

Segundo Alumur e Kara (2008), o primeiro trabalho sobre *Hub Location Problem* foi publicado em 1969 por Goldman, porém somente em 1987 O'Kelly apresentou a primeira formulação matemática para o *single allocation p-hub median problem*. Para resolução do *HLP* O'Kelly (1987) apresenta duas heurísticas. A primeira faz alocação de cada nó ao seu *hub* mais próximo, esta heurística ficou conhecida como *nearest hub allocation rule*. A segunda heurística verifica a alocação de todos os nós não *hub* para o primeiro ou segundo *hub* mais próximo do nó.

O'Kelly (1987) introduziu na literatura o uso da instância CAB, este conjunto de dados é baseado em informações de interações de passageiros de companhias aéreas entre 25 cidades dos EUA, posteriormente estas instâncias foram utilizadas por outros pesquisadores para validação de seus trabalhos.

Klincwicz (1992) utilizou as metaheurísticas Busca Tabu e *GRASP* para resolver o *p-Hub Location Problem*, ambas atingiram excelentes resultados, os algoritmos encontraram o ótimo, ou o melhor valor da função objetivo conhecido em 90,60% e em 93,75% dos casos, respectivamente. Como instâncias de teste o autor utilizou o conjunto de instâncias CAB, com n = [10, 15, 20 e 25] e p = 3 e 4.

Ernst e Mohan (1996) apresentaram uma nova formulação matemática para o *single allocation p-hub median problem*, o autor apresentou uma heurística baseada no *Simulated Annealing* (*SA*), onde conseguiu soluções exatas com tempo computacional razoável, e também apresentou resultados para um novo conjunto de instâncias, o AP *data set*, que consiste em 200 nós que representam distritos postais do correio Australiano.

Figueiredo (2005) apresentou um estudo de caso aplicado ao transporte aéreo de cargas no Brasil. Como instâncias para o problema foram utilizadas as informações do Departamento de Aviação Comercial (DAC) referente à movimentação de cargas entre as cinco regiões Brasileiras. O autor utilizou uma simplificação do modelo proposto por O'Kelly et al (1996) para identificar a alocação de uma única região para atuar como *hub*.

Cunha e Silva (2007) apresentaram um Algoritmo Genético (AG) para uma versão modificada do *Uncapacitated Single Allocation Hub Location Problem* (*USAHLP*), pois diferente da formulação original do problema, o fator de desconto da economia de escala para tráfego entre nós *hubs* não é constante e pode variar de acordo com a quantidade total de mercadorias transportadas. Para os testes computacionais foram utilizados o conjunto de instâncias CAB.

Siqueira (2008) detalhou os requisitos necessários para um aeroporto operar como *hub*. O trabalho apresentou informações dos principais aeroportos existentes no Brasil e no mundo, e após análise dos aeroportos Brasileiros caracterizados como *hub*, foram sugeridos quais outros aeroportos poderiam operar como *hub* no Brasil.

Almeida (2009) apresentou dois algoritmos combinados com a busca por agrupamentos (*clustering search*) para resolução do *HLP*. O autor utilizou um Algoritmo Genético e o *Simulated Annealing* com Lista Tabu (SATL). Os testes computacionais foram realizados com os algoritmos em suas versões originais e com os algoritmos propostos. Os resultados mostraram que, quando o AG e o SATL utilizados em conjunto com a busca por agrupamentos permite encontrar soluções de melhor qualidade em menor tempo computacional. Para os testes computacionais foram utilizados as instâncias CAB e AP.

Silva e Cunha (2009) propuseram três algoritmos baseados em Busca Tabu para resolução do USAHLP. Para validação dos dados os autores utilizaram as instâncias CAB e AP, os algoritmos propostos encontraram o valor ótimo conhecido para as instâncias testadas em um curto tempo de processamento. A eficácia dos algoritmos permitiu a resolução como de novas instâncias geradas a partir da AP, com 300 e 400 nós.

Cetiner et al (2010) consideraram o problema de roteamento em um sistema postal de entrega com o problema de localização de *hubs*. Os autores apresentaram um procedimento em dois estágios. No primeiro estágio é determinada a localizações dos *hubs* e feita as alocações dos centros de postagens. No segundo estágio rotas são criadas respeitando a estrutura de *hubs*. Em seus testes computacionais os autores utilizaram em um primeiro momento as instâncias já conhecidas da literatura. Posteriormente fez-se um estudo de caso com conjunto de dados do Serviço Postal da Turquia.

Costa et al (2010) forneceram um novo modelo matemático para calcular e identificar novos *hubs* no Brasil. Após uma discrepância encontrada entre os dados obtidos e o método utilizado pela *US Federal Aviation Administration – FAA* juntamente com o *Herfindahl–Hirschman Method*, foi criado um novo modelo para identificar um número de *hubs* dado uma rede.

Oktal e Asuman (2013) apresentaram um novo modelo para o *UMAHLP*. Um modelo misto de programação linear inteira foi desenvolvido acrescentado ao modelo tradicional restrições que são particulares do setor aéreo como: custos de viagem, disponibilidade de pistas e de carga, faixa de aeronaves. Os testes computacionais foram feitos com dados fornecidos por duas empresas aéreas de transporte de cargas da Turquia (*Turkish Air Cargo – TAC*).

Farahani et al (2013) apresentam um *review* sobre o *HLP*, onde os autores apresentaram alguns trabalhos mais recentes, entre os anos de 2008 à 2012. O trabalho apresenta cerca de 160 referências classificadas em termos de modelos, soluções e desempenho, onde podemos encontrar trabalhos abordando praticamente todas as variações do *HLP*.

Alzamora et al (2013) utilizaram o modelo q-Hub-Mediana para definir a localização de *hubs* para o transporte de passageiros no Brasil. O trabalho apresenta uma estratégia em duas fases, onde na primeira são identificados *hubs* regionais que são utilizados para interligar uma malha de 150 aeroportos e aeródromos que apresentaram movimentação de passageiros de acordo com as informações da ANAC em 2010. Nessa primeira fase foram feitos testes fixando a quantidade de *hubs* regionais em: 10, 19, 24, 34 e 40. Na segunda fase uma nova instância do problema é resolvida para encontrar os novos *hubs* interligando os *hubs* obtidos na primeira fase.

Bailey et al (2013) apresentaram o primeiro trabalho utilizando Otimização por Enxame de Partículas (*Particle Swarm Optimiztion - PSO*) para resolução do *USAHLP*. De acordo com os autores o método proposto conseguiu encontrar os melhores resultados conhecidos na literatura para o conjunto de instâncias CAB.

Adibi e Jafar (2015) forneceram algumas contribuições para literatura como: o primeiro modelo que formula *Uncapacitaded Multiple Allocation Hub Location Problem*, onde a incerteza de demanda e de custos de transportes é integrada ao modelo. A avaliação do modelo proposto é feita através de um estudo de caso aplicado a uma base de dados real referentes a uma rede com dados das 10 cidades do Iran com maior volume de transporte aéreo durante o ano de 2010.

An et al (2015) apresentaram modelos e algoritmos para uma rede confiável de hubs. Isto se dá pelo fato que rupturas podem ocorrer nos sistemas de hub ocasionado por diversos fatores: greves trabalhistas, condições climáticas ou ações terroristas. Segundo o autor a inclusão da escolha de *hubs* para atuarem como rotas alternativas aumentou drasticamente a complexidade do problema. Os testes computacionais foram realizados com a instância CAB e utilizados os métodos *Branch-and-Bound* e Relaxação Lagrangeana. D acordo com os autores os resultados gerados se mostraram como redes *hub and spoke* mais resilientes a rupturas.

Abyazi-Sani et al (2016) propuseram um algoritmo Busca Tabu para resolver o USAHLP, para diminuir o tempo computacional os autores utilizaram somente as estruturas de memória de curto e longo prazo, desprezando a estrutura de memória de médio prazo da Busca Tabu. Para validar o algoritmo proposto os autores compararam os resultados obtidos com os resultados apresentados por Silva et al (2009), que também propôs três algoritmos baseados em Busca Tabu. O algoritmo proposto encontrou as soluções ótimas com tempo computacional menor que os já conhecidos na literatura.

2.2. PROBLEMAS DE CORTE E ESTOQUE

O Problema de Corte e Estoque – PCE (do inglês *Cutting Stock Problem - CSP*), é o nome dado a uma classe geral de problemas, onde para os problemas de corte peças menores denominadas itens devem ser cortados a partir de peças de tamanho padrão denominadas objetos. Nos problemas de empacotamento os itens devem ser encaixados dentro dos objetos. Na Tabela 3 podemos encontrar diversas aplicações para PCE.

Tabela 3 – Aplicações para o PCE. (TEMPONI, 2007).

Problemas	Objetos	Itens	Dimensão
Alocação de Comerciais de TV	Tempo de cada intervalo	Tempo de cada propaganda	1D
Alocação de arquivos em mídias	Tamanho das mídias	Tamanho dos arquivos	1D
Empacotamento de caixas em galpões	Galpões	Caixas	3D
Empacotamento de cargas em containers	Containers	Cargas	3D
Corte de bobinas (papel, aço e tecido)	Bobinas	Peças menores	2D
Corte de vigas (madeira, aço)	Vigas	Peças menores	1D
Corte de placas (aço, vidro, madeira)	Placas	Peças menores	2D
Corte de espumas para colchões ou isopor	Espumas/isopor	Peças menores	3D

A tarefa de cortar: madeira, aço, vidro, papel, tecido ou esponja, faz parte do processo produtivo de diversas indústrias que utilizam estes itens como matéria prima. Geralmente armazenam em estoque peças de tamanho padrão (bobinas, chapas ou folhas), e devem produzir peças de tamanhos menores, que são cortadas de acordo com a demanda.

De acordo com Baldo (2006), o problema de corte consiste em encontrar a melhor maneira de cortar unidades maiores em estoque (objetos), para produzir unidades menores em quantidades previamente demandadas (itens), gerando assim um padrão de corte que possa maximizar o aproveitamento dos objetos utilizados.

2.2.1. Classificação dos Problemas de Corte e Estoque

Os Problemas de Corte e Estoque possuem diversas características tais como: dimensionalidade, medidas quantitativas, formato das peças, sortimento, disponibilidade, restrições de padrão, restrições de alocação, objetivos, estado da informação e variabilidade. Dyckhoff (1990) apresentou quatro critérios para classificação, destacando a dimensionalidade como característica mais importante, conforme pode ser visto na Tabela 4:

Tabela 4 – Classificação do PCE segundo Dyckhoff (1990).

1-Dimensionalidade	2-Tipo de Alocação
(1) Unidimensional (2) Bidimensional (3) Tridimensional (4) N-dimensional, com N > 3	(B) Todos os objetos e uma seleção de itens (V) Uma seleção de objetos e todos os itens
3-Sortimento de Objetos	4-Sortimento de itens
(O) Um objeto (I) Objetos de formatos idênticos (D) Objetos de formatos distintos	(F) Poucos itens de diferentes formatos (M) Muitos itens de muitos formatos distintos (R) Muitos itens de formatos distintos em relativa quantidade (C) Itens de formatos congruentes

Baseada nas ideias iniciais de Dyckhoff, Wäsher et al (2006) propuseram uma nova tipologia que proporciona uma completa categorização dos problemas de corte e empacotamento, que pode ser visualizada na Tabela 5.

Tabela 5 – Classificação do PCE segundo Wäsher et al (2006).

1-Dimensionalidade	2-Tipo de Atribuição
Unidimensional Bidimensional Tridimensional	Maximizar a saída Minimizar a entrada
3-Tipos dos itens	4-Tipo de objetos
Idênticos Pouco heterogêneos Muito heterogêneos	Um único objeto: com todas as dimensões fixas ou com uma ou mais dimensões variáveis. Muitos Objetos: idênticos, pouco heterogêneos, muito heterogêneos.
5-Forma dos itens	
Regulares **Irregulares**	

Como visto nas Tabelas 4 e 5, considera-se a dimensionalidade a característica mais importante para classificação dos problemas de corte e estoque. Quanto à dimensionalidade temos:

Unidimensional: apenas uma dimensão é relevante durante o processo de corte, este tipo de problema ocorre durante o processo de corte de barras de aço, bobinas de papel, tubos para produção de treliça etc. A Figura 3 apresenta um exemplo para o problema unidimensional.

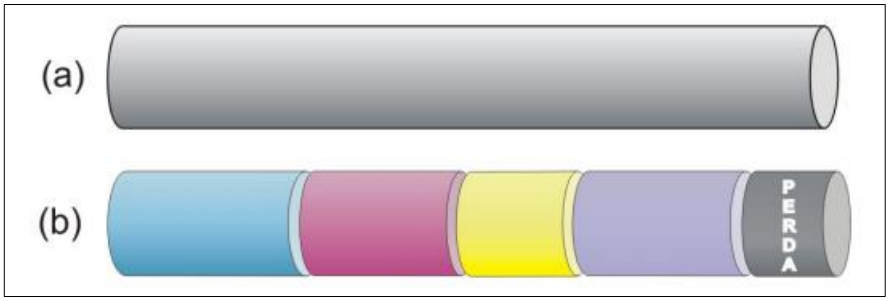


Figura 3 – (a) Objeto a ser cortado (b) Objeto cortado (CHERRY, 2006).

Bidimensional: como pode ser visto na Figura 4, duas dimensões (comprimento e largura) são relevantes durante o processo de corte, e são frequentes em indústrias de móveis, fabricação de espelhos e placas de metal.

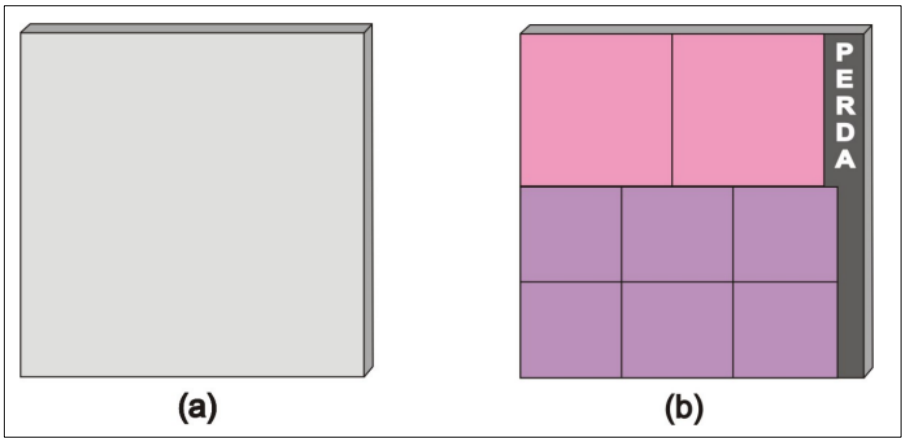


Figura 4 – (a) Objeto a ser cortado (b) Objeto cortado (CHERRY, 2006).

Tridimensional: três dimensões (altura, comprimento e largura) são relevantes durante o processo de corte, e são frequentes encontrados em indústrias fabricação de colchões, e também

no processo de carregamento de containers e caminhões. A Figura 5 apresenta um exemplo para o problema tridimensional.

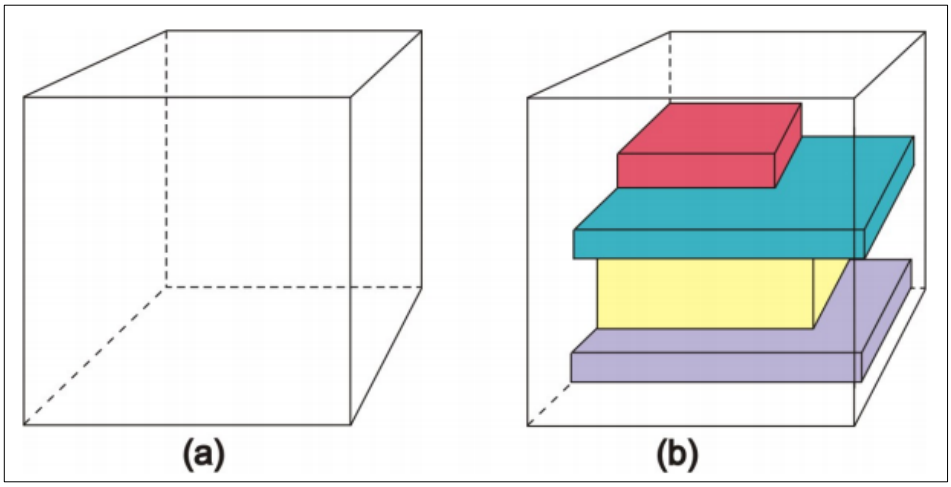


Figura 5 – (a) Container (b) Caixas empacotadas (CHERRY, 2006).

N-Dimensional: onde N > 3, mais de três dimensões são relevantes para solução do problema, um exemplo para esse tipo de problema seria alocação de tarefas num dia de trabalho as quais utilizam diferentes recursos renováveis. (CHERRY, 2008 apud MORABITO, 1992).

2.2.2. Problema de Corte Bidimensional Guilhotinado e Restrito

O Problema de Corte Bidimensional pode ser definido como dado um objeto de tamanho padrão em estoque $A = (L, W)$ onde L representa o comprimento e W a largura do objeto, e um conjunto $R=\{(l_1,w_1),(l_2,w_2),...,(l_m,w_m)\}$ de peças menores que devem ser cortadas, onde l_i é o comprimento da peça i e w_i a largura da peça i e o valor de utilidade v_i, $i = 1,...,m$.

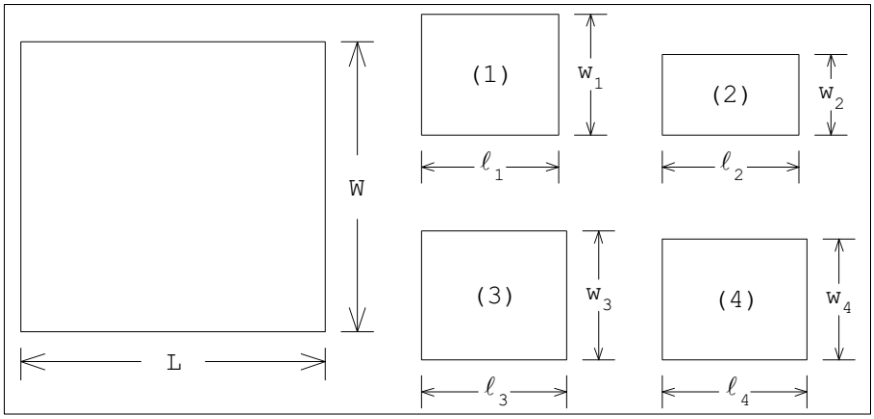


Figura 6 – Instância para o Problema de Corte Estoque (VIANA; POLDI, 2005).

A Figura 6 apresenta o que poderia ser uma instância para o PCE, pois temos um objeto maior com quatro itens menores que devem ser cortados a partir do objeto padrão. Quando o padrão de corte é do tipo guilhotinado, significa que a cada corte efetuado no objeto este irá gerar sempre dois outros retângulos. E em alguns casos são necessários mais um corte para fazer a apara do resíduo das peças. O caso guilhotinado é encontrado com frequência no processo de corte de papel, madeira, vidro e chapas de metal, por exemplo.

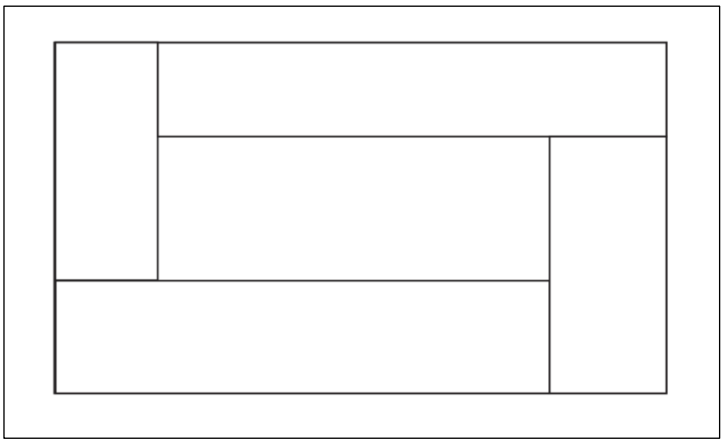


Figura 7 – Padrão inviável para o corte guilhotinado (CHRISTOFIDES; WHITLOCK, 1976).

Na Figura 7 é mostrado um padrão inviável de ser produzido com o corte guilhotinado, na Figura 8 é mostrado um padrão típico para o corte guilhotinado.

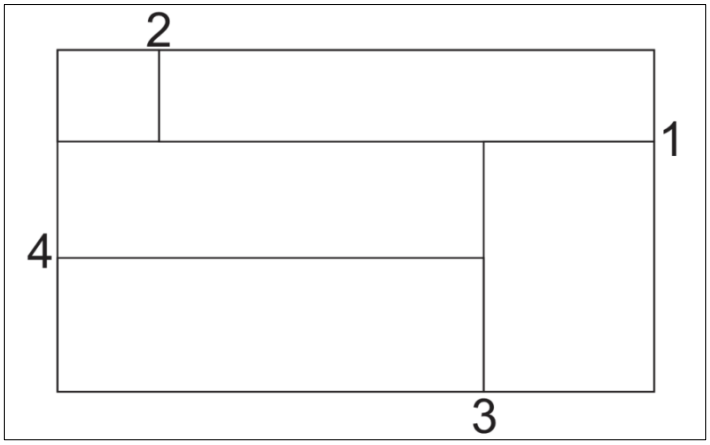


Figura 8 – Padrão típico do corte guilhotinado (CHRISTOFIDES; WHITLOCK, 1976).

Quando o problema não possui restrições em relação à quantidade de peças que devem ser produzidas dizemos que é do tipo irrestrito. Porém, quando se deve produzir a quantidade de peças de acordo com uma demanda previamente estabelecida não permitindo o excesso de produção, sendo um vetor d_i, $i = 1,\ldots,m$ de valores correspondentes as quantidade de cada item *R* que pode ser cortada em *A*, o problema passa a ser restrito.

2.2.3. Modelo matemático

Para definição do modelo matemático para o Problema de Corte Bidimensional Restrito em que devemos encontrar o melhor padrão de corte de maneira que maximize o valor total das peças produzidas. Conforme apresentado por Yanasse e Morabito (2006), podemos considerar:

- $m = número\ de\ itens$;
- $L, W = comprimento\ e\ largura\ do\ objeto\ padrão\ respectivamente$;
- $l_i = comprimento\ do\ item\ i; i = 1, \ldots, m$;
- $w_i = largura\ do\ item\ i; i = 1, \ldots, m$;
- $d_i = demanda\ do\ item\ i; i = 1, \ldots, m$;
- $v_{ijk} = valor\ do\ item\ i; i = 1, \ldots, m\ incluído\ em\ um\ retângulo\ de\ comprimento\ j,$

  $j = 1, \ldots, J\ e\ largura\ k, k = 1, .., K$;
- $J = a\ quantidade\ de\ comprimentos\ diferentes\ l_i, i = 1, \ldots, m$;
- $K = a\ quantidade\ de\ larguras\ diferentes\ w_i, i = 1, \ldots, m$;
- $M = número\ suficientemente\ grande$;
- $\lambda_j = número\ de\ vezes\ que\ o\ comprimento\ l_j (j = 1, \ldots, J)$ é $cortado\ ao\ longo\ de\ L$;
- $\mu_k = número\ de\ vezes\ que\ a\ largura\ w_k (k = 1, \ldots, K)$ é $cortada\ ao\ longo\ de\ W$;

$a_{ijk} = número\ de\ retângulos\ (l_j, w_k)\ que\ contém\ itens\ do\ tipo\ i;$

$modelo$

$$Max \sum_{i=1}^{m}\sum_{j=1}^{J}\sum_{k=1}^{K} v_{ijk}a_{ijk} \qquad (2.1)$$

$sujeito\ a$:

$$\sum_{j=1}^{J} l_j\,\lambda_j \leq L \qquad (2.2)$$

$$\sum_{k=1}^{K} w_k\,\mu_k \leq W \qquad (2.3)$$

$$\sum_{i=1}^{m} a_{ijk} \leq \lambda_j\mu_k, \forall j = 1, \dots, J; k = 1, \dots, K \qquad (2.4)$$

$$\sum_{j=1}^{J}\sum_{k=1}^{K} a_{ijk} \leq\ d_i, , \forall i = 1, \dots, m \qquad (2.5)$$

$$\lambda_k, \mu_k, a_{ijk} \in Z_+;\ i = 1, \dots, m;\ i = j, \dots, J;\ i = k, \dots, K. \qquad (2.6)$$

- A Equação (2.1) apresenta a função objetivo que maximiza o valor total dos itens cortados no padrão;
- As Equações (2.2) e (2.3) impõem que o comprimento e a largura dos retângulos não excedam o comprimento e largura do objeto, respectivamente;
- Na Equação (2.4) o termo $\lambda_j\mu_k$ corresponde ao número de retângulos (l_j, w_k) no padrão. Está restrição limita o total de itens de todos os tipos cortados no padrão;
- Na Equação (2.5) garante um padrão de corte restrito, isto é, limita pela demanda a quantidade de itens do tipo *i* produzidos no objeto;
- Por último a Equação (2.6) diz respeito a não negatividade e integralidade das variáveis.
- O valor de cada item é calculado por:

$$v_{ijk} = \begin{cases} v_i, & se\ l_i \leq l_j\ e\ w_i \leq\ w_k; \\ 0, & caso\ contrário. \end{cases}$$

2.2.4. Revisão da Literatura

Os primeiros trabalhos sobre problemas de corte foram apresentados por Gilmore e Gomory (1961), que utilizaram o método de geração de colunas para obter a solução ótima para o problema de corte.

Posteriormente diversos outros trabalhos foram publicados sobre o referido problema e destacamos aqui o trabalho de Christofides e Withlock (1977) que propuseram um algoritmo de busca em árvore para resolver o problema de corte bidimensional guilhotinado restrito (quando a quantidade produzida não pode exceder uma quantidade previamente estabelecida) e irrestrito (quando não há limite na produção de itens), para os testes computacionais os autores utilizaram três instâncias criadas aleatoriamente, estas instâncias foram utilizadas posteriormente por diversos outros autores para validações de métodos propostos.

Em Wang (1983) foram apresentados dois novos algoritmos para resolução do problema de corte bidimensional guilhotinado restrito. No algoritmo proposto são adicionados itens sucessivamente criando assim diversas soluções parciais. Para evitar que um número excessivo de soluções parciais fossem criadas, foi definido um parâmetro de satisfabilidade que as soluções devem atender. O algoritmo rejeita soluções parciais que excedam um percentual de perda. No mesmo trabalho, o autor utilizou um conjunto de instâncias derivadas das instâncias utilizadas em Christofides e Withlock (1977), e que serviram para validação de outros trabalhos propostos.

Vasko (1989) propôs dois algoritmos para a resolução do problema de corte bidimensional guilhotinado e restrito em dois estágios, o algoritmo Módulo de Aplicação de Placa Excedente *(*do inglês *Surplus Plate Application Module - SPAM).* O algoritmo apresentou um alto rendimento na geração de padrões que exigem poucos cortes, apesar de não ser um método exato o *SPAM* frequentemente encontra o valor ótimo para o problema em questão.

Dyckhoff (1990) introduziu na literatura uma tipologia de classificação para os problemas de corte e empacotamento. O autor apresentou diversas características que são levadas em consideração no processo de classificação dos problemas. Destacou como a característica mais importante a dimensão, classificando-as assim como unidimensional, bidimensional, tridimensional e multidimensional ou n-dimensional.

Oliveira e Ferreira (1990) criaram uma versão modificada do algoritmo de Wang (1983). Em sua proposta os autores fizeram uma modificação para que o algoritmo não utilize um percentual fixo de perda para rejeitar a solução parcial. O algoritmo proposto utiliza como

parâmetro o percentual de perda da melhor solução encontrada até o momento. Os autores criaram instâncias que são utilizadas para validação de outros trabalhos.

Morabito e Arenales (1995) apresentaram resultados computacionais para as heurísticas propostas em Beasley (1985) e Gilmore e Gomore (1965) para a resolução de instâncias grandes do problema de corte bidimensional guilhotinado irrestrito e não estagiado. A segunda heurística apresentada em Morabito e Arenales (1995) gerou melhores resultados para maioria das instâncias testadas. Na maior instância com L = 5000 e W = 5000 e m = 100 itens diferentes, a primeira heurística não conseguiu encontrar a solução e a segunda encontrou com tempo computacional abaixo de 4 segundos.

Hopper e Turton (2001) utilizaram metaheurísticas híbridas para resolução de problemas de corte bidimensional não guilhotinado. A partir da técnica de encaixe denominada *Bottom Left -BL* (em que os itens são alocados no objeto inicialmente no canto superior direito, e depois deslocada para baixo e para esquerda enquanto possível) juntamente com as metaheurísticas: Algoritmos Genéticos (do inglês *Genetic Algorithms - GA*), *Simulated Annealing – SA, Naive Evolution – NE* e uma heurística de busca local *Hill-Climbing*. Para validação do método os autores utilizaram um conjunto de sete instâncias com a quantidade de itens variando entre 16 e 197, para cada grupo foram criadas três instâncias com o valor ótimo conhecido.

Wäsher et al (2006) propuseram uma melhoria na tipologia para classificação dos problemas de corte e empacotamento inicialmente proposta por Dyckholf (1990). Com as melhorias propostas foi capaz de categorizar todos os problemas de corte e empacotamento conhecidos na literatura. Com o passar dos anos e com a evolução da pesquisa na área dos problemas de corte e empacotamento o método de classificação inicial se tornou ineficaz.

Temponi (2007) propôs três metaheurísticas: *Iterated Local Search (ILS), Greedy Randomized Adaptive Search Procedure (GRASP)* e uma metaheurística híbrida, que utiliza a metaheurística *GRASP* como geradora da solução inicial da metaheurística *ILS*. Para validação das metaheurísticas propostas foram utilizadas instâncias de grande e pequeno porte utilizadas em Hopper e Turton (2001). De acordo com o autor, a metaheurística híbrida *ILS-GRASP* foi a que obteve melhores resultados em termos da função de avaliação. Nos casos que a solução ótima era conhecida a *ILS-GRASP* conseguiu obtê-la.

Velasco (2005) criou uma heurística baseada na metaheurística *GRASP* para aplicação ao problema de corte e empacotamento guilhotinado e restrito. A heurística proposta pelo autor, assim como na *GRASP*, possui a seleção dos itens a partir de uma Lista Restrita de

Candidatos (LRC) e um parâmetro de controle de intensidade da gula (ou miopia) da escolha dos itens a compor a solução.

O que diferencia a metaheurística proposta em Velasco (2005) e a *GRASP* original é o fato de que na heurística proposta, a fase de melhoria não é aplicada a uma solução completa e sim a uma solução parcial. Para validação da heurística o autor utilizou instâncias da literatura onde o valor ótimo é conhecido, mais detalhes sobre a heurística serão mostrados na seção 3.3.

Silva et al (2011) utilizaram um algoritmo baseado na fase de construção da *GRASP* para resolução de uma instância real do problema de corte bidimensional guilhotinado e restrito aplicado a um estudo de caso em uma indústria de carrocerias. O autor obteve um bom resultado quando comparado o padrão de corte obtido pelo método proposto e o padrão de corte utilizado pelo setor produtivo da empresa em questão.

Kalrath et al (2014) apresentaram duas contribuições para literatura, à primeira foi um método para resolver o problema de corte unidimensional com dois critérios para minimização sendo a redução do número de rolos utilizados e redução do número de padrões, e a segunda contribuição são conhecimentos de gestão para indústria de papel. Segundo os autores foram apresentados aspectos relevantes para a indústria do papel, aspectos esses com pouco tratamento na literatura científica.

Dusberger e Raidl (2015) utilizaram programação dinâmica (do inglês Dynamic Programming - *DP*) e a metaheurística *Variable Neighboard Search (VNS)* para resolver o problema de corte bidimensional guilhotinado de três estágios. Para validação do método proposto os autores utilizaram um conjunto de dez instâncias da literatura. Os resultados apresentados a *VNS+DP* não obteve vantagem significativa sobre a *VNS* tradicional. Em algumas instâncias obteve valor inferior para a função objetivo e em termos de tempo computacional obteve um melhor resultado somente para uma das cinquenta instâncias testadas.

CAPÍTULO 3

METAHEURÍSTICAS

De acordo com Temponi (2007) os principais métodos para resolução de problemas de otimização combinatória são os métodos exatos, métodos aproximativos e os métodos heurísticos ou Metaheurísticas.

Os métodos exatos procuram obter a solução ótima para o problema a partir da construção de modelos matemáticos de otimização e da implementação de algoritmos específicos para sua resolução (TEMPONI, 2007). Os métodos exatos produzem soluções ótimas, porém, para problemas da classe NP-difíceis, possuem complexidade de tempo pelo menos exponencial, sendo então aplicáveis apenas para problemas de pequeno e médio porte (TEODORO, 2003).

Com a impossibilidade de se utilizar métodos exatos, métodos aproximativos são normalmente usados. Johnson (1974) define os métodos aproximativos como algoritmos polinomiais que buscam sacrificar o mínimo possível da qualidade, obtida nos métodos exatos, ganhando, simultaneamente, o máximo possível em eficiência (tempo polinomial). Assim os métodos aproximativos buscam o equilíbrio entre a qualidade da solução obtida em relação ao tempo computacional utilizado.

Os métodos heurísticos encontram soluções em um tempo computacional viável, mas sem garantia de que a solução encontrada seja ótima, os métodos heurísticos tendem a ficar presos em ótimos locais. Já as metaheurísticas fazem buscas de soluções no espaço de vizinhança tentando escapar de ótimos locais no intuito de encontrar um ótimo global (TEMPONI, 2007).

De acordo com Ribeiro (1996), metaheurística é um procedimento destinado a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico.

Encontramos também em Velasco (2005) a seguinte definição:

> Uma heurística é uma técnica que busca alcançar uma boa solução utilizando um esforço computacional considerado razoável, sendo capaz de garantir a viabilidade ou a otimalidade da solução encontrada ou, ainda, em muitos casos, ambas, especialmente nas ocasiões em que essa busca partir de uma so-

lução viável próximo ao ótimo (VELASCO, 2005 *apud* GOLDBARG e LUNA, 2000).

As próximas seções trazem o detalhamento das metaheurísticas *GRASP*, *GRASP* Reativo e *GRASP-2d*.

3.1. METAHEURÍSTICA GRASP

A metaheurística *GRASP* é composta por duas fases distintas, uma fase de construção da solução inicial e outra de busca local, que a partir da solução inicial, tenta-se encontrar soluções melhores através de buscas no espaço de vizinhança. Essa busca persiste até que um critério de parada seja alcançado, o que na maioria das vezes é um número máximo de iterações (*Nmax*).

```
1: procedimento GRASP(D, Nmax, α)
2:     f(S*) ← +∞; iter ← 1
3:     enquanto iter ≤ Nmax faça
4:         S1 ← FaseConstrutiva(D, α)
5:         S2 ← BuscaLocal(D, S1)
6:         se f(S2) < f(S*) então
7:             S* ← S2
8:         fim se
9:         iter ← iter + 1
10:    fim enquanto
11:    retorne S*
12: fim procedimento
```

Figura 9 – Pseudocódigo do *GRASP* (LIMA JÚNIOR, 2009) adaptado.

Na Figura 9, é possível visualizar o algoritmo padrão da *GRASP*, onde podemos observar na linha 4, que uma solução é encontrada na primeira fase, e depois essa solução é passada como entrada para a fase de busca local. Observa-se que o algoritmo armazena a melhor solução encontrada e não faz uso de informações de iterações passadas.

3.1.1. Fase de construção

Durante a fase de construção o algoritmo constrói uma solução inserindo um elemento de cada vez até que uma solução esteja completa. A escolha dos elementos é feita de acordo com um critério guloso aleatório que sofre influência do parâmetro α que tem seu valor variando entre [0,1]. A solução inicial é formada a partir de uma lista de candidatos (LC) criando-se uma lista restrita de candidatos (LRC), onde a quantidade de elementos que compõem a LRC é formada de acordo com a seguinte equação: $LRC = 1 + \propto (N - 1)$, onde N é o número de todos os elementos candidatos a compor a solução. Quanto mais próximo de 1 o valor de α estiver mais aleatório será a escolha dos elementos, e quanto mais próximo de 0 mais guloso será a estratégia de escolha do próximo elemento a compor a solução (LIMA JÚNIOR, 2009).

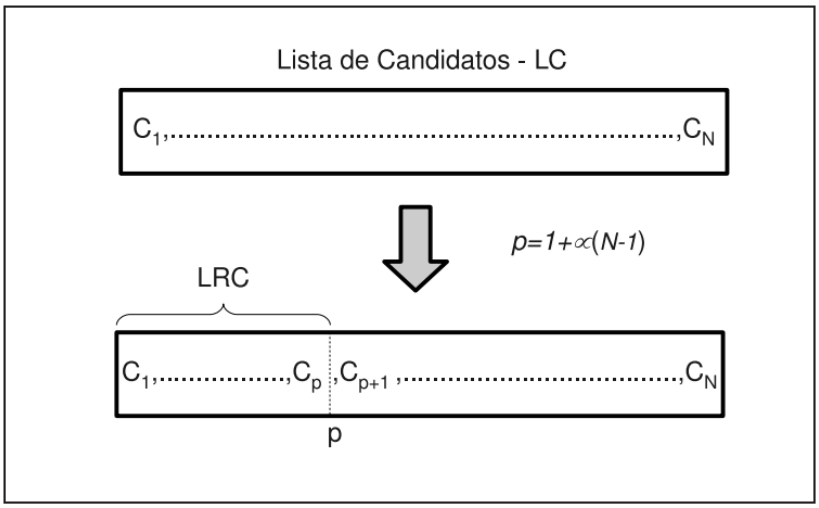


Figura 10 – Formação da LRC (LIMA JÚNIOR, 2009).

Como pode se observar na Figura 10, o valor de α influencia diretamente na escolha dos candidatos a compor a solução. A função guloso-aleatória é adaptada de acordo com a solução parcial que está sendo construída, pois a lista de elementos candidatos a compor a solução é ordenada de acordo com o benefício de cada candidato para a solução corrente. Estratégias completamente aleatórias (com α = 1) geram soluções de baixa qualidade, mas com grande diversidade. Soluções de baixa qualidade podem influenciar no processo de busca local tornando-o mais lento. Porém uma solução de baixa qualidade pode vir a se tornar uma solução de melhor qualidade após o processo de busca local.

3.1.2. Fase de busca local

A fase de busca local consiste em tentar encontrar uma solução *S'* melhor que a solução inicial *S*, realizando modificações em *S*. As modificações realizadas em *S* são chamadas de busca em vizinhança. A cada melhora encontrada na solução *S'* substitui-se o valor da função objetivo de *S* pelo valor da função objetivo de *S'*. Esse procedimento se repete até que um critério de parada seja atingido, retornando assim o melhor valor encontrado na fase de busca local.

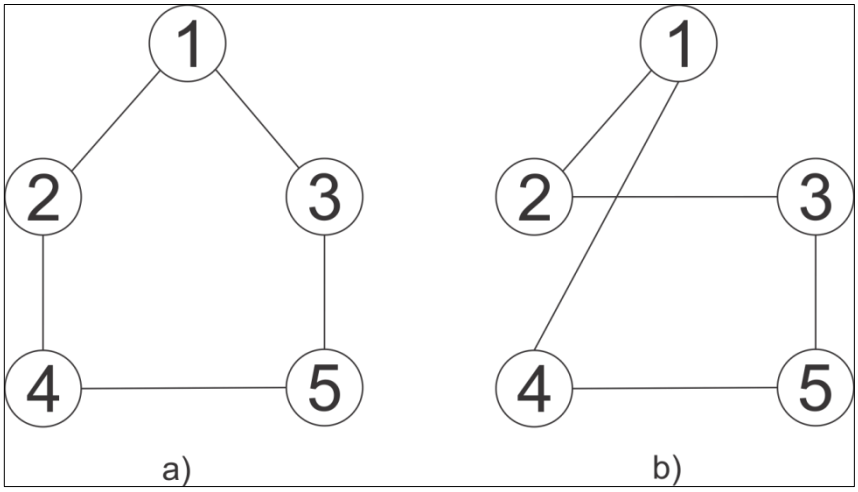


Figura 11 – Solução inicial S e uma solução S'.

Observa-se na Figura 11.a uma solução S, e na Figura 11.b o caminho de 1 a 3 foi desconectado e feito à conexão de 2 a 3. Criando assim uma nova solução S', caso o valor de S' seja melhor que o valor de S, a solução S recebe o valor de S' e o processo se repete até que um critério de parada seja atingido.

3.2. METAHEURÍSTICA *GRASP* REATIVO

Segundo Prais e Ribeiro (2000), o uso do valor fixo de α poderia impedir a construção de soluções de melhor qualidade que poderiam ser obtidas com valores diversificados de α. Ou seja, se α utilizado com valores muito próximos de uma escolha gulosa não geraria soluções diversificadas o bastante para permitir que soluções ótimas sejam encontradas. Dessa maneira propuseram uma modificação no critério de escolha do α a ser utilizado na fase de construção da solução inicial. Na *GRASP* tradicional define-se um valor para o α e todas as iterações são executadas com o valor fixado. Na *GRASP Reativo* (GR), este valor é escolhido

de maneira probabilística dentre um conjunto discreto $\psi = \{\alpha_1,\alpha_2,..,\alpha_m\}$ de α pré-estabelecidos.

Inicialmente todos os valores de α possuem a mesma probabilidade de serem escolhidos. Após um número *y* de execuções os α com soluções cujo valor da função objetivo esteja mais distante do melhor valor já encontrado recebem uma punição, e os *α* com melhor desempenho passam a ter mais chances de serem escolhidos nas próximas iterações (PRAIS; RIBEIRO, 2000).

```
1: procedimento GRASP_REATIVO(D, Nmax)
2:     f(S*) ← +∞; iter ← 1
3:     Defina A = {α_1, α_2, ..., α_v}
4:     para k ← 1 ate v faça
5:         count[k] ← 0; score[k] ← 0; avg[k] ← 0; p_k ← 1/v
6:     fim para
7:     enquanto iter ≤ Nmax faça
8:         α ← Selecione α_k ∈ A com probabilidade de escolha em p_k
9:         S_1 ← FaseConstrutiva(D, α)
10:        S_2 ← BuscaLocal(D, S_1)
11:        se f(S_2) < f(S*) então
12:            S* ← S_2
13:        fim se
14:        count[k] ← count[k] + 1; score[k] ← score[k] + f(S_2)
15:        se iter mod y = 0 então
16:            avg[k] ← score[k]/count[k] para todo k ∈ {1,2, ..., v}
17:            σ ← Σ(fmin/avg[k])^θ para todo k ∈ {1,2, ..., v}
18:            p_k ← (fmin/avg[k])^θ/σ para todo k ∈ {1,2, ..., v}
19:        fim se
20:        iter ← iter + 1
21:    fim enquanto
22:    retorne S*
23: fim procedimento
```

Figura 12 – Pseudocódigo GRASP Reativo (PRAIS; RIBEIRO, 2000) adaptado.

Como pode ser visto na Figura 12, na linha 3 um conjunto A com valores predefinidos de α é inicializado. Nas linhas de 4 a 6 todos os α recebem inicialmente o mesmo valor e possuem a mesma probabilidade de serem escolhidos. A cada *y* iterações essas probabilidades são atualizadas (linhas 15 a 19). Os α com melhor desempenho têm uma maior probabilidade de serem escolhidos em iterações futuras.

3.3. ALGORITMO *GRASP*-2D

O algoritmo *GRASP-2d* foi proposto por Velasco (2005), e tem como base a metaheurística *GRASP*. O *GRASP-2d* semelhantemente a *GRASP* possui duas fases: uma fase de construção e uma fase de busca local. Na fase de construção a solução é construída inserindo-se um elemento de cada vez, com os mesmos sendo selecionados a partir de uma lista com todos os itens candidatos a compor a solução. De acordo com o valor do parâmetro α e obedecendo a equação $LRC = 1 + \alpha(\mathrm{N} - 1)$ é formada uma *LRC* com os melhores candidatos a compor a solução, porém sem garantia que o melhor candidato seja escolhido.

Porém diferente da *GRASP* tradicional, em que a fase de melhoria inicia após uma solução completa ser criada, e fazendo busca no espaço de vizinhança, no *GRASP-2d*, a fase de melhoria atua logo após a inserção da(s) primeira(s) peça(s) tentando melhorar cada faixa de corte criada. A Figura 13 nos mostra o pseudocódigo do *GRASP-2d.*

```
1: procedimento GRASP2d(p_i, v(p_i), Nmax, α)
2:     S* ← ∅; iter ← 1
3:     enquanto iter ≤ Nmax faça
4:         Defina C = {p_i, ..., p_m}
5:         padcorte ← ∅; melhor ← ∅; Rf ← R
6:         equanto |C| ≠ 0 faça
7:             Defina LRC = {1 + α(n − 1) ∈ C}
8:             Selecione aleatoriamente um elemento de LRC
9:             Defina Fh e Fv
10:            Melhore Fh e Fv
11:            melhor ← max(Fh(v(p_i)), Fv(v(p_i)))
12:            padcorte ← padcorte ∪ {melhor}
13:            Atualiza C e Rf ← Rf − f(melhor)
14:        fim enquanto
15:        se f(padcorte) > f(S*) então
16:            S* ← padcorte
17:        fim se
18:        iter ← iter + 1
19:    fim enquanto
20:    retorne (S*, f(S*))
21: fim procedimento
```

Figura 13 – Pseudocódigo *GRASP*-2d (VELASCO, 2005) adaptado.

Como pode ser visto no pseudocódigo na Figura 13, inicialmente o retângulo R_f tem o tamanho do objeto, e a cada faixa criada este tem seu tamanho reduzido (linha 13). A fase de melhoria do algoritmo ocorre logo após a criação das faixas horizontais e verticais (linha 10). Nas linhas 11 e 12, a faixa que contém o maior valor de peças alocadas é armazenada na solução corrente. Nas linhas de 15 a 17 é armazenada a melhor solução encontrada até o momento. Ao final do algoritmo o melhor padrão de corte, juntamente com valor da função objetivo e são retornados (linha 20).

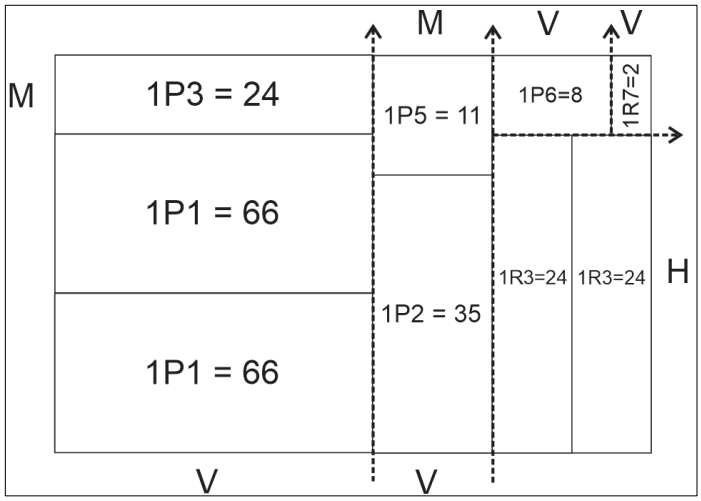


Figura 14 – Representação gráfica de um padrão de corte.

Para representar uma solução para o problema de corte e empacotamento foi definido uma *string* da seguinte maneira: V1P11P1M1P3V1P2M1P5H1R31R3V1P6V1R7, em que a letra V indica que a faixa vai ser criada na vertical, 1P1 indica que a peça 1 deve ser alocada uma vez, a letra M indica que a alocação inicial da faixa gerou uma espaço que pode ser preenchido, e M1P3 indica que a peça 3 vai ocupar o espaço restante, a presença da letra R indica que a peça deve ser rotacionada em 90º. A Figura 14 apresenta uma representação gráfica para uma instância do PCB.

3.3.1. Fase de construção do algoritmo *GRASP-2d*

Durante a fase de construção do algoritmo uma peça p_i é selecionada aleatoriamente a partir da LRC, a peça selecionada é alocada no canto inferior esquerdo do objeto padrão a ser cortado respeitando as restrições de largura e comprimento, dessa maneira são criadas duas faixas de corte: Faixa Vertical (FV) e Faixa Horizontal (FH), conforme pode ser visto na Figura 15.a.

Após o encaixe da primeira peça e a criação das faixas verticais e horizontais, o próximo passe é alocar a mesma peça quantas vezes forem possíveis, respeitando a restrição de quantidade de peças a serem produzidas e do comprimento e largura da faixa que está sendo construída.

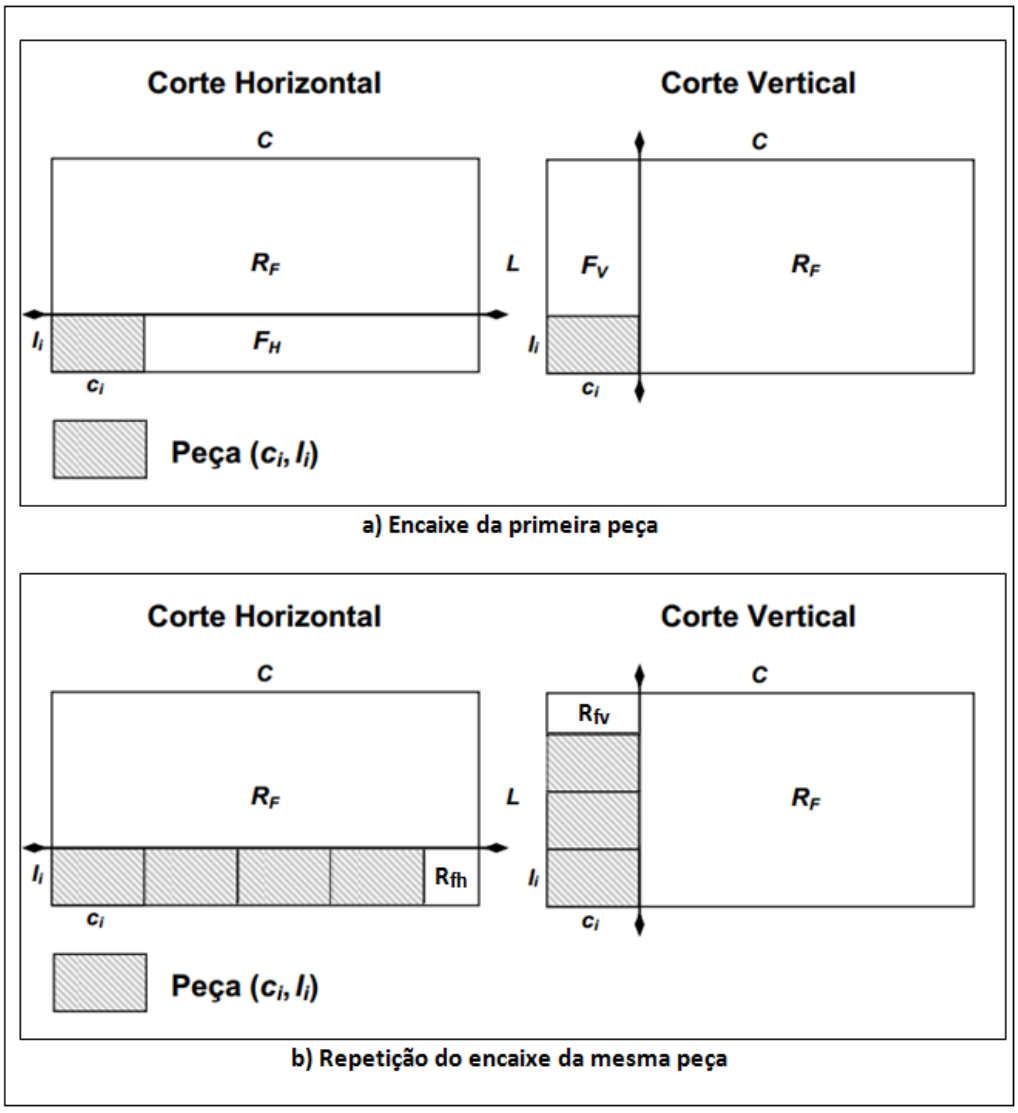


Figura 15 – Sequência da fase de construção. Velasco (2005) adaptado.

Pode-se observar na Figura 15.b que as faixas FH e FV possuem um espaço não utilizado chamado de R_{fh} (resto da faixa horizontal) e R_{fv} (resto da faixa vertical), que ainda podem ter sua utilização melhorada durante a fase de busca local, e na mesma Figura é possível observar o retângulo R_f que deverá ser utilizado para compor as demais faixas do padrão de corte.

3.3.2. Fase de melhoria do algoritmo *GRASP-2d*

Após a criação das faixas *FH* e *FV* verificam-se os retângulos R_{fh} e $R_{fv,}$ caso eles ainda possam alocar algum item é feito então uma lista de itens que ainda possuem demanda a ser atendida e que podem ser encaixados nos retângulos R_{fh} e R_{fv}. Após a criação da lista os itens são ordenados de forma decrescente de acordo com o valor de sua peça V_i, e o item de maior valor é encaixado no espaço restante, este processo se repete até que nenhum item possa mais ser encaixado nas faixas *FH* e *FV*.

Após o término de criação das faixas *FH* e *FV* verificam-se qual das duas faixas possui o maior valor referente ao somatório dos valores das peças alocadas, a faixa com maior valor fará parte do padrão de corte e a outra faixa é descartada.

Terminado a construção da primeira faixa todo o processo desde a fase de construção é repetido com o retângulo R_f até que nenhum outro item possa mais ser encaixado no objeto padrão.

CAPÍTULO 4

APRENDIZAGEM POR REFORÇO

Segundo Almeida (2014), Aprendizagem por Reforço – AR (do Inglês *Reinforcement Learning – RL*) é um paradigma da Inteligência Artificial com aplicações de sucesso em vários problemas de otimização. De acordo com Lima Júnior (2009), AR é um paradigma computacional onde um agente aprendiz procura maximizar uma medida de desempenho baseado em reforços que recebe por interagir com o ambiente.

Três fatores estão presentes no processo de aprendizagem: o conhecimento do estado do agente no ambiente, as ações efetuadas e as mudanças de estado decorrentes das ações (ALMEIDA, 2014). Segundo Sutton e Barto (1998), AR não é definido por caracterizar um método de aprendizagem, e sim por caracterizar um problema de aprendizagem. Qualquer método que seja bem adequado a esse problema é considerado um método de AR. A Figura 16 apresenta o ciclo do aprendizado por reforço.

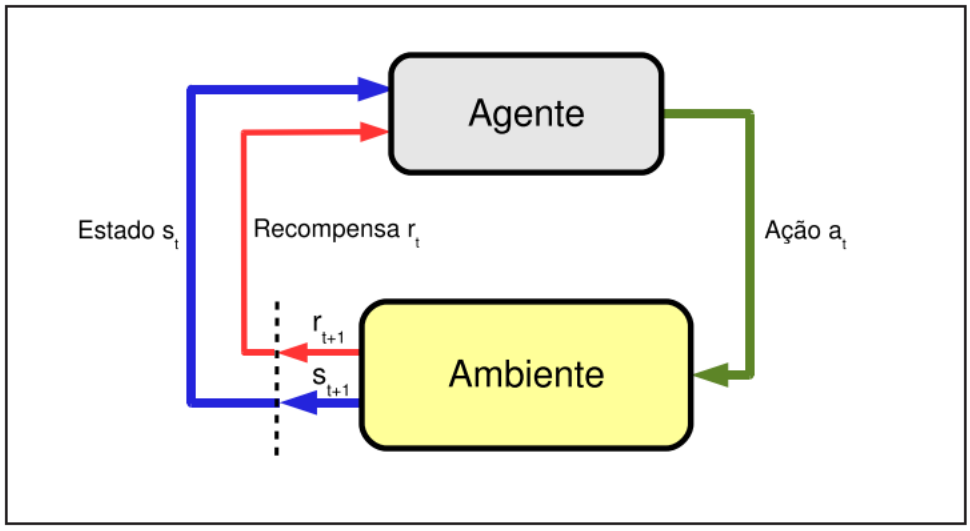


Figura 16 – Ciclo do aprendizado por reforço (LIMA JÚNIOR, 2009).

Conforme a Figura 16, no espaço de tempo t, o agente interage com o ambiente recebendo uma indicação do estado(s) atual, o agente escolhe, então, uma ação a tomar, e gera sua saída. A ação altera então o estado do ambiente, e uma medida dessa mudança de estado é informada ao agente através de um valor de sinal de reforço, r, este processo interativo permi-

te ao agente aprendiz após um determinado número de iterações definir qual ação tomar em cada estado (LIMA JÚNIOR, 2009).

De acordo com Almeida (2014, *apud* SUTTON; BARTO, 1998), além de agente e ambiente, pode-se considerar quatro elementos básicos de AR a saber:

- **Política:** uma política define o comportamento do agente aprendiz, isto é, como ele deve escolher suas ações em um dado instante de tempo. Representa o núcleo de um agente de AR no sentido que ela sozinha é suficiente para determinar o comportamento. Seja S o conjunto de estados do ambiente e A(s) o conjunto de ações disponíveis para um estado s ∈ S. Uma política π é uma função de mapeamento de estados para ações:

$$\alpha = \pi(s), \forall \alpha \in A(s), s \in S,$$

  ou seja, dado um estado s a política determina qual é a ação que o agente deverá tomar. Geralmente, as políticas podem ser estocásticas.

- **Função de recompensa:** é quem define o objetivo em um problema de AR. Ela associa cada estado (ou par estado-ação) a um sinal numérico (uma recompensa), indicando a atratividade do estado. Assim, o agente tem como objetivo a maximização da quantidade total de reforços recebidos ao longo da execução, denominado de retorno acumulado. Para que o valor do retorno não se torne muito grande um fator de desconto γ ($0 \leq \gamma \leq 1$) é introduzido para reduzir gradativamente os valores futuros.

- **Função valor:** é obtida utilizando o reforço atual e futuro. Enquanto a função de recompensa atribui um sinal imediato às boas escolhas, a função valor indica uma estimativa do ganho total que pode ser acumulado pelo agente durante a aprendizagem, partindo de um estado s, e considerando os estados que o sucedem. Por exemplo, um estado pode sempre produzir baixas recompensas imediatas, mas ainda assim ter um alto valor, pois ele é regularmente frequentado por outros estados que produzem altas recompensas. Quando a função valor considera apenas o estado s, ela é denominada função valor-estado, e denotada por V (s):

$$V(S) = E\{r_{t+1} + r_{t+2} + \ldots + r_{t+n} \,|s_t\} = s,$$

  onde E é o valor total esperado dos retornos acumulados pelo agente seguindo uma política π, a partir de um estado s_t = s, no instante t . No caso em que a função valor considera as ações possíveis para um dado estado, esta é denominada função valor estado-ação e é denotada por Q(s, a), como segue:

$$Q(s,a) = E\{r_{t+1} + r_{t+2} + \ldots + r_{t+n} \,|s_t = s, a_t = a\}.$$

- **Modelo do ambiente**: o modelo representa o comportamento do ambiente, ou seja, dados o estado e a ação, o modelo pode antecipar o próximo estado e a próxima recompensa de acordo com a probabilidade de transição.

4.1. Algoritmo *Q-learning*

O intuito de um sistema de Aprendizado por Reforço, mostrado na Figura 16 é de encontrar uma política ideal (menor custo possível) após tentar várias sequências de ações, e observar os custos associados a cada mudança de estado.

O algoritmo *Q-learning* pode ser considerado o mais importante método de aprendizagem por reforço, pois consiste num método no qual um agente toma decisões de acordo com o benefício ou perda que as decisões lhes trouxeram (ALMEIDA, 2014). Segundo Lima Júnior (2009) o *Q-learning* foi o primeiro método de AR a possuir fortes provas de convergência, e que apesar do critério de convergência exigir um número infinito de episódios, na prática executa-se um número suficientemente grande de acordo com o tamanho da tarefa a ser aprendida.

O *Q-learning* é capaz de aprender diretamente a partir da experiência, dispensando a necessidade de uma modelagem completa do ambiente para a determinação de uma política ótima de atuação. Assim, sua convergência para valores ótimos de Q independe da política utilizada. Seja s_t o estado corrente, a_t a ação realizada no estado s_t, r_t o reforço imediato obtido após executar a ação a_t em s_t , s_{t+1} o estado seguinte, max $Q(s_{t+1}, a)$ a máxima função valor (que encontra e seleciona a ação do estado seguinte que a maximize), e $\gamma \in [0, 1]$ e $\alpha q \in [0, 1]$, respectivamente, o fator de desconto e o coeficiente de aprendizagem (LIMA JÚNIOR, 2009). Para atualizar a matriz dos Q-valores, o *Q-learning* utiliza a seguinte expressão:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_q[r_t + \gamma max Q(s_t + 1, a) - Q(s_t, a_t)]$$

Segundo Almeida (2014) na implementação do *Q-learning*, um agente aprende qual a melhor ação a ser escolhida a cada nova iteração, chamada de episódio. A Figura 17 apresenta o pseudocódigo do *Q-learning* de forma simplificada.

```
1: procedimento QLearning(α_q, ε, γ, NumEpis)
2:     Inicializar R(s,a) {Matriz de recompensas}
3:     Inicializar Q(s,a) {Matriz dos Q-valores inicialmente zerada}
4:     repita
5:         Inicializar s {Estado inicial}
6:         repita
7:             Selecionar uma ação a de acordo com a política apropriada
8:             Receber a recompensa R(s,a) e observar o próximo estado s'
9:             Atualizar Q(s,a)
10:            s ← s'
11:        até encontrar um estado final
12:    até NumEpis
13:    retorne Q(s,a)
14: fim procedimento
```

Figura 17 – Pseudocódigo Q-Learning (ALMEIDA, 2014).

4.2. Metaheurística *GRASP-Learning*

Lima Júnior (2009) propôs a utilização da metaheurística *GRASP* com AR utilizando o algoritmo *Q-learning* (*GRASP-Learning*) em substituição a fase de construção guloso-aleatória da *GRASP* (Figura 18).

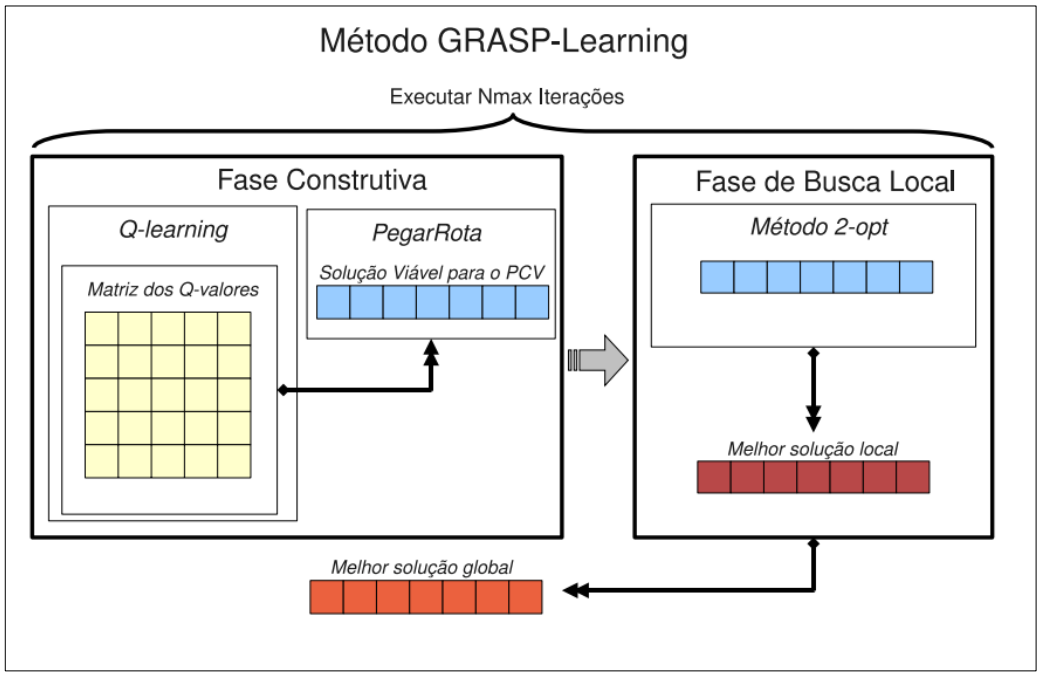


Figura 18 – Visão geral do método *GRASP-Learning* (LIMA JÚNIOR, 2009).

Na *GRASP* tradicional as iterações são independentes, a iteração atual não faz uso de informações das iterações passadas. No método *GRASP-Learning*, o *Q-learning* foi utilizado como construtor de soluções iniciais para a *GRASP*, de forma que a cada iteração do algoritmo uma solução viável para PCV fosse construída.

A ideia básica do método proposto foi fazer uso das informações contidas na matriz dos Q-valores do algoritmo *Q-learning*, como forma de uma memória que permitisse repetir boas decisões tomadas em iterações anteriores e evitar aquelas que não foram interessantes.

Conforme já apresentado, a taxa de aleatoriedade ou gula da *GRASP* é definida pelo parâmetro α, no *GRASP-Learning* esta taxa é definida pelo parâmetro ε, em que inicialmente escolhas mais aleatórias são feitas e à medida que o número de episódios aumenta o aspecto guloso é mais explorado (LIMA JÚNIOR, 2009).

4.3. Metaheurística *GR-Learning*

Almeida (2014) apresentou mais uma abordagem híbrida utilizando a metaheurística *GRASP* Reativo juntamente com o algoritmo *Q-learning* (*GR-Learning*). No *GRASP* Reativo tradicional o parâmetro α utilizado para o controle da intensidade de gula/aleatoriedade da geração da solução na fase construtiva é escolhido através uma distribuição probabilística, onde um conjunto de valores α previamente determinados que possuírem melhor desempenho tem uma maior probabilidade de serem escolhidos novamente, porém sem garantia que sejam realmente escolhidos.

No algoritmo *GR-Learning* o parâmetro α é escolhido a partir da matriz dos Q-valores, que representa o conhecimento adquirido do agente. O trabalho proposto se diferencia do trabalho de Lima Júnior (2009) por não ser necessário modelar o problema a ser resolvido como um Processo de Decisão de Markov (PDM), e nesse caso a fase construtiva do *GRASP* Reativo não ser alterada, visto que o agente vai utiliza a matriz dos Q-valores retornado pelo *Q-learning* para aprender qual α deverá ser utilizado durante a fase de construção nas próximas iterações.

```
1: procedimento GR_Learning(D, α_q, ε, γ, NumEpis)
2:     f(x*) ← +∞; iter ← 1
3:     Defina A = {α_1, α_2, ..., α_v}
4:     enquanto não CritérioParada faça
5:         α ← AprenderAlfa(Q)
6:         x_1 ← FaseConstrutiva(D, α)
7:         x_2 ← BuscaLocal(D, x_1)
8:         se f(x_2) < f(x*) então
9:             x* ← x_2
10:        fim se
11:        se iter mod y = 0 então
12:            Q ← Qlearning(A, f(x*), α_q, ε, γ, NumEpis)
13:        fim se
14:        iter ← iter + 1
15:    fim enquanto
16:    retorne x*
17: fim procedimento
```

Figura 19 – Pseudocódigo GR-Learning (ALMEIDA, 2014) adaptado.

Conforme apresentado na Figura 19, a estrutura da *GRASP* original não sofre nenhuma alteração significativa, uma vez que o método `AprenderAlfa` na linha 5 não modifica a forma como a solução inicial é construída, na linha 11, a cada *y* iterações, o método `Q-learning` é executado devolvendo a matriz dos Q-valores que será utilizada para escolha do α em futuras iterações.

CAPÍTULO 5

O FRAMEWORK PROPOSTO

Como foi visto no Capítulo 3, uma metaheurística tem que ser modelada de acordo com o problema a qual está sendo aplicada, ficando assim sua implementação fortemente acoplada ao problema ao qual foi projetada para resolver. Em Lima Júnior (2009) foi desenvolvido o método *GRASP-Learning*, onde o mesmo foi utilizado para resolver instâncias do problema do caixeiro viajante. Porém foi implementado de maneira que o problema deveria ser modelado como um Processo de Decisão de Markov, ou seja, para utilizar o mesmo método em um problema diferente, teria um novo esforço para modelar esse problema novamente.

Em Almeida (2014) o autor aprimorou a ideia de Lima Júnior (2009) e desenvolveu o *GR-Learning*. No algoritmo proposto o autor utilizou o *Q-learning* para aprender qual α deve ser utilizado para gerar as melhores soluções iniciais na fase de construção. O *GR-Learning* foi e utilizado para resolver o problema dos p-Centros. Apesar da evolução alcançada, o código ficou fortemente acoplado ao problema abordado pelo autor.

A proposta deste trabalho é desenvolver um *framework* que permita utilizar o algoritmo *GR-Learning* desenvolvido em Almeida (2014) para ser utilizado em diversos problemas de otimização combinatória sem que seja necessário despender tanto esforço para adaptação a cada novo problema abordado.

5.1. Detalhes da implementação do *framework* proposto

O *framework* foi desenvolvido em Java devido ser uma extensão do trabalho de Almeida (2014), e o código original ter sido desenvolvido nesta referida linguagem. Após o *refactoring* no código original o *framework* ficou com as seguintes classes: `Qlearning`, `GRLearning`, `Metodos` e `MetodosHeuristicos`.

A classe `QLearning` pode ser considerada como a principal classe do *framework*, pois nesta classe é implementado os métodos responsáveis pela construção da matriz dos Q-valores utilizada para definir qual α deve ser utilizado. A classe `GRLearning` contém a implementação da *GRASP* Reativo com utilização de AR, juntamente com os métodos responsáveis pelas chamadas para as funções contidas na classe `QLearning`, e também o método

main responsável pela execução do algoritmo. A classe *MetodosHeuristicos* é uma classe abstrata que possui dois métodos abstratos: *GeraSolucao* e *BuscaLocal*.

A classe *Metodos* estende a classe *MetodosHeuristicos* e herda da classe mãe os dois métodos: *GeraSolucao* e *BuscaLocal*, que devem ser sobrescritos. Essa é uma condição obrigatória para o funcionamento do *framework*, a partir daí o desenvolvedor fica livre para criar as classes, métodos e atributos que forem necessários para abordar o problema que deseje trabalhar. A seguir veremos a descrição das classes e os principais métodos utilizados no *framework*.

Tabela 6 – Descrição da Classe `QLearning`.

Métodos	Descrição
QLearning	Método construtor responsável por inicializar os parâmetros necessários para execução do algoritmo.
setRecompensa	Método responsável por atualizar a matriz de recompensas.
acaoEps	Método responsável por decidir se a mudança de estado será feita aleatoriamente ou através da gula com base no parâmetro épsilon.
acaoAleatoria	Método que escolhe uma ação em um estado de maneia aleatória.
acaoGulosa	Método responsável por escolher o estado que gera a melhor solução.
Acao	Método que decide se há uma mudança de estado ou não. A política de escolha de ação ocorre de forma aleatória.
escolherAlfa	Método responsável por escolher um α dentre os não visitados.
qLearning	Método responsável por implementar o algoritmo QLearning.

A Tabela 6 apresenta os principais métodos da classe *QLearning*, juntamente com a descrição das ações executadas por cada um deles.

Tabela 7 – Descrição da Classe `GRLearning`.

Métodos	Descrição
reativoLearning	Método responsável por executar as etapas da metaheurística *GRASP* Reativo com AR.
aprenderAlfa	Método responsável por escolher o melhor α de acordo com o aprendizado do algoritmo *Q-learning*.

A Tabela 7 apresenta os dois principais métodos da classe *GRLearning*. O método *reativoLearning*, que implementa a metaheurística *GRASP* Reativo com AR, e o método *aprenderAlfa*, que é o método responsável por escolher o melhor α de acordo com o aprendizado do algoritmo *Q-learning*.

Tabela 8 – Descrição da Classe Abstrata *MetodosHeuristicos*.

Métodos	Descrição
geraSolucao	Método abstrato que deverá ser implementado na classe filha *Metodos*. Este método é o responsável por gerar uma solução inicial durante a fase construtiva da metaheurística.
buscaLocal	Método abstrato que deverá ser implementado na classe filha *Metodos*. Este método é o responsável pelas estratégias de refinamento da solução inicial durante a fase de busca local.

A Tabela 8 apresentou os detalhes da classe abstrata *MetodosHeuristicos*. conforme a descrição dos métodos geraSolucao e buscaLocal, os mesmos devem ser implementados na classe filha *Métodos*, que será descrita na Tabela 9.

Tabela 9 – Descrição da Classe *Metodos*.

Métodos	Descrição
geraSolucao	Este método é herdado da classe *MetodosHeuristicos*, e deverá ser implementado as ações necessárias para a geração de uma solução inicial.
buscaLocal	Este método é herdado da classe *MetodosHeuristicos*, e deverá ser implementado as ações necessárias para o refinamento da solução inicial.

Para utilização do *framework* aplicado a resolução do *HLP* foi criada uma classe chamada *Solucao*, responsável pelo armazenamento das soluções criadas durante a execução do algoritmo. E na classe *Metodos* foram adicionados os atributos e métodos necessários para aplicação da metaheurística ao problema do *HLP*. Além disso, foi feito a sobrescrita dos métodos *Gerasolucao* e *BuscaLocal*, que conforme já mencionado é condição obrigatória para o funcionamento do *framework*.

Tabela 10 – Descrição da Classe *Metodos* com aplicação do *framework* ao *CSP*.

Métodos	Descrição
geraSolucao	Este método é herdado da classe *MetodosHeuristicos*, e deverá ser implementado as ações necessárias para a geração de uma solução inicial.
buscaLocal	Este método é herdado da classe *MetodosHeuristicos*, e deverá ser implementado as ações necessárias para o refinamento da solução inicial.
escolheHubInicial	Método responsável por escolher de maneira aleatória *p-hubs* a serem utilizados na solução inicial.
alocaSpokes	Método responsável por fazer a alocação dos nodos não *hubs* aos *hubs* selecionados no método anterior.

Conforme pode ser visto na Tabela 10, além de os métodos *Gerasolucao* e *BuscaLocal* que devem ser implementados, o desenvolvedor ficar livre para adicionar outros métodos, atributos e classes que julgue ser necessário.

Para aplicação do *framework* na resolução do *CSP* foram criadas duas classes chamadas *Objeto* e *Itens*, que são necessárias para compor uma solução. E em destaque, na Tabela 10, na classe *Metodos* foi adicionados o método *GRASP2d*, e feito à implementação dos métodos *Gerasolucao* e *BuscaLocal*. A seguir a Tabela 11 com a descrição da classe *Metodos* com aplicação do *framework* ao *HLP*.

Tabela 11 – Descrição da Classe *Metodos* com aplicação do *framework* ao *CSP*.

Métodos	Descrição
geraSolucao	Este método é herdado da classe *MetodosHeuristicos*, e deverá ser implementado as ações necessárias para a geração de uma solução inicial.
buscaLocal	Este método é herdado da classe *MetodosHeuristicos*, e deverá ser implementado as ações necessárias para o refinamento da solução inicial.
GRASP2d	Método responsável por escolher de maneira aleatória *p-hubs* a serem utilizados na solução inicial.

Como foi apresentado, com a utilização do *framework* pode-se adaptar a metaheurística *GR-Learning* para diversos problemas sem que seja necessário se preocupar em modelá-la de acordo com o problema a ser tratado. Todas as classes responsáveis pelo funcionamento da metaheurística estão implementadas de maneira que o seu funcionamento depende apenas da sobrescrita dos métodos *GeraSolução* e *BuscaLocal*. Com a atual estrutura do *framework* é possível utilizar os benefícios do *GR-Learning* independente do problema tratado, ficando ainda o pesquisador livre para criar as classes e métodos que achar necessário sem que essas alterações afetem o funcionamento da metaheurísitca *GRASP* Reativo, ou prejudique a técnica aplicada para aprendizagem por reforço.

Os detalhes dos resultados obtidos com a utilização do *framework*, juntamente com as instâncias utilizadas para execução dos testes, são apresentados nos capítulos 6 e 7 respectivamente.

CAPÍTULO 6

INSTÂNCIAS UTILIZADAS

Para validação dos resultados obtidos com o *framework* proposto foram utilizadas instâncias da literatura que serviram de *benchmark* para outros autores em diversos trabalhos, com exceção da instância BR2010 que foi criada a partir de dados constantes no Anuário da ANAC 2011. A seguir será apresentado o detalhamento de cada instância utilizada.

6.1. A INSTÂNCIA CAB

A instância CAB contém dados referentes ao fluxos de passageiros entre 25 aeroportos que juntos correspondiam ao percentual de 51% do tráfego dos *EUA* nos anos 70, apesar de ser uma instância de pequeno porte (apenas 25 nodos) a mesma foi bastante utilizada por diversos outros autores para validar seus método propostos. A instância está disponível em: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/phub4.txt (acessado em 03/04/2016).

6.2. A INSTÂNCIA BR2010

A instância BR2010 contém dados relativos a fluxo de passageiros referente ao ano de 2010 entre 147 aeroportos e aeródromos do Brasil. O arquivo é composto por duas matrizes 147 x 147, a primeira matriz é referente ao fluxo de passageiros entre os aeroportos e a segunda matriz referente ao custo unitário por fluxo movimentado de um aeroporto para outro. A última linha do arquivo representa os números dos aeroportos que atendem as condições para atuarem como hub conforme Siqueira (2008).

Para o cálculo do custo unitário por fluxo foi utilizado os as informações referentes às coordenadas de latitude e longitude dos aeroportos para calcular a distância entre dois pontos em uma esfera (d), conforme Alzamora (2013), depois calculado o custo médio por Km (c) de acordo com a Tabela 12, e dividido pela quantidade média de passageiros por Km pago (do inglês *Revenue Passenger Kilometers* - RPK), ficando assim a equação: $custo\ unitário\ por\ fluxo = \frac{d*c}{rpk}$.

Tabela 12 – Evolução do tráfego aéreo doméstico brasileiro - 2002 a 2010.

Ano	Km voados	RPK	Despesa de Voo (R$)	Custo Médio Km Voado	Quantidade Média RPK
2002	427.485.399	27.677.332.808	8.624.651.944	20,17531	64,74451
2003	361.268.780	26.026.654.918	8.503.030.078	23,53658	72,04236
2004	354.107.919	28.263.315.152	9.223.946.884	26,04841	79,81554
2005	387.679.455	35.565.915.245	10.163.601.168	26,21651	91,74052
2006	421.988.607	40.565.954.820	10.601.231.714	25,12208	96,13045
2007	470.334.476	45.749.598.117	11.546.127.095	24,54876	97,27035
2008	502.648.064	49.717.181.793	14.749.507.720	29,34361	98,91052
2009	580.829.657	56.862.461.425	14.546.190.854	25,04382	97,89869
2010	**688.831.373**	**70.103.130.399**	**16.863.466.997**	**24,48127**	**101,7711**

A decisão de utilizar os dados constantes no anuário de 2011 (referente aos dados de 2010) se deu pelo motivo que a partir da edição de 2012 (referente aos dados de 2011) não mais foi disponibilizado o fluxo de passageiros transportados entre os aeroportos.

6.3. INSTÂNCIAS PARA O PROBLEMA DE CORTE BIDIMENSIONAL

Para validação do algoritmo na resolução do problema de corte bidimensional guilhotinado e restrito, foi utilizado um total de três conjuntos de instâncias da literatura: o primeiro conjunto formado por oito instâncias CW1, CW2 e CW3 utilizadas em Christofides e Withlock (1977), OF1 e OF2 utilizados em Oliveira e Ferreira (1990), WANG1, WANG2 e WANG3 utilizados em Wang (1983), o segundo conjunto de instâncias formado pelas instâncias de CU01 a CU11 e o terceiro conjunto de instâncias nomeadas de CW1 a CW11 ambos utilizados em Morabito e Pureza (2002). Instâncias disponíveis em: https://web.fe.up.pt/~esicup/datasets?category_id=3 (Acessado em 03/04/2016).

Tabela 13 – Instâncias utilizadas para validação do *framework*.

1º Conjunto			2º Conjunto			3º Conjunto		
Instância	**N**	**L x W**	**Instância**	**N**	**L x W**	**Instância**	**N**	**L x W**
CW1	7	15x10	CU1	25	100x125	CW1	25	125x105
CW2	10	40x70	CU2	35	150x175	CW2	35	145x165
CW3	20	40x70	CU3	45	134x125	CW3	40	267x207
OF1	10	70x40	CU4	45	285x354	CW4	39	465x387
OF2	10	70x40	CU5	50	456x385	CW5	35	524x678
WANG1	20	69x33	CU6	45	356x447	CW6	55	781x657
WANG2	20	70x39	CU7	45	563x458	CW7	45	276x374
WANG3	20	70x40	CU8	35	587x756	CW8	60	305x287

	CU9	25	856x785	CW9	50	405x362
	CU10	40	794x985	CW10	60	992x970
	CU11	50	977x953	CW11	60	982x967

Conforme pode ser observado na Tabela 13, o primeiro conjunto possui instâncias com no máximo 20 itens, já no segundo conjunto encontramos instâncias com até 50 itens e o terceiro conjunto apresenta instâncias com até 60 itens.

CAPÍTULO 7

TESTES COMPUTACIONAIS

O *framework* foi implementado utilizando a linguagem Java e os testes foram executados em uma máquina com processador Intel Core I7, com 8GB de memória RAM, utilizando o sistema operacional Windows 8.1 64 bits. A seguir é apresentado os detalhes referente à implementação dos algoritmos e dos resultados computacionais obtidos.

7.1. *GR-LEARNING* APLICADO AO HUB LOCATION PROBLEM

Para representar uma solução para o *HLP* foi estabelecido dois vetores conforme pode ser visto na Figura 20.

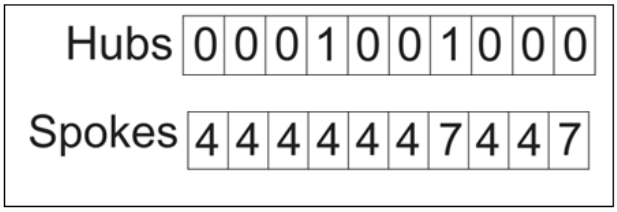


Figura 20 – Representação de uma solução para o *HLP*.

No vetor *hubs* quando o valor for igual a 1 significa que o nodo *i* é um *hub*, e zero caso contrário. No vetor *spokes* o valor significa a qual *hub* o *spoke* está alocado. Na Figura 20, os nodos 4 e 7 estão alocados como *hubs*, e os nodos 1,2,3,5,6,8 e 9 estão alocados ao *hub* 4, já o nodo 10 está alocado ao *hub* 7.

7.1.1. Fase de construção

Para a fase de construção foi implementado uma função que sobrescreve o método `GeraSolucao` do *framework* proposto. Na Figura 20 é apresentado o pseudocódigo da fase de construção.

```
1: procedimento GeraSolucao(α)
2:     escolheHubInicial()
3:     alocaSpokes(α)
4:     calculaFo()
5:     retorne fo
6: fim procedimento
```

Figura 21 – Pseudocódigo da fase de construção.

Observando a Figura 21, na linha 2 temos o procedimento `escolheHubInicial`, que sorteia uma número *p* de *hubs* de maneira aleatória. Na linha 3 a função `alocaSpokes` recebe o parâmetro α e a partir do vetor de *hubs*, partindo do primeiro *hub*, ordena uma lista de candidatos (todos os nodos que ainda não estão alocados) a compor a solução de acordo com o valor de contribuição para função objetivo, faz o calculo da quantidade de elementos da LRC e sorteia aleatoriamente um elemento para compor a solução. Após isso vai para o próximo *hub* e repete o procedimento até que todos os nodos estejam alocados. No próximo passo, na linha 4, faz o cálculo do valor da função objetivo e retorna o valor encontrado.

7.1.2. Fase de busca local

Durante a fase de busca local, foram estabelecidas duas estratégias de busca em vizinhança, a primeira estratégia consiste em escolher um *hub* e um *spoke* aleatoriamente, então o *hub* passa a ser *spoke* e o *spoke* passa a ser *hub*.

Na Figura 22.a, uma solução inicial para o *HLP* é criada e na primeira estratégia da fase de busca local e na Figura 22.b um *hub* e um *spoke* são sorteados de maneira aleatória, no exemplo foram sorteados o *hub* 4 e o *spoke* 3, após isso *hub* 4 passou a ser um *spoke*, e o *spoke* 3 passou a ser um *hub*, e os *spokes* alocados ao *hub* 4 são trocados e alocados agora ao *hub* 3, Figura 22.c.

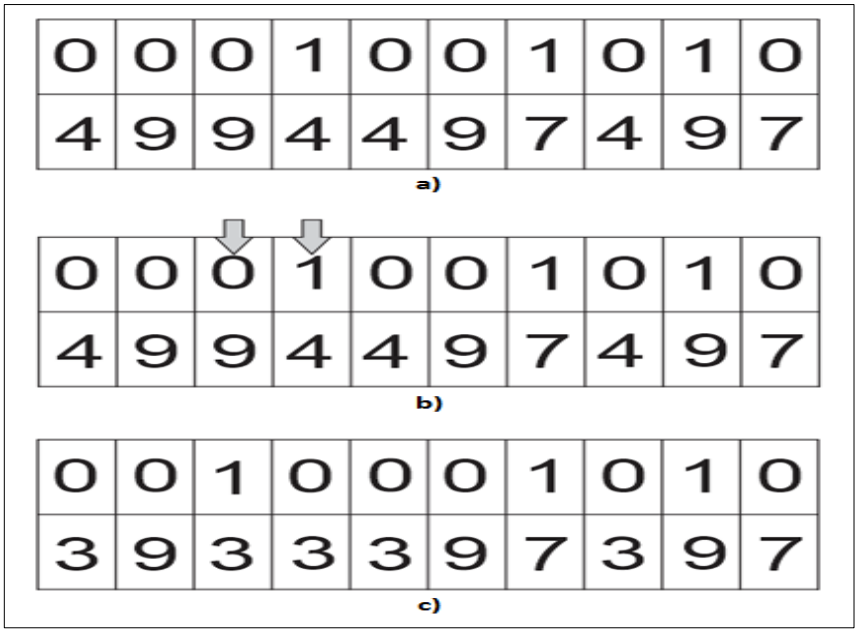


Figura 22 – Estratégia de busca local 1.

Esse procedimento se repete até um número x de tentativas sem melhoras que deve ser calibrado pelo usuário, e para essa estratégia somente é aceito movimentos de melhora, ou seja caso não exista melhora na função objetivo após a permutação dos *spokes* as alterações são descartadas e retorna a solução inicial.

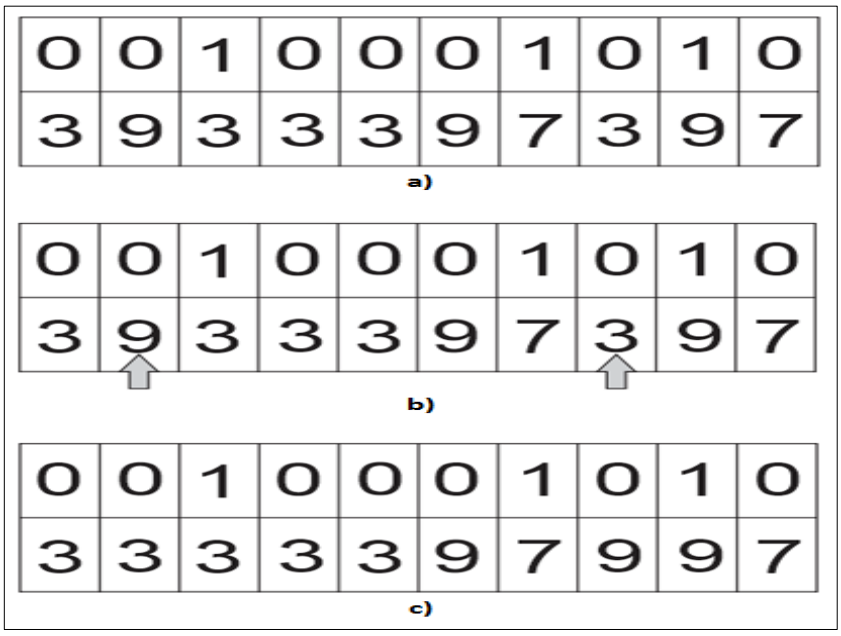


Figura 23 – Estratégia de busca local 2.

A segunda estratégia de busca local verifica estruturas de vizinhança a partir da solução obtida após a primeira estratégia de busca local, dois *spokes* são sorteados de maneira aleatória e tem seus *hubs* permutados.

Na Figura 23.a, a solução inicial é originada a partir da primeira busca local, na Figura 23.b dois *spokes* são sorteados de maneira aleatória, no exemplo foram sorteados os *spokes* 3 e 8 que estavam alocados aos *hubs* 9 e 7 respectivamente, após isso os *spokes* tiveram seus *hubs* permutados entre si, o que pode ser visto na Figura 23.c, de maneira semelhante à primeira estratégia de busca local esse procedimento se repete por um número x de tentativas sem melhoras que deve ser calibrado pelo usuário.

7.1.3. Resultados computacionais para a instância CAB

A Tabela 14 apresenta os resultados para os testes computacionais com a instância CAB, a Tabela apresenta resultados para configuração de 10x10, 15x15, 20x20 e 25x25 ambos com 2, 3 e 4 hubs, a coluna Z* se refere ao valor ótimo encontrado em O'Kelly (1996), a coluna Sol. representa a melhor solução encontrada pelo algoritmo, a coluna média foi obtida após 30 execuções do *GR-Learning* com 1000 iterações a cada execução, na coluna Desv. tem-se o desvio padrão calculado por: $Desv = 100\frac{(médias - Sol.)}{Sol.}$, e finalizando a coluna GAP se refere ao valor de aproximação da melhor solução encontrada em relação a solução ótima e pode ser definida pela seguinte equação: $gap = 100\frac{(Sol. - z*)}{z*}$.

Tabela 14 – Resultados do GR-Learning para instância CAB.

Instância	Z*	Sol.	Média	Maior	Desv.	GAP	T(s)	Hubs
10x2	835,81	835,81	835,81	835,81	0,00	0,00	0,93	4, 7
10x3	776,68	776,68	776,68	776,68	0,00	0,00	0,90	4, 7, 9
10x4	736,26	736,26	736,26	736,26	0,00	0,00	0,60	1, 4, 7, 9
15x2	1221,92	1221,92	1221,92	1221,92	0,00	0,00	1,26	4, 11
15x3	1167,23	1167,23	1174,64	1182,79	0,63	0,00	1,26	1, 4, 12
15x4	1118,22	1118,22	1129,27	1135,92	0,99	0,00	1,13	1, 4, 7, 8
20x2	1210,08	1210,08	1210,08	1210,08	0,00	0,00	2,30	4, 20
20x3	1156,07	1156,07	1162,88	1173,86	0,59	0,00	2,30	4, 11, 20
20x4	**1107,92**	**1111,81****	1118,86	1127,74	0,63	3,89	2,40	4, 7, 8, 20
25x2	**1359,19**	**1362,15**	1362,15	1362,15	0,00	2,96	3,90	4, 20
25x3	1256,63	1256,63	1283,21	1309,05	2,12	0,00	3,90	4, 8, 20
25x4	1211,23	1211,23	1220,00	1246,49	0,72	0,00	4,03	4, 7, 8, 20

**** Não possui valor ótimo para solução em O'Kelly (1996).**

Conforme pode ser visto na Tabela 14, o *GR-Learning* conseguiu obter o valor ótimo em quase todas as instâncias testadas, com exceção da instância 25x25 com 2 hubs, e a instância 20x20 com 4 hubs, porém para essa última não foi encontrado valores inteiros para solução obtida em O'Kelly (1996). Pode-se observar também o desvio padrão em relação as soluções obtidas, onde o maior foi 2,12%, o que demonstra que na maioria das vezes as soluções obtidas são de boa qualidade.

Os resultados até agora apresentados se referem a formulação (1.1), e não foram levados em consideração nenhum custo de alocação de hubs, como também o fator de desconto β, que representa a economia de escala entre o fluxo roteado via hubs.

Tabela 15 – Resultados do GR-Learning para instância CAB. Custos fixos e n = 20.

β	f_k	Z*	Sol.	Gap.	Média	Maior	Desv.	T(s)	Hubs
0,2	100	967,74	**977,62**	**1,02**	980,48	986,18	0,29	1,63	4, 12, 16, 17
	150	1174,53	1174,53	0,00	1179,22	1187,82	0,40	1,72	4, 12, 17
	200	1324,53	1324,53	0,00	1329,22	1337,82	0,35	1,72	4, 12, 17
	250	1474,53	1474,53	0,00	1479,22	1487,82	0,32	1,72	4, 12, 17
0,4	100	1127,09	1127,09	0,00	1129,32	1136,81	0,20	1,64	1, 4, 12, 17
	150	1297,76	1297,76	0,00	1305,76	1323,55	0,62	1,72	4, 12, 17
	200	1442,56	1442,56	0,00	1442,56	1442,56	0,00	1,78	4, 17
	250	1542,56	1542,56	0,00	1542,56	1542,56	0,00	1,78	4, 17
0,6	100	1269,15	**1270,99**	**0,14**	1285,75	1298,26	1,16	1,73	4, 12, 18
	150	1406,04	1406,04	0,00	1406,36	1410,84	0,02	1,74	4, 17
	200	1506,04	1506,04	0,00	1506,36	1510,84	0,02	1,74	4, 17
	250	1570,91	1570,91	0,00	1570,91	1570,91	0,00	1,73	6
0,8	100	1369,52	1369,52	0,00	1370,43	1373,43	0,07	1,79	4, 17
	150	1469,52	1469,52	0,00	1470,43	1473,43	0,06	1,79	4, 17
	200	1520,91	1520,91	0,00	1520,91	1520,91	0,00	1,77	6
	250	1570,91	1570,91	0,00	1570,91	1570,91	0,00	1,77	6
1	100	1410,07	1410,07	0,00	1410,07	1410,07	0,00	1,71	4, 20
	150	1470,91	1470,91	0,00	1470,91	1470,91	0,00	1,71	6
	200	1520,91	1520,91	0,00	1520,91	1520,91	0,00	1,71	6
	250	1570,91	1570,91	0,00	1570,91	1570,91	0,00	1,71	6

Tabela 16 – Resultados do GR-Learning para instância CAB. Custos fixos e n = 25.

B	f_k	Z*	Sol.	Gap.	Média	Maior	Desv.	T(s)	Hubs
0,2	100	1029,63	1029,63	0,00	1033,43	1037,84	0,37	2,67	4, 12, 17, 24
	150	1217,34	1217,34	0,00	1217,34	1217,34	0,00	2,76	4, 12, 17
	200	1367,34	1367,34	0,00	1367,34	1367,34	0,00	2,76	4, 12, 17
	250	1500,9	1500,9	0,00	1500,9	1500,9	0,00	2,82	12, 20
0,4	100	1187,51	1187,51	0,00	1190,63	1197,53	0,26	2,62	1, 4, 12, 17
	150	1351,69	1351,69	0,00	1355,05	1363,06	0,25	2,85	4, 12, 18
	200	1501,62	1501,62	0,00	1524,88	1569,39	1,55	2,75	12, 20
	250	1601,62	1601,62	0,00	1624,88	1669,39	1,45	2,75	12, 20

0,6	100	1333,56	1333,56	0,00	1339,02	1358,83	0,41	2,71	2, 4, 12
	150	1483,56	1483,56	0,00	1489,02	1508,83	0,37	2,71	2, 4, 12
	200	1601,2	**1633,56**	**2,02**	1639,02	1508,83	0,33	2,71	2, 4, 12
	250	1701,2	1701,2	0,00	1732,44	1760,22	1,84	2,73	12, 20
0,8	100	1458,83	**1462,06**	**0,22**	1476,06	1504,58	0,96	2,72	2, 4, 12
	150	1594,08	**1597,51**	**0,22**	1613,67	1624,19	1,01	2,74	21, 25
	200	1690,57	1690,57	0,00	1690,57	1690,57	0,00	2,71	5
	250	1740,57	1740,57	0,00	1740,57	1740,57	0,00	2,71	5
1	100	1556,63	1556,63	0,00	1609,05	1583,21	3,37	2,69	4, 8, 20
	150	1640,57	1640,57	0,00	1640,57	1640,57	0,00	2,65	5
	200	1690,57	1690,57	0,00	1690,57	1690,57	0,00	2,65	5
	250	1740,57	1740,57	0,00	1740,57	1740,57	0,00	2,65	5

Verificando as Tabelas 15 e 16 é possível observar que em quase todos os casos o algoritmo encontrou a solução ótima, e nos casos onde não obteve a solução ótima gerou soluções de boa qualidade com o *GAP* abaixo de 2,1% para a maior diferença encontrada. As próximas Tabelas apresentam um comparativo entre o desempenho obtido pelo algoritmo e os resultados obtidos em Almeida (2009).

Tabela 17 – GR-Learning x Almeida (2009) – Custos fixos e n = 20.

B	f_k	Z*	Sol.	Gap	Desv.	T(s)	CSSATL	GAP	Desv.	T(s)	SATL	Gap	Desv.	T(s)
0,2	100	967,74	977,62	1,02	0,29	1,63	977,62	1,02	0,00	1,25	977,62	1,02	0	0,23
	150	1174,53	1174,53	0,00	0,40	1,72	1174,53	0,00	0,00	0,12	1174,53	0,00	0	0,16
	200	1324,53	1324,53	0,00	0,35	1,72	1324,53	0,00	0,00	0,10	1324,53	0,00	0	0,15
	250	1474,53	1474,53	0,00	0,32	1,72	1474,53	0,00	0,34	0,16	1474,53	0,00	0	0,12
0,4	**100**	**1127,09**	**1127,09**	**0,00**	**0,20**	**1,64**	1127,09	0,00	0,42	**1,00**	**1136,74**	**0,86**	**0,47**	**0,16**
	150	1297,76	1297,76	0,00	0,62	1,72	1297,76	0,00	0,00	0,48	1297,76	0,00	0	0,17
	200	1442,56	1442,56	0,00	0,00	1,78	1442,56	0,00	0,00	0,37	1442,56	0,00	0	0,17
	250	1542,56	1542,56	0,00	0,00	1,78	1542,56	0,00	0,37	0,06	1542,56	0,00	0,37	0,1
0,6	**100**	**1269,15**	**1270,99**	**0,14**	**1,16**	**1,73**	**1269,15**	**0,00**	**0,00**	**0,15**	**1270,99**	**0,14**	**0**	**0,15**
	150	1406,04	1406,04	0,00	0,02	1,74	1406,04	0,00	0,75	1,45	1406,04	0,00	0,75	0,12
	200	1506,04	1506,04	0,00	0,02	1,74	1506,04	0,00	0,00	0,11	1506,04	0,00	0	0,08
	250	1570,91	1570,91	0,00	0,00	1,73	1570,91	0,00	0,00	0,07	1570,91	0,00	0	0,08
0,8	100	1369,52	1369,52	0,00	0,07	1,79	1369,52	0,00	0,44	0,07	1369,52	0,00	0,46	0,2
	150	1469,52	1469,52	0,00	0,06	1,79	1469,52	0,00	0,00	0,07	1469,52	0,00	0	0,09
	200	1520,91	1520,91	0,00	0,00	1,77	1520,91	0,00	0,00	0,08	1520,91	0,00	0	0,09
	250	1570,91	1570,91	0,00	0,00	1,77	1570,91	0,00	0,00	0,09	1570,91	0,00	0	0,08
1	100	1410,07	1410,07	0,00	0,00	1,71	1410,07	0,00	0,00	0,14	1410,07	0,00	0	0,09
	150	1470,91	1470,91	0,00	0,00	1,71	1470,91	0,00	0,00	0,12	1470,91	0,00	0	0,09
	200	1520,91	1520,91	0,00	0,00	1,71	1520,91	0,00	0,00	0,10	1520,91	0,00	0	0,08
	250	1570,91	1570,91	0,00	0,00	1,71	1570,91	0,00	0,00	0,07	1570,91	0,00	0	0,08

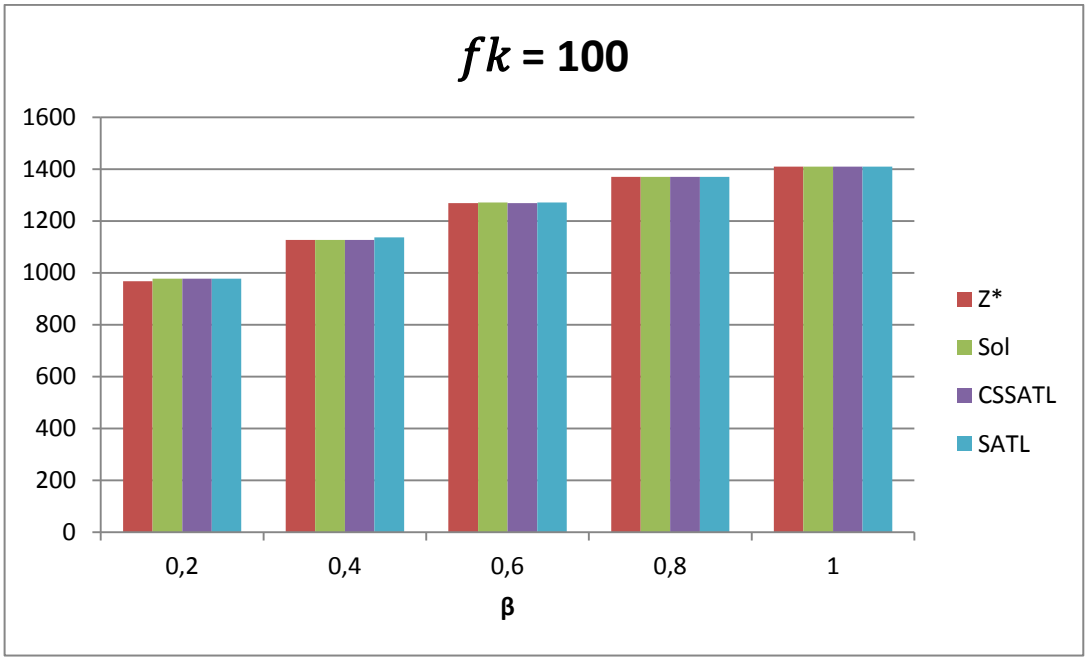


Figura 24 – GR-Learning x Almeida (2009) – Custo fixo = 100 e n = 20.

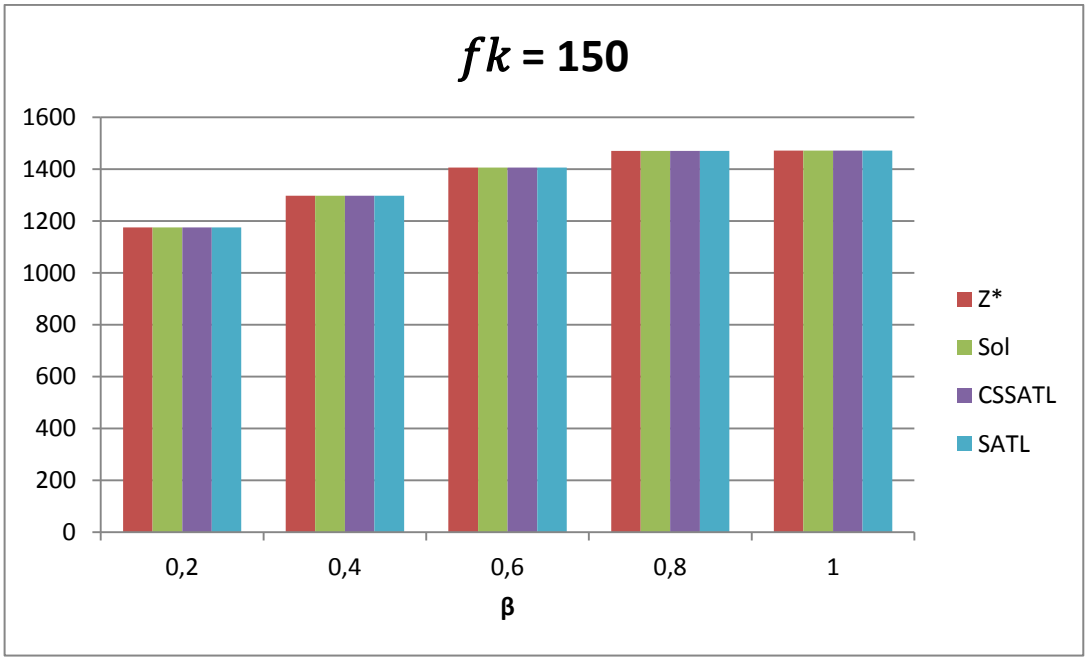


Figura 25 – GR-Learning x Almeida (2009) – Custo fixo = 150 e n = 20.

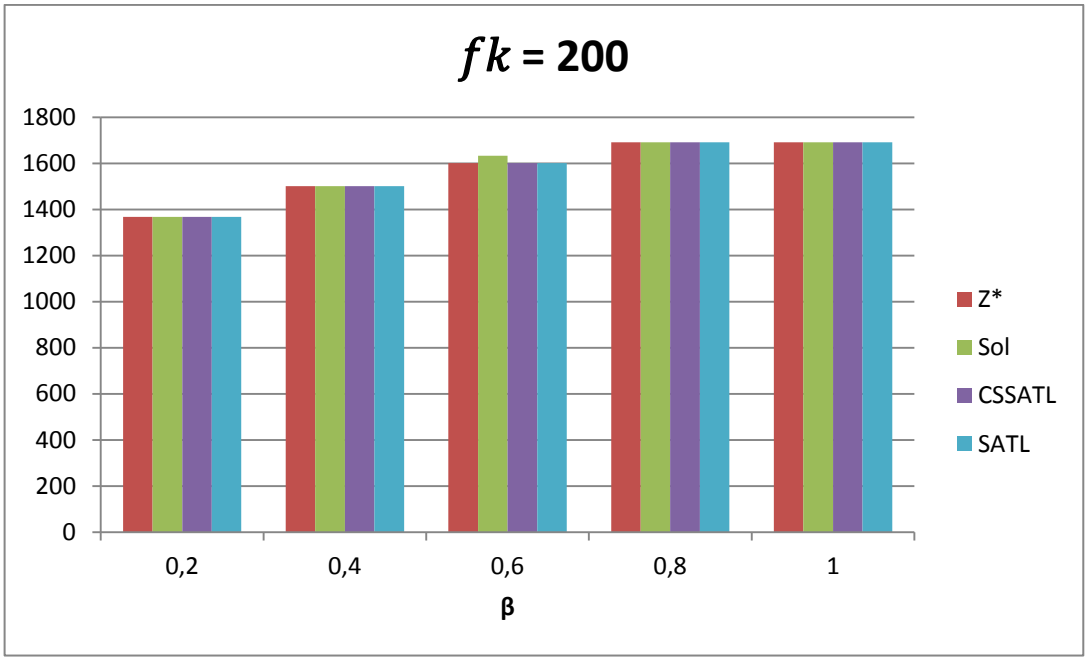


Figura 26 – GR-Learning x Almeida (2009) – Custo fixo = 200 e n = 20.

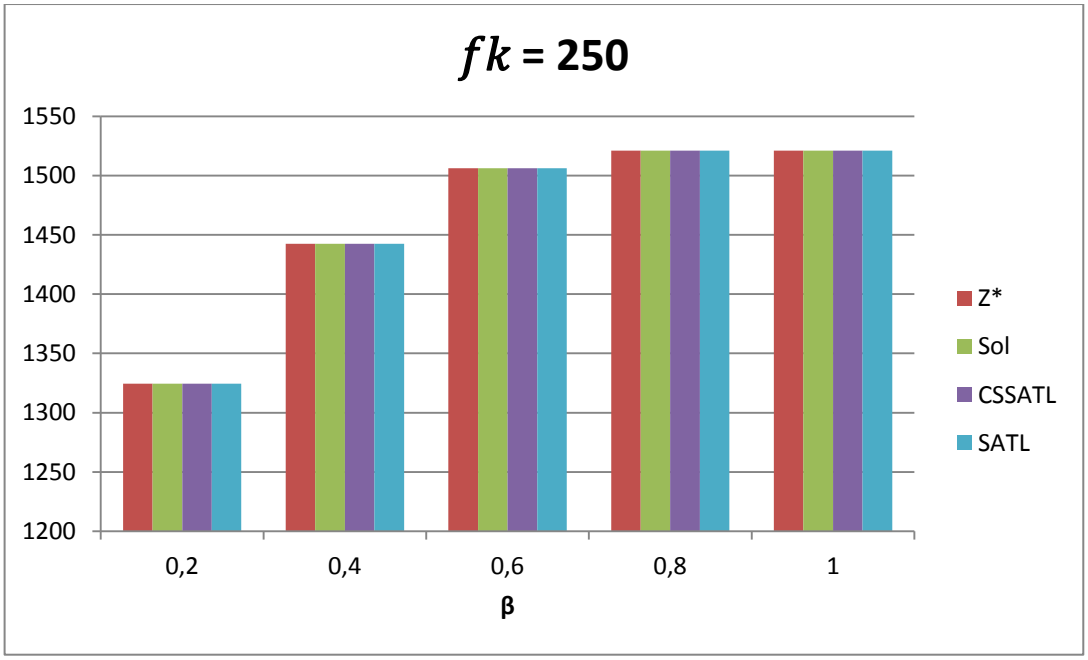


Figura 27 – GR-Learning x Almeida (2009) – Custo fixo =250 e n = 20.

Tabela 18 – GR-Learning x Almeida (2009) – Custos fixos e n = 25.

B	f_k	Z*	Sol.	Gap	Desv.	T(s)	CSSATL	GAP	Desv.	T(s)	SATL	Gap	Desv.	T(s)
0,2	100	1029,63	1029,63	0,00	0,37	2,67	1029,63	0	0	2,01	1029,63	0	0	0,23
	150	1217,34	1217,34	0,00	0,00	2,76	1217,34	0	0	0,44	1220,55	0,3	0	0,16
	200	1367,34	1367,34	0,00	0,00	2,76	1367,34	0	0	0,23	1367,34	0	0	0,15
	250	1500,9	1500,90	0,00	0,00	2,82	1500,90	0	0,34	0,22	1500,90	0	0	0,12
0,4	100	1187,51	1187,51	0,00	0,26	2,62	1187,51	0	0,42	0,27	1195,99	0,7	0,47	0,16
	150	1351,69	1351,69	0,00	0,25	2,85	1351,69	0	0	0,22	1361,24	0,7	0	0,17
	200	1501,62	1501,62	0,00	1,55	2,75	1501,62	0	0	0,16	1501,62	0	0	0,17
	250	1601,62	1601,62	0,00	1,45	2,75	1601,62	0	0,37	0,38	1601,62	0	0,37	0,1
0,6	100	1333,56	1333,56	0,00	0,41	2,71	1333,99	0	0	0,27	1341,87	0,6	0	0,15
	150	1483,56	1483,56	0,00	0,37	2,71	1483,99	0	0,75	1,51	1491,87	0,6	0,75	0,12
	200	1601,20	1633,56	2,02	0,33	2,71	1601,20	0	0	0,19	1601,20	0	0	0,08
	250	1701,20	1701,20	0,00	1,84	2,73	1701,20	0	0	0,15	1701,20	0	0	0,08
0,8	**100**	**1458,83**	**1462,06**	**0,22**	**0,96**	**2,72**	**1458,83**	**0**	**0,44**	**0,33**	**1458,83**	**0**	**0,46**	**0,2**
	150	**1594,08**	**1597,51**	**0,22**	**1,01**	**2,74**	**1594,08**	**0**	**0**	**0,24**	**1597,51**	**0,2**	**0**	**0,09**
	200	1690,57	1690,57	0,00	0,00	2,71	1690,57	0	0	0,16	1690,57	0	0	0,09
	250	1740,57	1740,57	0,00	0,00	2,71	1740,57	0	0	0,12	1740,57	0	0	0,08
1	**100**	**1556,63**	**1556,63**	**0,00**	**3,37**	**2,69**	**1559,19**	**0,2**	**0**	**0,29**	**1559,19**	**0,2**	**0**	**0,09**
	150	1640,57	1640,57	0,00	0,00	2,65	1640,57	0	0	0,14	1640,57	0	0	0,09
	200	1690,57	1690,57	0,00	0,00	2,65	1690,57	0	0	0,10	1690,57	0	0	0,08
	250	1740,57	1740,57	0,00	0,00	2,65	1740,57	0	0	0,19	1740,57	0	0	0,08

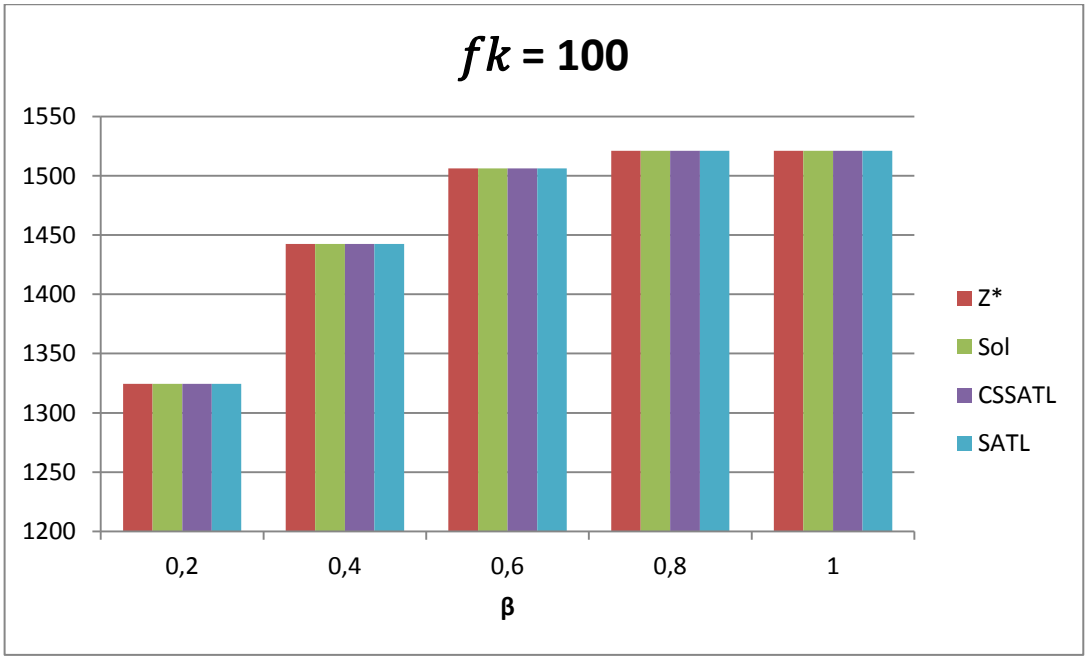


Figura 28 – GR-Learning x Almeida (2009) – Custos fixo = 100 e n = 25.

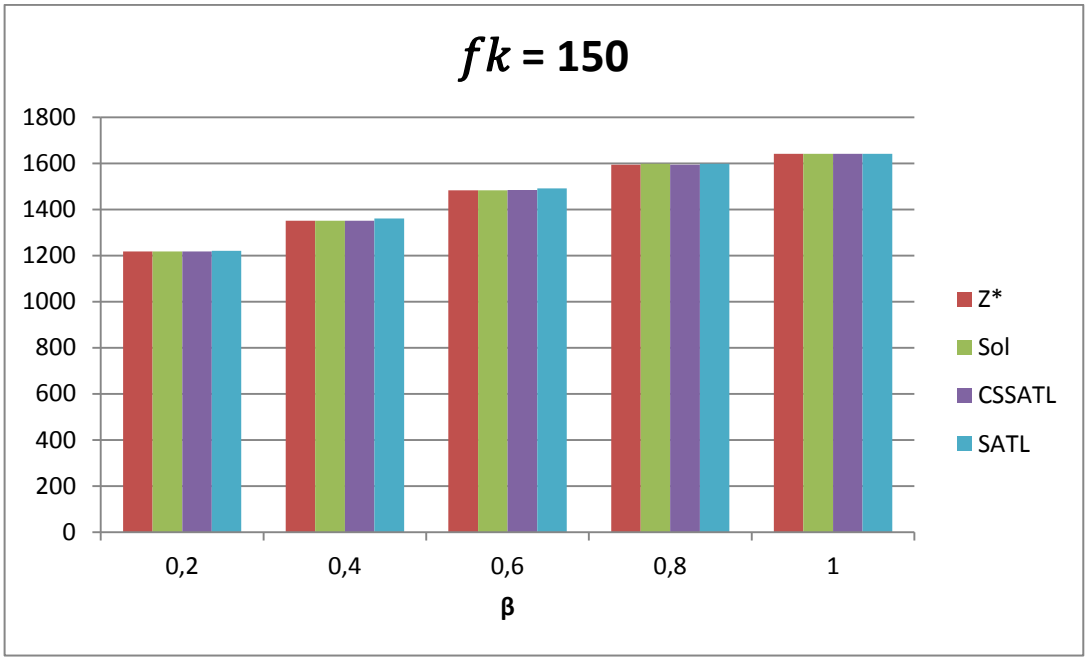


Figura 29 – GR-Learning x Almeida (2009) – Custo fixo = 150 e n = 25.

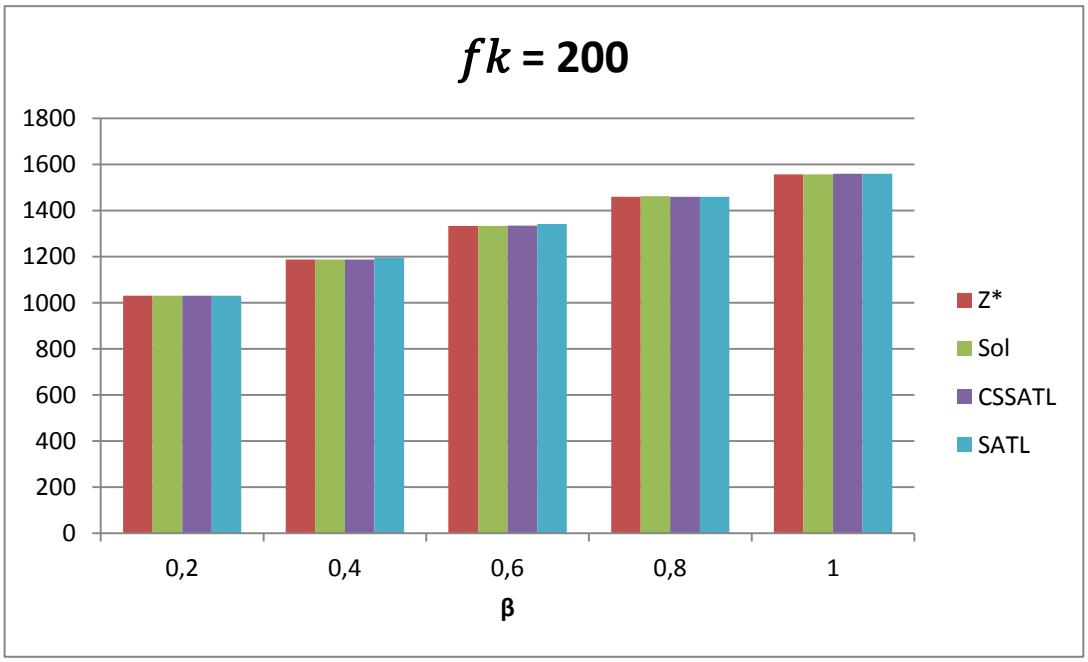


Figura 30 – GR-Learning x Almeida (2009) – Custo fixo = 200 e n = 25.

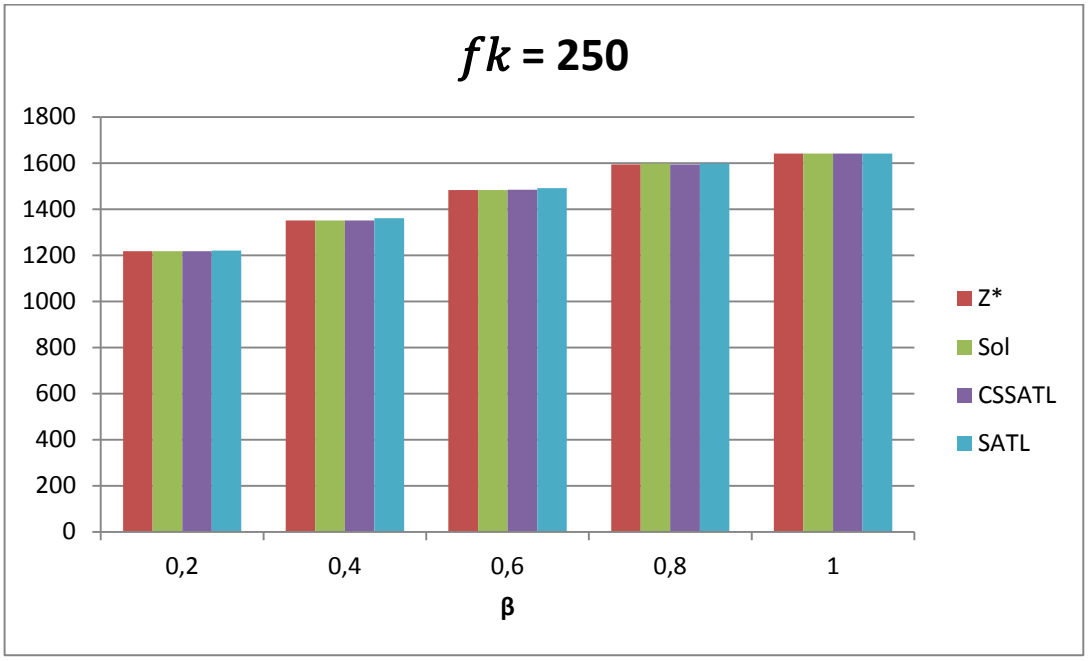


Figura 31 – GR-Learning x Almeida (2009) – Custo fixo = 250 e n = 25.

As Tabelas 17 e 18, juntamente com os gráficos das Figuras 24 a 31, mostram que o algoritmo proposto obteve um comportamento bem semelhante a outros métodos já utilizados na literatura, apesar de algumas poucas instâncias terem obtido um resultado inferior em relação a função objetivo, e em outras em relação ao tempo, pode-se considerar que o algoritmo proposto tem um desempenho satisfatório pela qualidade das soluções obtidas em um tempo razoável.

7.1.4. Resultados computacionais para a instância BR2010

A Tabela 19 apresenta os resultados para a instância BR2010, foram efetuados testes com configurações de 2 a 10 *hubs*, a coluna N. representa a quantidade de *hubs* de 2 a 10 as colunas Média, Maior e Sol. apresentam os valores para a função objetivo após trinta execuções do algoritmo com mil iterações em cada execução, sendo que a coluna Sol. apresenta a melhor solução obtida, a coluna T(s) traz o tempo medido em segundos, a coluna Hubs mostra os aeroportos alocados como hubs e a coluna código ICAO apresenta o código da Organização Civil da Aviação Internacional (do inglês *International Civil Aviation Organization - ICAO*).

Tabela 19 – Resultados para instância BR2010

N.	Média	Maior	Sol.	Desv.	T(s)	Hubs	Código ICAO
2	501,85	506,11	497,11	0,95	189,17	36, 56	SBBR, SNPY*
3	495,01	510,78	489,83	1,06	255,11	36, 53, 136	SBBR, SBMK*, SBKP
4	485,14	497,13	479,94	1,08	245,19	36, 51, 58, 136	SBBR, SNAP*, SBUL, SBKP
5	481,02	493,91	470,41	2,26	245,80	26, 36, 66, 56, 140	SNGI*, SBBR, SJHG*, SNPY, SDSC*
6	465,22	474,96	457,44	1,70	261,07	31, 38, 40, 55, 66, 136	SBSV, SBCN*, SBMC*, SNJR*, SJHG, SBKP
7	459,11	475,23	446,49	2,83	258,70	28, 36, 54, 68, 85, 133, 144	SBLE*, SBBR, SNPD*, SIZX*, SNFX*, SBAU*, SBGR
8	456,98	468,3	431,93	5,80	255,18	28, 36, 46, 56, 66, 68, 138, 144	SBLE*, SBBR, SBCF, SNPY*, SJHG*, SIZX*, SBDN*, SBGR
9	450,55	466,66	438,18	2,82	261,50	28, 36, 39, 55, 72, 83, 133, 139, 144	SBLE*, SBBR, SBGO, SNJR*, SWSI*, SNKE*, SBAU*, SBRP*, SBGR
10	443,74	452,48	431,53	2,83	257,54	28, 36, 46, 54, 64, 67, 82, 98, 134, 144	SBLE*, SBBR, SBCF, SNPD*, SBAT*, SBCY, SNDC*, SBLO, SBBT*, SBGR

***Os itens destacados não atendem as restrições para atuar como Hub**

Para os primeiros testes assumiu-se que qualquer aeroporto poderia ser alocado como hub, porém em Siqueira (2008) o autor apresenta algumas características que servem para determinar se um aeroporto pode operar como hub, dentre as características listadas a única que seria possível mensurar com os dados públicos disponíveis é a quantidade de PAX (Passageiros Transportados por Ano), aplicando essa classificação alguns dos hubs escolhidos não atendem essa classificação. No apêndice a Tabela A1 apresenta a lista com todos os aeroportos utilizados na para criação da instância e na Tabela A2 é apresentado os aeroportos que atendem a restrição conforme a Tabela 20.

Tabela 20 – Classificação de Hubs (SIQUEIRA, 2008).

Tipo	**N. de Passageiros**
Grande	mais do que 7.102.993 passageiros por ano
Médio	de 1.775.747 a 7.102.992 passageiros por ano
Pequeno	de 355.150 a 1.775.747 passageiros por ano
Não Hub	menos de 355.149 passageiros por ano

Tabela 21 – Resultados para instância BR2010 atendendo restrições da Tabela 20.

N.	Média	Maior	Sol.	Desv.	T(s)	Hubs	Código ICAO
2	508,71	514,89	503,87	0,96	205,64	36, 136	SBBR, SBKP
3	497,88	509,22	492,12	1,17	231,18	36, 46, 136	SBBR, SBCF, SBKP
4	487,92	500,56	482,20	1,19	233,10	36, 39, 46, 136	SBBR, SBGO, SBCF, SBKP
5	486,51	503,5	479,66	1,43	238,16	36, 39, 46, 58, 144	SBBR, SBGO, SBCF, SBUL, SBGR
6	485,86	496,98	478,31	1,58	256,84	36, 39, 46, 58, 95, 144	SBBR, SBGO, SBCF, SBUL, SBCT, SBGR
7	480,25	496,47	468,56	2,49	260,61	33, 36, 39, 46, 58, 105, 144	SBQV, SBBR, SBGO, SBCF, SBUL, SBGL, SBGR
8	478,59	494,13	465,95	2,71	257,18	33, 36, 39, 46, 58, 136, 143, 144	SBQV, SBBR, SBGO, SBCF, SBUL, SBKP, SBSP, SBGR
9	469,21	489,37	456,27	2,84	236,30	33, 36, 39, 46, 67, 95, 105, 136, 144	SBQV, SBBR, SBGO, SBCF, SBCY, SBCT, SBGL, SBKP, SBGR
10	466,06	472,99	459,47	1,43	233,19	33, 36, 39, 46, 58, 67, 95, 98, 105, 136, 144	SBQV, SBBR, SBGO, SBCF, SBUL, SBCY, SBCT, SBLO, SBGL, SBGR

Na Tabela 21 os resultados apontam para soluções de qualidade inferior quando comparadas com a Tabela 19, o aeroporto SBBR (Aeroporto Internacional de Brasília/DF) pode ser considerado como um dos principais hubs, pois o mesmo aparece em todas as nove configurações testadas, seguido de SBCF (Aeroporto Tancredo Neves – Confins/MG) e SBGO (Aeroporto Internacional de Goiânia/GO) aparecendo em sete das nove configurações e do aeroporto SBGR (Aeroporto Internacional de Guarulhos/SP) presente em seis das nove configurações testadas.

7.1.4.1. A escolha do novo hub da LATAM no Nordeste

A LATAM Airlines Brasil, anteriormente TAM Linhas Aéreas, desde 2015 estuda a possibilidade de instalar um novo hub no território Brasileiro, mas especificamente na região Nordeste, a Latam possui hoje três principais centros de operações nos seguintes aeroportos: SBGR (Aeroporto Internacional de São Paulo – Guarulhos/SP), SBBR (Aeroporto Internacional de Brasília – Brasília/DF) e SBGL (Aeroporto Internacional do Rio de Janeiro - Galeão – Rio de Janeiro/RJ) e outros centros de operações como: SBSP (Aeroporto de Congonhas – São Paulo/SP), SBCF (Aeroporto Internacional de Belo Horizonte – Confins/MG), SBCT (Aeroporto Internacional de Curitiba – Curitiba/PR), SBEG (Aeroporto Internacional de Manaus – Manaus/AM), SBPA (Aeroporto Internacional de Porto Alegre – Porto Alegre/RS),

SBFZ (Aeroporto Internacional de Fortaleza – Fortaleza/CE), SBRJ (Aeroporto Santos Dumont – Rio de Janeiro/RJ). Dentre os aeroportos listados temos SBGR, SBBR, SBSP que podem ser classificados como hub de grande porte, e SBGL, SBRJ, SBCF, SBPA como hub de médio porte.

Com base na instância criada, e com as informações acima citadas foram executados dois testes para tentar simular a escolha do novo hub da LATAM no Nordeste. Na primeira etapa foram fixados os hubs iniciais sendo os três principais centros de operações conjuntamente com cada uma das novas opções: Fortaleza, Natal e Recife.

Tabela 22 – Escolha do novo hub em relação aos principais centros de operação.

Hub	Média	Maior	Sol.	Desv.	T(s)	Hubs	Código ICAO
For.	570,62	577,93	562,49	1,46	213,92	34, 36, 105, 144	SBFZ, SBGR, SBBR, SBGL
Nat.	573,39	600,22	546,25	4,96	213,15	36, 105, 109, 144	SBGR, SBBR, SBNT, SBGL
Rec.	562,81	578,36	546,67	2,86	220,03	36, 92, 105, 144	SBGR, SBRF, SBBR, SBGL

A Tabela 22 apresenta os resultados para simulação de escolha do novo *hub*, observando a Tabela 22, o *hub* em Natal produz uma solução um pouco melhor do que a opção de Recife, porém, a média das soluções obtidas com a escolha do hub em Natal é bem mais alta que a opção em Recife, e da mesma forma o desvio padrão.

Tabela 23 – Escolha do novo hub em relação a todos os centros de operação.

Hub	Média	Maior	Sol.	Desv.	T(s)	Hubs	Código ICAO
For.	551,10	562,04	539,20	2,20	220,22	34, 36, 46, 95, 105, 107, 119, 143, 144	SBFZ, SBBR, SBCF, SBCT, SBGL, SBRJ, SBPA, SBSP, SBGR
Nat.	548,46	558,37	539,17	1,72	221,11	36, 46, 95, 105, 107, 109, 119, 143, 144	SBBR, SBCF, SBCT, SBGL, SBRJ, SBNT, SBPA, SBSP, SBGR
Rec.	539,66	551,72	530,18	1,78	219,77	36, 46, 92, 95, 105, 107, 119, 143, 144	SBBR, SBCF, SBRF, SBCT, SBGL, SBRJ, SBPA, SBSP, SBGR

A Tabela 23 apresenta a segunda etapa de testes considerando agora além de os três principais centros de operações, os demais centros que são utilizados pela LATAM no Brasil. Os resultados apresentam a instalação do *hub* em Recife como sendo a opção que produz o melhor resultado.

7.2. GR-LEARNING APLICADO AO PROBLEMA DE CORTE E ESTOQUE

Os resultados apresentados daqui para frente foram obtidos após a execução do algoritmo *GR-Learning* 30 vezes, com 1000 iterações a cada execução, a coluna LxW representa o tamanho do objeto padrão, a coluna N refere-se ao número de itens diferentes, a coluna Z* apresenta o melhor valor da função objetivo conhecido na literatura, e o restante das colunas seguem o mesmo padrão das Tabelas já apresentadas.

Primeiramente serão apresentados os resultados individuais para cada conjunto de instâncias, e após isso será apresentado um quadro comparativo entre o desempenho do *GR-Learning* com valores de soluções de algoritmos competitivos apresentados em Morabito e Pureza (2007) e referenciados como se segue:

• **CW77** (CHRISTOFIDES e WHITLOCK, 1977): Algoritmo de busca em árvore com programação dinâmica e uma rotina de transporte – computador CDC 7600.

• **W83** (WANG, 1983): Algoritmo 1 com β= 0,01, 0,02 e 0,03, respectivamente – computador Univac 1100.

• **OF90** (OLIVEIRA e FERREIRA, 1990): Variação do algoritmo 1 de Wang (1983) – microcomputador 80286/7.

• **VB93** (VISWANATHAN e BAGCHI, 1993): Algoritmo de busca best-first– computador VAX 11.

• **CH95** (CHRISTOFIDES e HADJICONSTANTINOU, 1995): Algoritmo de busca em árvore com relaxação do espaço de estados de uma formulação de programação dinâmica – microcomputador IBM 486.

• **H97** (HIFI, 1997): Variação do algoritmo de Viswanathan e Bagchi – computador Data General AV 8000.

• **FHZ98** (FAYARD et al., 1998): Algoritmo geral de corte da melhor faixa, explorando a abordagem em dois estágios de Gilmore and Gomory – computador Sparc 20.

• **APT02** (ALVAREZ-VALDÉS et al., 2002): Algoritmo de busca tabu – microcomputador Pentium II.

• **MP e MPAOG** (MORABITO E PUREZA, 2007): Algoritmo de programação dinâmica e busca em grafo e/ou – microcomputador Pentium IV.

7.2.1.1. Resultados para o primeiro conjunto de instâncias

Tabela 24 – Resultados para o primeiro conjunto de instâncias da literatura.

Instância	LxW	N	Z*	Sol.	Gap.	Média	Menor	Desv.	T(s)
CW1	15x10	7	244	260*	-6,56	260	260	0,00	0,63
CW2	40x70	10	2892	2858	1,18	2822,9	2805	1,23	9,8
CW3	40x70	20	1860	1880*	-1,08	1814,66	1780	3,48	2,86
OF1	70x40	10	2737	2724	0,47	2713,36	2713	0,39	10,86**
OF2	70x40	10	2717	2700	0,63	2686,96	2669	0,48	1,23
WANG1	69x33	20	2277	2277	0,00	2277	2277	0,00	3
WANG2	70x39	20	2694	2716*	-0,82	2716	2716	0,00	3,4
WANG3	70x40	20	2721	2754*	-1,21	2754	2754	0,00	3

***Novos valores para função objetivo** ****Resultados obtidos com 10000 iterações**

Os resultados apresentados na Tabela 24 possuem excelente qualidade, uma vez que foi encontrado o melhor valor conhecido na literatura em diversas instâncias e onde não foi possível encontrar o melhor valor o maior GAP apresentado está próximo a 1% e com desvio padrão com valores relativamente baixo. Temos um destaque para o resultado das instâncias CW1, CW3, Wang2 e Wang3 que foi possível encontrar um novo valor para a função objetivo.

Tabela 25 – Comparativo com outros resultados da literatura.

Instância	LxW	N	Z*	Sol.	CW77	W83	OF90	VB93	CH95	H97	APT02	MP	MPAOG
CW1	15x10	7	244	260	244			244	244	244		244	244
T(s)				0,63	2,5			1,6	120	< 1		0,6	< 1
CW2	40x70	10	2768	2858	2892			2892	2892	2892	2892	2768	2892
T(s)				9,8	24,61			12	4236	7,1	N/D	242,5	38,4
CW3	40x70	20	1860	1880	1860			1860	1860	1860	1860	1860	1860
T(s)				2,86	66,1			69,8	3912	29,8	N/D	93,8	19,9
OF1	70x40	10	2737	2724			2737					2737	2737
T(s)				10,86			N/D					106,5	11,2
OF2	70x40	10	2717	2700			2690					2690	2690
T(s)				1,23			N/D					28,5	23,6
WANG1	69x33	20	2277	2277		2277						2277	2277
T(s)				3		23						0,1	0,1
WANG2	70x39	20	2694	2716		2694						2694	2694
T(s)				3,4		34						0,2	0,2
WANG3	70x40	20	2721	2754		2721		2721				2721	2721
T(s)				3		73		180				2,8	0,6

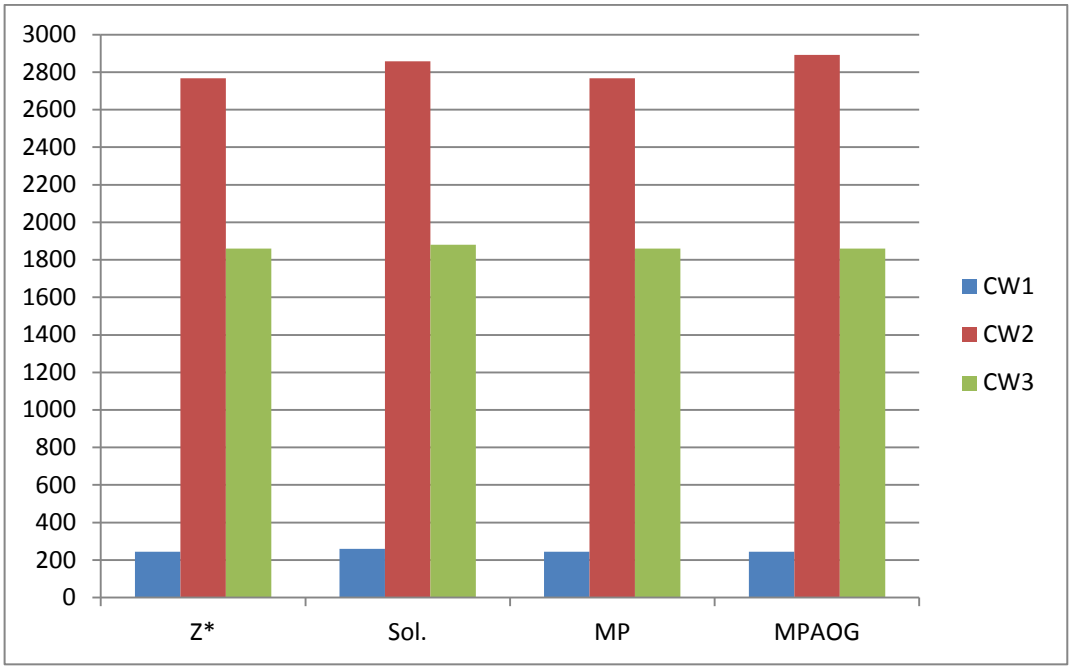


Figura 32 – Comparativo entre o GR-Learning e Morabito e Pureza (2007).

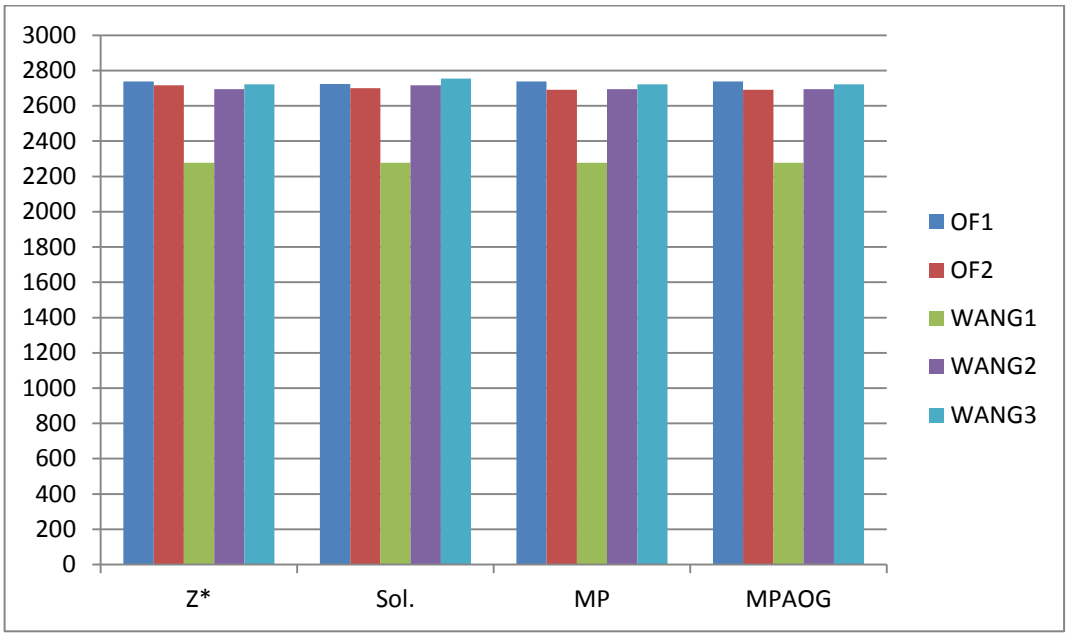
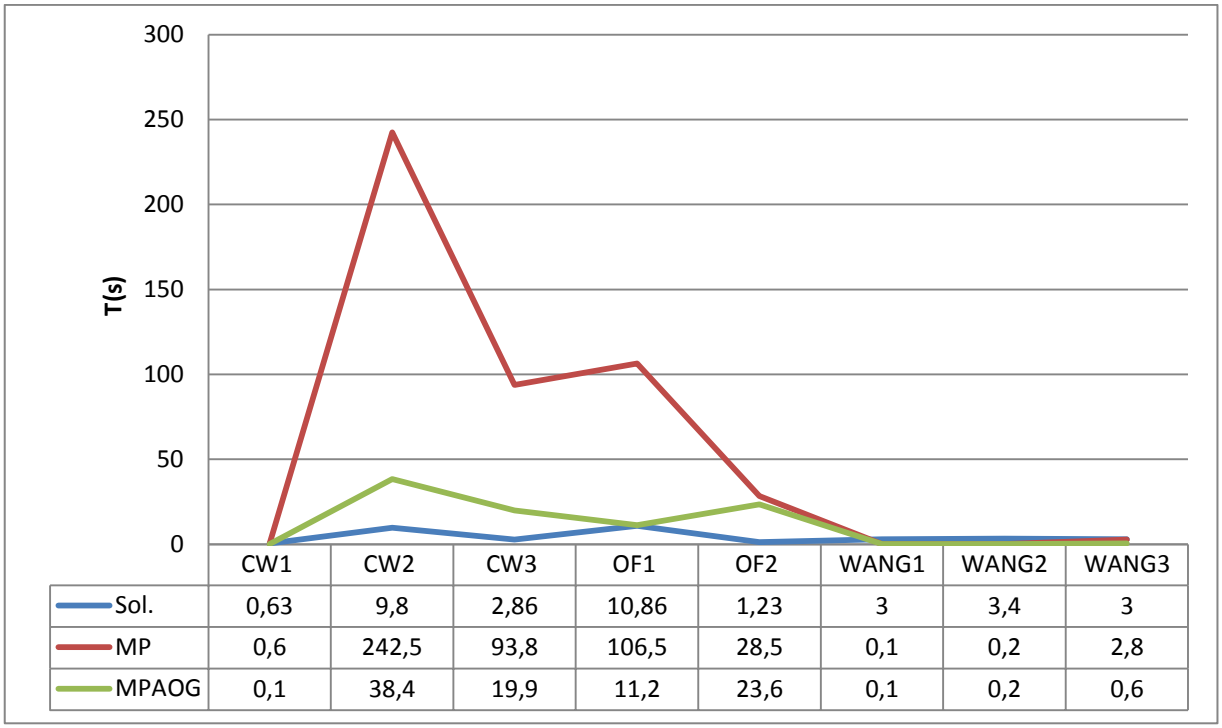


Figura 33 – Comparativo entre o GR-Learning e Morabito e Pureza (2007).



	CW1	CW2	CW3	OF1	OF2	WANG1	WANG2	WANG3
Sol.	0,63	9,8	2,86	10,86	1,23	3	3,4	3
MP	0,6	242,5	93,8	106,5	28,5	0,1	0,2	2,8
MPAOG	0,1	38,4	19,9	11,2	23,6	0,1	0,2	0,6

Figura 34 – Comparativo do tempo computacional entre o GR-Learning e Morabito e Pureza (2007).

A Tabela 25 apresenta um quadro comparativo entre o *GR-Learning* e outros métodos já utilizados na literatura. A Figura 34, apresenta um gráfico comparativo entre o *GR-Learning* e os algoritmos propostos e Morabito e Pureza (2007), e como já mencionado o algoritmo encontrou novos valores para as instâncias CW1, CW3, Wang2, Wang 3 e foi melhor que os outros algoritmos na instância OF2. Em relação ao tempo computacional o *GR-Learning* foi melhor em diversas ocasiões e o mesmo mantém um comportamento mais equilibrado em relação aos outros algoritmos apresentados.

7.2.1.2. Resultados para o segundo conjunto de instâncias

Na Tabela 26 serão apresentados os resultados obtidos para o segundo conjunto de instâncias.

Tabela 26 – Resultados para o segundo conjunto de instâncias da literatura.

Instância	L x W	N	Z*	Sol.	Gap.	Média	Menor	Desv.	T(s)
CU1	**100x125**	**25**	12330	12500*	-1,38	12478,46	12309	0,17	7,43
CU2	**150x175**	**35**	26100	26100	0,00	25864,00	25650	0,90	6,36
CU3	**134x125**	**45**	16723	16637	0,51	16554,93	16446	0,49	32,00
CU4	**285x354**	**45**	99945	99616	0,33	98837,00	97662	0,78	22,3
CU5	**456x385**	**50**	173364	173010	0,20	172169,00	171014	0,49	36,03
CU6	**356x447**	**45**	158572	158572	0,00	157934,93	157764	0,40	128,73

CU7	**563x458**	**45**	247150	252300*	-2,08	250951,00	247230	0,53	5,6
CU8	**587x756**	**35**	433331	439145*	-1,34	434745,00	430299	1,00	12,33
CU9	**856x785**	**25**	657055	658602*	-0,24	657209,00	657055	0,21	9,23
CU10	**794x985**	**40**	773772	772719	0,14	766824,00	762660	0,76	22,8
CU11	**977x953**	**50**	924696	919260	0,59	914559,00	911244	0,51	128,83

***Novos valores para função objetivo**

As soluções obtidas para o segundo conjunto de instâncias apresentaram novamente excelentes resultados, conseguindo obter o valor ótimo em diversas instâncias e apresentando novos valores paras as instâncias CU1, CU7, CU8 e CU9. E nas instâncias em que obteve valores abaixo do valor ótimo o GAP não foi superior a 0,6%. A seguir o quadro comparativo com valores de outros algoritmos será mostrado.

Tabela 27 – Comparativo com outros resultados da literatura.

Instância	L x W	N	Z*	Sol.	FHZ98	APT02	MP	MPAOG
CU1	**100x125**	**25**	12330	12500*	12312	12330	12330	12330
T(s)				7,43	7,1	N/D	0,4	0,5
CU2	**150x175**	**35**	26100	26100	25806	26100	26100	26100
T(s)				6,36	19,8	N/D	1,2	1,4
CU3	**134x125**	**45**	16723	16637	16608	16679	16723	16723
T(s)				32	30,5	N/D	45,9	5,8
CU4	**285x354**	**45**	99945	99616	98190	99366	99945	99945
T(s)				22,3	55,8	N/D	105,9	35
CU5	**456x385**	**50**	173364	173010	171651	173364	173364	173364
T(s)				36,03	89,6	N/D	768,2	873,2
CU6	**356x447**	**45**	158572	158572	158572	158572	158572	158572
T(s)				54,45	29,8	N/D	10,3	11,8
CU7	**563x458**	**45**	247150	252300*	246860	247150	247150	247150
T(s)				5,6	26,7	N/D	1800	1800
CU8	**587x756**	**35**	433331	439145*	432198	432714	433331	433331
T(s)				12,33	68,8	N/D	20,6	26,8
CU9	**856x785**	**25**	657055	658602*	657055	657055	657055	657055
T(s)				9,23	47,2	N/D	19,7	19,7
CU10	**794x985**	**40**	773772	772719	764696	773485	773772	773772
T(s)				22,8	123,9	N/D	1800	1800
CU11	**977x953**	**50**	924696	919260	913387	922161	924696	924696
T(s)				128,83	219,6	N/D	1800	1293,1

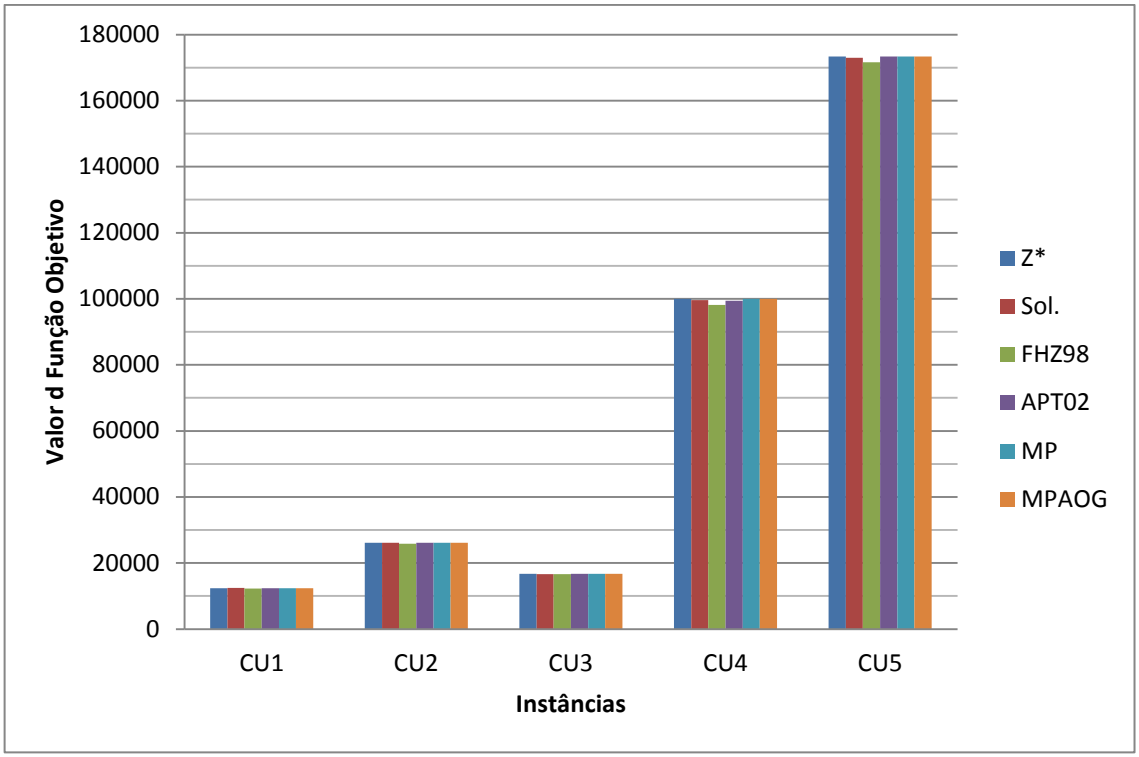


Figura 35 – Comparativo com outros resultados da literatura. (Instâncias CU1 a CU5).

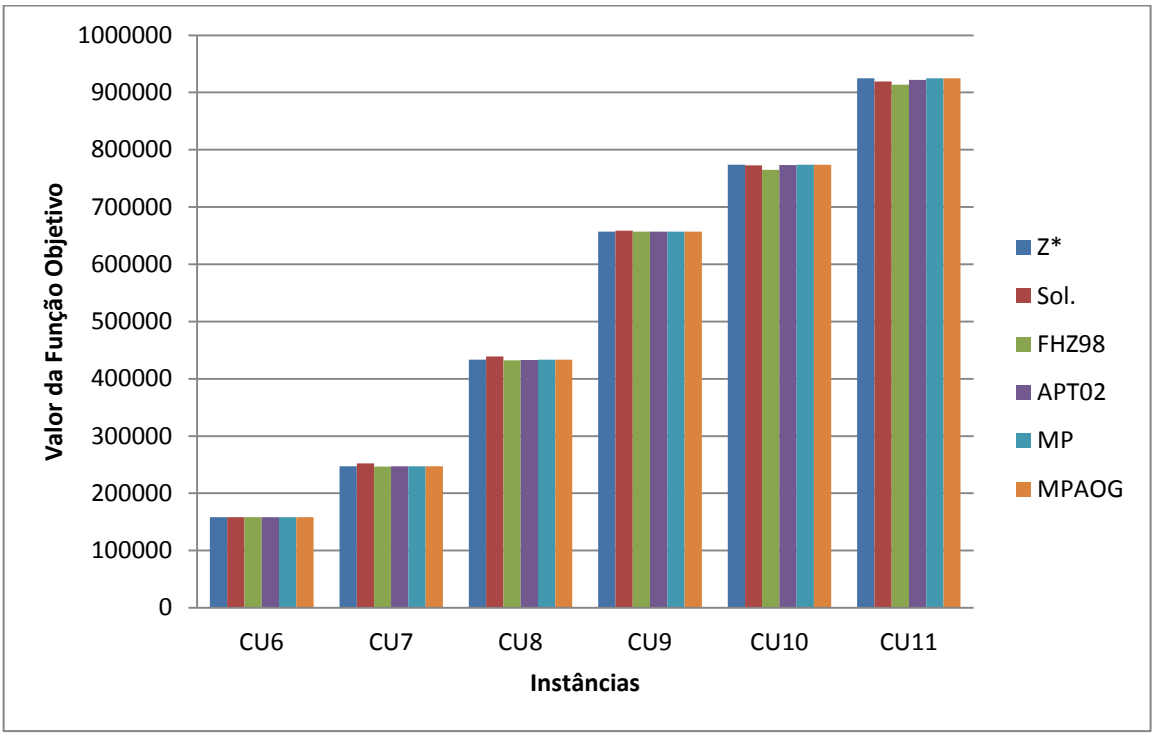
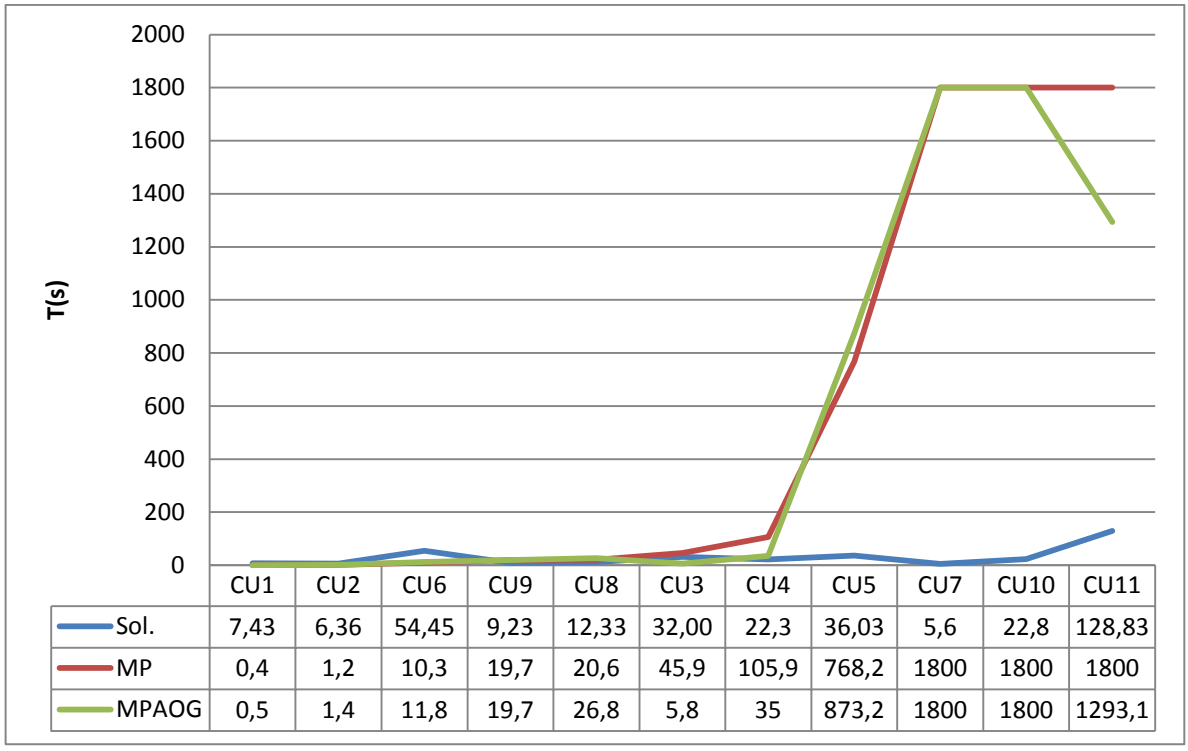


Figura 36 – Comparativo com outros resultados da literatura. (Instâncias CU6 a CU11).



	CU1	CU2	CU6	CU9	CU8	CU3	CU4	CU5	CU7	CU10	CU11
Sol.	7,43	6,36	54,45	9,23	12,33	32,00	22,3	36,03	5,6	22,8	128,83
MP	0,4	1,2	10,3	19,7	20,6	45,9	105,9	768,2	1800	1800	1800
MPAOG	0,5	1,4	11,8	19,7	26,8	5,8	35	873,2	1800	1800	1293,1

Figura 37 – Comparativo do tempo computacional entre o GR-Learning e Morabito e Pureza (2007).

Mais uma vez o GR-Learning se mostrou bastante competitivo frente a outros algoritmos, conseguindo superar o valor da função objetivo em quatro das onze instâncias, e nas instâncias que não foi superior conseguiu um valor muito próximo com um tempo de processamento menor, porém o que chama atenção em relação ao tempo não é o fato de ser menor, pois existe uma diferença considerável em relação às máquinas utilizadas, e sim no comportamento apresentado pela linha do gráfico do tempo apresentado na Figura 37.

7.2.1.3. Resultados para o terceiro conjunto de instâncias

O terceiro conjunto apresenta instâncias com até 60 itens diferentes, na Tabela a seguir serão apresentados os resultados obtidos.

Tabela 28 – Resultados para o terceiro conjunto de instâncias da literatura.

Instância	L x W	N	Z*	Sol.	Gap.	Média	Menor	Desv.	T(s)
CW1	**125x105**	**25**	6402	6402	0	6656,76	6212	3,97	3,00
CW2	**145x165**	**35**	5354	5548*	-3,62	5373,53	5238	3,14	4,37
CW3	**267x207**	**40**	5689	5657	0,56	5479,76	5202	3,13	15,87
CW4	**465x387**	**39**	6175	7466*	-20,91	7304	6902	2,17	15,17
CW5	**524x678**	**35**	11659	11659	0,00	11659	11659	0,00	14,60
CW6	**781x657**	**55**	12923	12635	2,23	11792,7	11022	6,67	40,30

CW7	**276x374**	**45**	9898	10601*	-7,10	9935,16	9472	6,28	24,50
CW8	**305x287**	**60**	4605	4621*	-0,35	4463,7	4146	3,40	48,37
CW9	**405x362**	**50**	10748	10501	2,30	10136,46	9513	3,47	27,63
CW10	**992x970**	**60**	6515	6438	1,18	6242,1	5964	3,04	21,83
CW11	**982x967**	**60**	6321	6318	0,05	6078,3	5717	3,79	41,87

***Novos valores para função objetivo**

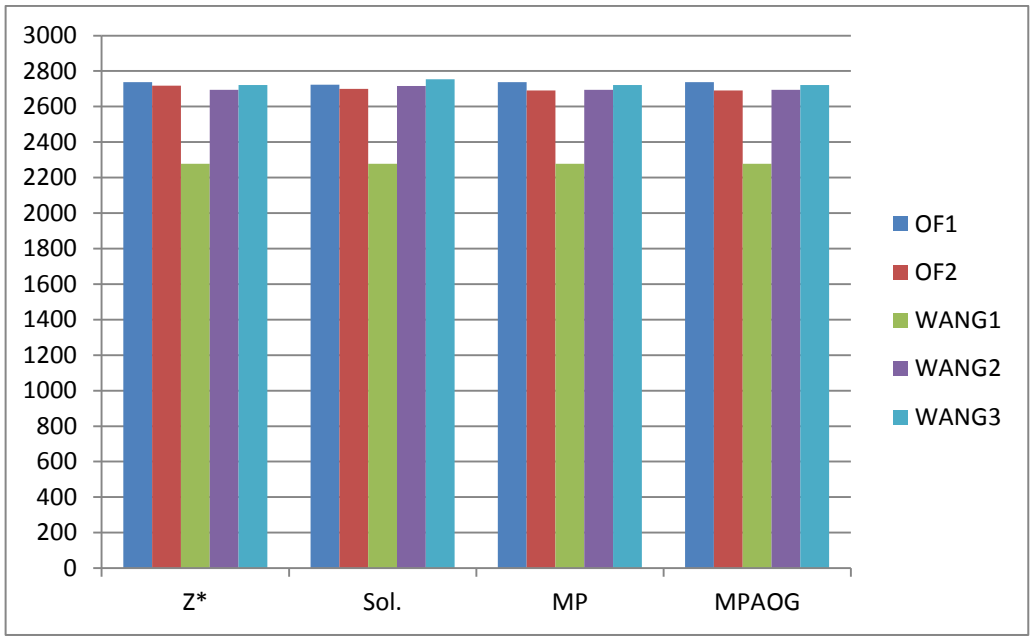


Figura 38 – Comparativo com outros resultados da literatura.

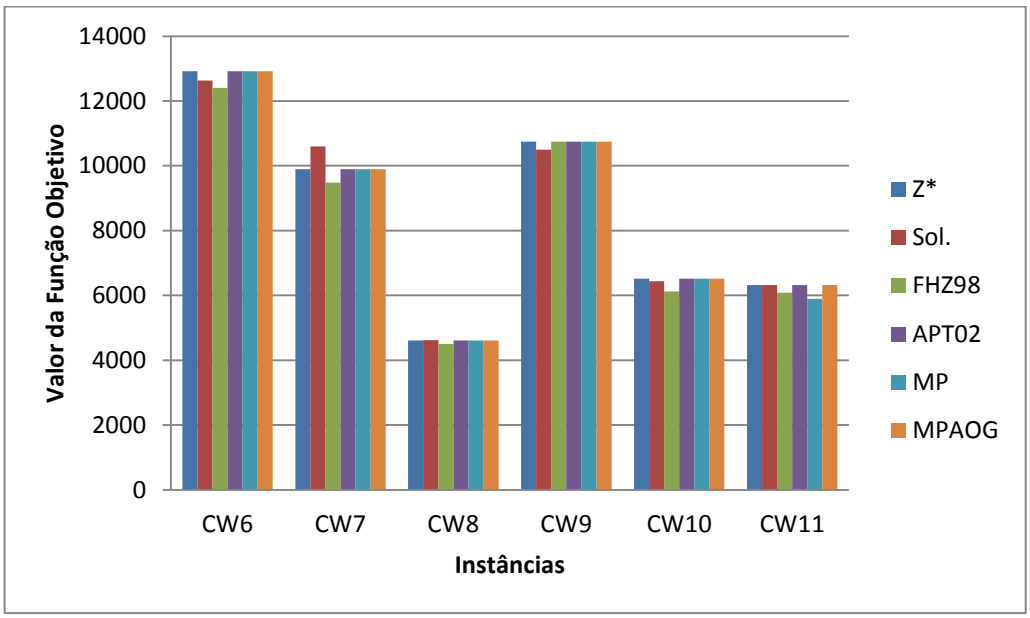


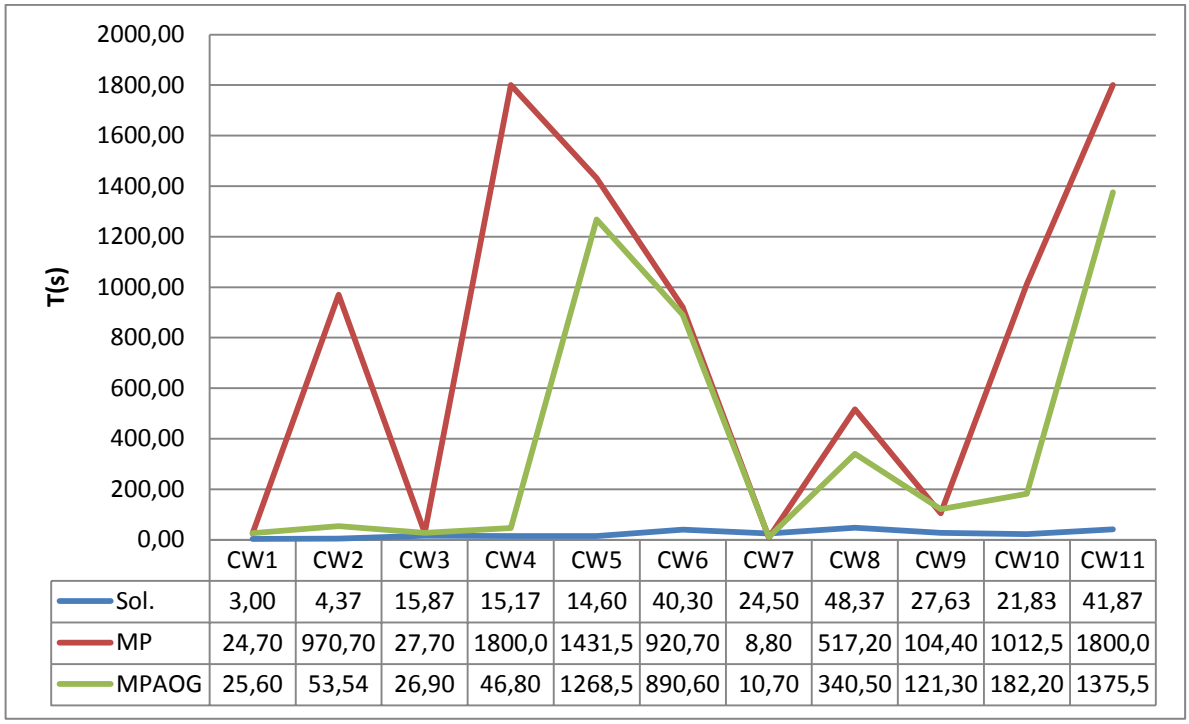
Figura 39 – Comparativo com outros resultados da literatura.

Os resultados apresentados na Tabela 28 mostram que mesmo com instâncias maiores (60 itens) o algoritmo mantem o seu desempenho, permanecendo com excelentes soluções

com tempos computacionais relativamente baixos, e mais uma vez apresentou novos valores para função objetivo conforme pode ser visto nas instâncias CW2, CW4, CW7 e CW8. A seguir temos o quadro comparativo com resultados de outros algoritmos.

Tabela 29 – Comparativo com outros resultados da literatura.

Instância	L x W	N	Z*	Sol.	FHZ98	APT02	MP	MPAOG
CW1	**125x105**	**25**	6402	6402	6402	6402	6402	6402
T(s)				3,00	5,30	N/D	24,70	25,60
CW2	**145x165**	**35**	5354	5548	5354	5354	5281	5354
T(s)				4,37	6,80	N/D	970,70	1118,80
CW3	**267x207**	**40**	5689	5657	5148	5689	5689	5689
T(s)				15,87	17,30	N/D	27,70	26,90
CW4	**465x387**	**39**	6175	7466	6168	6170	6170	6175
T(s)				15,17	30,80	N/D	1800,00	46,80
CW5	**524x678**	**35**	11659	11659	11550	11644	11659	11659
T(s)				14,60	19,80	N/D	1431,50	1268,50
CW6	**781x657**	**55**	12923	12635	12403	12923	12923	12923
T(s)				40,30	79,00	N/D	920,70	890,60
CW7	**276x374**	**45**	9898	10601	9484	9898	9898	9898
T(s)				24,50	61,40	N/D	8,80	10,70
CW8	**305x287**	**60**	4605	4621	4504	4605	4605	4605
T(s)				48,37	107,40	N/D	517,20	340,50
CW9	**405x362**	**50**	10748	10501	10748	10748	10748	10748
T(s)				27,63	125,30	N/D	104,40	121,30
CW10	**992x970**	**60**	6515	6438	6116	6515	6515	6515
T(s)				21,83	281,10	N/D	1012,50	182,20
CW11	**982x967**	**60**	6321	6318	6084	6321	5897	6321
T(s)				41,87	197,50	N/D	1800,00	1375,50



	CW1	CW2	CW3	CW4	CW5	CW6	CW7	CW8	CW9	CW10	CW11
Sol.	3,00	4,37	15,87	15,17	14,60	40,30	24,50	48,37	27,63	21,83	41,87
MP	24,70	970,70	27,70	1800,0	1431,5	920,70	8,80	517,20	104,40	1012,5	1800,0
MPAOG	25,60	53,54	26,90	46,80	1268,5	890,60	10,70	340,50	121,30	182,20	1375,5

Figura 40 – Comparativo do tempo computacional entre o GR-Learning e Morabito e Pureza (2007).

Podemos observar que mesmo trabalhando com instâncias maiores o algoritmo proposto conseguiu excelentes resultados, sendo superior em algumas instâncias e mantendo o tempo computacional abaixo dos outros algoritmos apresentados.

CAPÍTULO 8

CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentado um estudo sobre dois problemas clássicos da pesquisa operacional: O Problema de Localização de Hubs e o Problema de Corte e Estoque. Ambos os problemas pertencentes a classe NP-Difícil, o que justifica o uso de metaheurísticas para busca da solução ótima.

Em Almeida (2014) foi criado uma metaheurística híbrida que utiliza a metaheurística *GRASP* Reativo com o uso de aprendizagem por reforço. Especificamente o algoritmo *Q-learning (GR-Learning)*, onde o *Q-learning* foi utilizado como memória adaptativa para com o passar das iterações aprender qual o melhor α deve ser utilizado durante a fase de construção da *GRASP*. O método proposto em Almeida (2014) foi utilizado para resolver o problema dos *p*-Centros para localização estratégica de viaturas policiais na cidade de Mossoró/RN, que segundo o autor o método proposto apresentou resultados superiores quando comparados com o *GRASP Reativo* tradicional.

Conforme visto no Capítulo 3 uma metaheurística deve ser modelada para cada problema específico, isso leva muitas vezes a construção de códigos fortemente acoplados ao problema ao qual foi projetado para resolver, e quando necessário aplicar a outros problemas, exige muitas vezes um grande esforço em termos de modificações são necessárias para ajustar a metaheurística ao novo problema abordado.

Com o objetivo de poder utilizar os benefícios do *GR-Learning* em diversos outros problemas de otimização combinatória. Foi proposto aqui o desenvolvimento de um *framework* que permita ao pesquisador aplicar o *GR-Learning* a outros problemas de otimização combinatória sem que seja necessário tanto esforço para adaptação do código ao novo problema abordado.

Para validação do *framework* proposto aplicamos o mesmo a Problema de Localização de p-Hubs aplicado ao setor aeroviário e também a uma variação do Problema de Corte e Estoque: O Problema de Corte Bidimensional Guilhotinado e Restrito. Os dois problemas citados são diferentes, o primeiro trata de um problema de minimizar custos e o segundo de maximizar o aproveitamento da matéria prima.

Como foi detalhada na Seção 5.1, a única condição obrigatória para que o *framework* possa ser utilizado independente do problema é sobrescrever os métodos `GeraSolução` e

`BuscaLocal`, ficando o pesquisador livre para criar quaisquer outros métodos, classes ou atributos que julgar necessário.

Foi utilizado para validar o *framework* aplicado ao *HLP* dois conjuntos de instâncias: o conjunto de instancia CAB inserido na literatura por O'Kelly (1987) possui os valores ótimos conhecidos e uma segunda instância foi criada (BR2010) com dados aeroviários do Brasil do ano de 2010, os detalhes de criação da instância foram vistos na seção 6.2.

Para validar a utilização do framework aplicado ao *CSP* foram utilizados três conjuntos de instâncias com os valores ótimos conhecidos na literatura e também utilizados em Morabito e Pureza (2002) para validação de dois métodos propostos.

Em relação aos resultados apresentados referentes a utilização do *GR-Learning* ao *HLP* estes se mostraram satisfatórios pois no primeiro conjunto de testes o algoritmo encontrou o ótimo em dez das doze configurações testadas, e no segundo conjunto de testes, com um número de 20 nodos o algoritmo encontrou o ótimo em dezoito das 20 configurações testadas, e com um número de nodos igual a 25 conseguiu encontrar o ótimo em dezessete de vinte configurações testadas, além disso quanto tem seus resultados comparados com outros trabalhos da literatura o *GR-Learning* também apresentou resultados bastante competitivos.

Para a instância BR2010 foram feitos três conjuntos de testes: o primeiro considerando que qualquer nodo poderia ser alocado como um hub, o segundo conjunto de testes foi feito uma seleção nos nodos que realmente poderiam atuar como hub de acordo com a classificação apresentada em Siqueira (2008) e definido no método de escolha dos hubs que somente eles poderiam ser alocados como hubs, o terceiro conjunto de testes foi feito uma simulação para escolha do novo hub da LATAM no Nordeste, uma vez que desde 2015 a empresa vem fazendo estudos para decidir entre Fortaleza, Natal ou Recife como novo centro de operações.

Como destaque para os testes realizados com a instância BR2010 temos o aeroporto SBBR (Aeroporto Internacional de Brasília) que apareceu em todas as configurações testadas com nodos que atendem a classificação de hubs, e o tempo computacional que mesmo aumentando o tamanho da instância em quase seis vezes o tempo de processamento permaneceu dentro de um valor aceitável.

Em relação aos resultados apresentados referentes a utilização do *GR-Learning* aplicado ao *CSP*, o método proposto apresentou excelentes resultados para os três conjuntos de instâncias utilizados, e em alguns casos encontrando novos valores em relação ao melhor valor conhecido na literatura, e quando comparado com outros algoritmos já validados, mostrou-

se extremamente competitivo conseguindo superar os algoritmos concorrentes em algumas instâncias.

Como pode ser visto nos resultados apresentados a utilização do *framework* mostrou-se satisfatório, pois em termos de esforço para adaptação a outros problemas o mesmo não exige implementação extra para utilizar os recursos do algoritmo *Q-learning*, e se tratando de resultados computacionais, mesmo o foco desse trabalho não sendo encontrar o valor ótimo, mas sim comprovar a eficácia do *framework* proposto, foi possível encontrar o ótimo para diversas instâncias, como também encontrar novos valores em relação ao valor ótimo já conhecido na literatura.

8.1. CONTRIBUIÇÕES DA DISSERTAÇÃO

O *framework* desenvolvido se mostrou bastante eficaz e provou sua robustez ao obter resultados satisfatórios nos dois problemas aplicados, como também permite ao pesquisador usufruir dos benefícios dos *GR-Learning* sem precisar executar grandes alterações para adaptar a metaheurística a outros problemas. Assim foi possível alcançar as seguintes contribuições:

- Desenvolvimento de um *framework* para utilização do *GR-Learning* em diversos problemas de otimização combinatória sem que sejam necessários grandes alterações no código.

- Criação e disponibilização de uma instância com dados aeroviários do Brasil.

- Obtenção de soluções de melhor qualidade que os melhores valores conhecidos na literatura para algumas instâncias do *CSP*.

8.2. TRABALHOS FUTUROS

Como perspectivas de trabalhos futuros decorrentes desta pesquisa podemos citar:

- Aplicação do *framework* proposto a outros problemas de otimização combinatória;

- Utilização de um método exato para resolução da instância BR2010 e comparação do desempenho do framework utilizado;

- Verificar a possibilidade de adaptar o framework proposto a outras metaheurísticas.

REFERÊNCIAS

ABYAZI-SANI, R.; GHANBARI, R. An efficient tabu search for solving the uncapacitated single allocation hub location problem. Computers & Industrial Engineering, 2016.

ADIBI, A.; JAFAR, R. 2-Stage stochastic programming approach for hub location problem under uncertainty: A case study of air network of Iran. Journal of Air Transport Management, 2015.

ALMEIDA, C. R. Aeroportos, HUB, Spoke e Bases Operacionais. Revisão de Conceitos. CONPEHT Brasil 2012, Rio de Janeiro, RJ, 2012.

ALMEIDA, I. I. Metaheurística Híbrida Utilizando GRASP Reativo e Aprendizagem Por Reforço: Uma Aplicação na Segurança Pública. Dissertação de Mestrado – Universidade Estadual do Rio Grande do Norte – UERN, Mossoró, RN, 2014.

ALMEIDA, W. G. Métodos Heurísticos para o Problema de Localização de Concentradores. Dissertação (Mestrado) – Instituto Nacional de Pesquisas Espaciais – INPE, São José dos Campos, SP, 2009.

ALUMUR, S.; KARA, Y. B. Network hub location problems: The state of the art. European Journal of Operational Research, Ankara, Turkey, 2008.

ALZAMORA, Guina Sotomayor. Estratégias de Localização de Hubs para o Sistema de Transporte Aéreo Brasleiro. Tese de Doutorado. Pontifícia Universidade Católica do Rio de Janeiro – PUC. Rio de Janeiro, RJ, 2013.

AN, Y.; ZHANG, Y.; ZENG, B. The reliable hub-and-spoke design problem: Models and algorithms. Transportation Research Part B, 2015.

ANAC. Anuário do Transporte Aéreo 2010. 2011. Internet. www.anac.gov.br.

BALDO, Tâmara Angélica. Estudo de Heurísticas para o Problema de Corte e Estoque com Aproveitamento de Sobras. Universidade Estadual de Londrina, Londrina, PR, 2006.

BAILEY, A.; ORNBUKI-BERRNAN, B.; ASOBIELA, S. Discrete PSO for the uncapacitated single allocation hub location problem. Computational intelligence in production and logistics systems (CIPLS), IEEE workshop, 2013.

BECCENARI, José Carlos; Meta-heurísticas e Otimização Combinatória: Aplicações em Problemas Ambientais. Instituto Nacional de Pesquisas Espaciais – INPE, São José dos Campos, SP, 2012.

CAMPBELL, J.F. Integer programming formulations of discrete hub location problems. European Journal of Operational Research,, n. 72, 1994.

CHRISTOFIDES, N.; WHITLOCK, C. An Algorithm for Two-dimensional Cutting Problems. Operations Research, n. 25, 1977.

CUNHA, C. B.; SILVA, M. R. A genetic algorithm for the problem of configuring a hub-and-spoke network for a LTL trucking company in Brasil. European Journal of Operational Research, 2007.

CETINER, S., SEPIL, C., SURAL, H., Hubbing and routing in postal delivery systems. Technical Report, Industrial Engineering Department, Ankara, Turkey, 2006.

COSTA, T. F. G.; LOHMANN, G.; OLIVEIRA, A. V. M. A model to identify airport hubs and their importance to tourism in Brazil. Research in Transportation Economics, 2010.

CHERRY, Adriana Cristina. O Problema de Corte de Estoque com Reaproveitamento de Sobras de Material. Dissertação de Mestrado – ICMSP – USP – São Paulo, SP, 2006.

ÇIFTÇI, M. E.; MEHMET, S. A new hub and spoke system proposal: A case study for Turkey's. Journal of Air Transport Management, Turkey, 2015.

DYCKHOFF, H. A typology of cutting and packing problems. European Journal of Operational Research, 44, 1990.

DUSBERGER, Frederico; Raidl, Gunther R. Solving the 3-Staged 2-Dimensional Cutting Stock Problem by Dynamic Programming and Variable Neighborhood Search. Electronic Notes in Discrete Mathematics, n. 47, 2015.

ERNST, A.; MOHAN, K. Efficient Algorithms For The Uncapacited Single Allocation p-Hub Median Problem, Austrália, 1996.

FEO, T. A.; RESENDE, M. G. C. A Probabilistic Heuristic for a Computational Difficult Set Covering Problem. Operations Research Letter, 1989.

FEO, T. A.; RESENDE, M. G. C. Greedy Randomized Adaptative Search Procedure. J. of Global Optimization. 1995.

FARAHANI, R. Z. et al. Hub location problems: A review of models, classification, solution, techniques, and applications. Computers & Industrial Engineering, 2013.

FIGUEIREDO, R. M. A., Um estudo de Localização de Hubs no Transporte Aéreo de Cargas Brasileiro. Dissertação de Mestrado – DEI – PUC – Rio de Janeiro, RJ, 2005.

GILMORE, P.C.; GOMORY, R.E. A Linear Programming Approach to the Cutting Stock Problem. Operations Research, n. 9, 1961.

GOLDMAN, A.J. Optimal location for centers in a network. Transportation Science 3, 1969.

HOPPER, E; TURTON, B. C. H. An Empirical Investigation of Meta-heuristic and Heuristic Algorithms for a 2D Packing Problem. European Journal of Operational Research, n. 128, 2001.

JOHNSON, D. S. Approximation algorithms for combinatorial problems, J. Computer System, 1974

KALLRATH, Julia; REBENNACK, Steffen; KALLRATH, Josef; KUSCHE, Rüdiger. Solving Real-world Cutting Stock-problems in the Paper Industry: Mathematical Approaches, Experience and Challenges. European Journal of Operational Research, n. 238, 2014.

KLINCEWICZ, J. G. Avoiding Local Optima in The p-Hub Location Problem Using Tabu Search and GRASP. Anals of Operation Research, 1992.

KORTE, Bernhard; VYGEN, Jens. Combinatorial Optimization: Theory and Algorithms. Springer, 4 Ed. 2008.

LIMA JÚNIOR, F. C. Algoritmo Q-learning como Estratégia de Exploração e/ou Explotação para as Metaheurísticas GRASP e Algoritmo Genético. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte – UFRN, Natal, RN, 2009.

MORABITO, Reinaldo; ARENALES, Marcos N. Performance of Two Heuristics for Solving Large Scale Two-dimensional Guillotine Cutting Problems. INFOR, n. 2, 1995.

OLIVEIRA, F. M. O Problema de Localização de Seções Eleitorais e Alocação de Eleitores: Uma Abordagem Exata e Metaheurística. Dissertação de Mestrado – Universiade Estadual do Rio Grande do Norte - UERN, Mossoró, RN, 2013.

OLIVEIRA, José Fernando; FERREIRA, José Soeiro. An Improved Version of Wang's Algorithm for Two-dimensional Cutting Problems. European Journal of Operation Research, n. 44, 1990.

OKTAL, H.; ASUMAN, O. Hub location in air cargo transportation: A case study. Journal of Air Transport Management, 2014.

O'KELLY, M.E. A quadratic integer program for the location of interacting hub facilities. European Journal of Operational Research 32, 1987.

O'KELLY, M.E. Hub facility location with fixed costs. Papers in Regional Science 71, 1992.

O'KELLY, M.E., BRYAN, D., SKORIN-KAPOV, D., SKORIN-KAPOV, J., Hub network design with single and multiple allocation: a computational study, Location Science, v.4, 1996.

PRAIS, M.; RIBEIRO, C. C. Reactive GRASP: An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment. Journal On Computing, 123, 2000.

RIBEIRO, Paulo César Fernandes. Um enfoque na Localização de Facilidades Baseado em Teste de Redução ADD/DROP. Faculdade Lourenço Filho – Fortaleza, CE, 2008.

SIQUEIRA, M. C. Critérios para Preparação de Aeroportos para Operar como Hub. Brasília. 2008.

SILVA, Alexsandro Trindade Sales; LIMA JÚNIOR, Francisco Chagas; OLIVEIRA, Francisco Márcio, FIGUEIREDO, Jéssica Neiva. Algoritmo Guloso Aleatório Aplicado ao Problema de Corte Bidimensional Guilhotinado e Restrito. II WTCC, 2011.

SILVA, M. R.; CUNHA, C. B. New simple and efficient heuristics for the uncapacitated single allocation hub location problem. Computers & Operations Research, 2009.

SOUTELINO, André Luís Dias. Desmistificando o *Hub and Spoke*. Faculdade Lourenço Filho – Fortaleza, CE, 2008.

SUTTON, R. S.; BARTO, A. G. Reinforcement Learning: An Introduction. MA: MIT Press, 1998.

TEMPONI, Elias Carlos Correa. Uma Proposta de Resolução do Problema de Corte Bidimensional via Abordagem Metaheurística. Dissertação (Mestrado Modelagem Matemática e Computacional), CEFET, Belo Horizonte, MG, 2007.

TEODORO, Alan Augusto. O Problema de Corte Bidimensional: uma abordagem utilizando o método de geração de colunas. Dissertação (Mestrado Profissional em Computação), UNICAMP, São Paulo, SP, 2003.

VASKO, Francis J. A Computational Improvement to Wang's Two-dimensional Cutting Stock Algorithm. Cuputers Ind. Engineering, Vol. 16, 1989.

VELASCO, André Soares. GRASP para o Problema de Corte Bidimensional Guilhotinado e Restrito. Dissertação (Mestrado em Ciências de Engenharia), Centro de Ciência e Tecnologia da Universidade Estadual do Norte Fluminense. Campo dos Goyatacazes, RJ, 2005.

VIANNA, Andréa Carla Gonçalves, POLDI, Kelly Cristina. O Problema de Corte de estoque Bidimensional Aplicado a Uma Indústria de Esquadrias Metálicas. XXXVII Simpósio Brasileiro de Pesquisa Operacional, Gramado, RS, 2005.

WANG, P. Y. Two Algorithms for Constrained Two-dimensional Cutting Stock Problems. Operations Research, n. 31, 1983.

WÄSHER, G; HAUBNER, H; SCHUMMAN, H. An Improved typology cutting and Paccking Problems. European Journal of Operational Research, 2006.

YANASSE, H.H.; MORABITO. A note on linear models for two-group and three-group two-dimensional guillotine cutting problems. International Journal of Production Research, 2006.

APÊNDICE

A Tabela A1 traz a lista dos aeroportos utilizados para criação da instância BR2010, conforme ANAC (2011).

Tabela A1 – Aeroportos atendidos conforme ANAC (2011).

Nr.	ICAO	Aeroporto	Município	UF	Latitude	Longitude
1	SBCZ	Cruzeiro Do Sul	Cruzeiro Do Sul	AC	-7,5994	-72,7694
2	SBRB	Plácido De Castro	Sena Madureira	AC	-9,8688	-67,8981
3	SBMO	Zumbi Dos Palmares	Rio Largo	AL	-9,5172	-35,7836
4	SWBC	Barcelos	Barcelos	AM	-0,9815	-62,922
5	SWNK	Novo Campo	Boca Do Acre	AM	-8,8336	-67,3122
6	SWBR	Borba	Borba	AM	-4,4086	-59,5978
7	SWCA	Carauari	Carauari	AM	-4,8786	-66,8956
8	SWKO	Coari	Coari	AM	-4,1339	-63,1311
9	SWEI	Eirunepé	Eirunepé	AM	-6,6375	-69,8831
10	SWOB	Fonte Boa	Fonte Boa	AM	-2,5336	-66,0672
11	SWHT	Francisco Correa Da Cruz	Humaitá	AM	-7,5336	-63,0506
12	SWLB	Lábrea	Lábrea	AM	-7,2503	-64,7839
13	SBEG	Eduardo Gomes	Manaus	AM	-3,0411	-60,0506
14	SBMY	Manicoré	Manicoré	AM	-5,8169	-61,2839
15	SWMW	Maués	Maués	AM	-3,3569	-57,7122
16	SWPI	Regional Júlio Belem	Parintins	AM	-2,6694	-56,7711
17	SWTP	Tapuruquara	Santa Isabel Do Rio Negro	AM	-0,4169	-65,0339
18	SBUA	São Gabriel Da Cachoeira	São Gabriel Da Cachoeira	AM	-0,1481	-66,9858
19	SDCG	Senadora Eunice Michiles	São Paulo De Olivença	AM	-3,4694	68,9583
20	SBTT	Tabatinga	Tabatinga	AM	-4,2505	69,9377
21	SBTF	Tefé	Tefé	AM	-3,3803	64,7253
22	SBMQ	Macapá	Macapá	AP	0,0508	51,0703
23	SBAM	Amapá	Amapá	AP	2,0728	50,8625
24	SNBR	Barreiras	Barreiras	BA	-12,0792	45,0094
25	SBLP	Bom Jesus Da Lapa	Bom Jesus Da Lapa	BA	-13,2614	43,4075
26	SNGI	Guanambi	Guanambi	BA	-14,2069	42,7511
27	SBIL	Jorge Amado	Ilhéus	BA	-14,815	39,0333
28	SBLE	Horácio De Mattos	Lençois	BA	-12,4833	41,2731
29	SBUF	Paulo Afonso	Paulo Alfonso	BA	-9,4022	38,2542
30	SBPS	Porto Seguro	Porto Seguro	BA	-16,4381	39,0778
31	SBSV	Luís Eduardo Magalhães	Salvador	BA	-12,9086	38,3225
32	SBTC	Hotel Transamérica	Uma	BA	-15,3533	38,9972
33	SBQV	Vitória Da Conquista	Vitória Da Conquista	BA	-14,8636	40,8631
34	SBFZ	Pinto Martins	Fortaleza	CE	-3,7758	38,5322
35	SBJU	Orlando B. De Menezes	Juazeiro Do Norte	CE	-7,2183	39,2717
36	SBBR	Pdte. Juscelino Kubitschek	Brasília	DF	-15,8692	47,9208
37	SBVT	Eurico De Aguiar Salles	Vitória	ES	-20,2581	40,2864
38	SBCN	Nelson R. Guimarães	Caldas Novas	GO	-17,7247	-48,61

39	SBGO	Santa Genoveva	Goiânia	GO	-16,6297	49,2267
40	SBMC	Minaçu	Minaçu	GO	-13,5506	48,2006
41	SWLC	General Leite De Castro	Rio Verde	GO	-17,8347	-50,9561
42	SBIZ	Prefeito Renato Moreira	Imperatriz	MA	-5,5306	-47,4583
43	SBSL	Marechal Cunha Machado	São Luis	MA	-2,5869	-44,2361
44	SBAX	Romeu Zema	Araxá	MG	-19,5606	-46,9656
45	SBBQ	Mj. Doorgal Gomes	Barbacena	MG	-21,2672	-43,7606
46	SBCF	Tancredo Neves	Confins	MG	-19,6244	-43,9719
47	SBBH	Pampulha	Belo Horizonte	MG	-19,8519	-43,9506
48	SNDT	Diamantina	Diamantina	MG	-18,2333	-43,6511
49	SBGV	Coronel Altino Machado	Governador Varadales	MG	-18,8969	-41,9861
50	SBIP	Usiminas	Santana Do Paraiso	MG	-19,4706	-42,488
51	SNAP	Janaúba	Janaúba	MG	-15,7331	-43,3239
52	SBJF	Francisco Alvaro De Assis	Juiz De Fora	MG	-21,7931	-43,3856
53	SBMK	Mário Ribeiro	Montes Claros	MG	-16,7061	-43,8219
54	SNPD	Patos De Minas	Patos De Minas	MG	-18,6722	-46,4914
55	SNJR	São João Del Rei	São João Del Rei	MG	-21,0864	-44,2264
56	SNPY	São Sebastião Do Paraíso	São Sebastião Do Paraíso	MG	-20,9492	-46,9842
57	SBUR	Mário De Almeida Franco	Uberaba	MG	-19,7647	-47,9661
58	SBUL	Aviador Cesar Bombonato	Uberlândia	MG	-18,8836	-48,2253
59	SBVG	Mj. Brig. Trompowsky	Varginha	MG	-21,5925	-45,4744
60	SBDB	Bonito	Bonito	MS	-21,2294	-56,4561
61	SBCG	Campo Grande	Campo Grande	MS	-20,4694	-51,6703
62	SBCR	Corumbá	Corumbá	MS	-19,0119	-57,6714
63	SBDO	Francisco De M. Pereira	Dourados	MS	-22,1992	-54,9081
64	SBAT	Oswaldo Marques Dias	Alta Floresta	MT	-9,8664	-56,105
65	SWRP	Aripuanã	Aripuanã	MT	-10,2531	-59,3892
66	SJHG	Confresa	Confresa	MT	-10,6336	-51,5672
67	SBCY	Marechal Rondon	Várzea Grande	MT	-15,65	-56,1175
68	SIZX	Inácio Luiz Do Nascimento	Juara	MT	-11,2867	-57,5389
69	SWJN	Juína	Juína	MT	-11,4194	-58,7017
70	SWRD	Rondonópolis	Rondonópolis	MT	-16,5853	-54,7242
71	SWFX	São Felix Do Araguaia	São Félix Do Araguaia	MT	-11,6319	-50,6883
72	SWSI	João Batista Figueiredo	Sinop	MT	-11,885	-55,5851
73	SBMD	Monte Dourado	Almeirim	PA	-0,8897	-52,6022
74	SBHT	Altamira	Altamira	PA	-3,2508	-52,2522
75	SBBE	Val De Cans -Júlio Cezar	Belém	PA	-1,3847	-48,4789
76	SBAA	Conceição Do Araguaia	Conceição Do Araguaia	PA	-8,3486	-49,3031
77	SBIH	Itaituba	Itaituba	PA	-4,2422	-56,0008
78	SBMA	João Correa Da Rocha	Marabá	PA	-5,3678	-49,1383
79	SBTB	Porto Trombetas	Oriximiná	PA	-1,4844	-56,3992
80	SDOW	Ourilândia Do Norte	Ourilândia Do Norte	PA	-6,7758	-51,06
81	SBCJ	Carajas	Parauapebas	PA	-6,1153	-50,0014
82	SNDC	Redenção	Redenção	PA	-8,0306	-49,9806
83	SNKE	Santana Do Araguaia	Santana Do Araguaia	PA	-9,3353	-50,3503
84	SBSN	Maestro Wilson Fonseca	Santarém	PA	-2,4247	-54,7858
85	SNFX	São Félix Do Xingu	São Félix Do Xingu	PA	-6,6333	-51,9834

86	SBTU	Tucuruí	Tucuruí	PA	-3,7769	-49,7197
87	SBKG	Presidente João Suassuna	Campina Grande	PB	-7,2692	-35,895
88	SBJP	Presidente Castro Pinto	Bayeux	PB	-7,1458	-34,9486
89	SNRU	Oscar Laranjeiras	Carauari	PE	-8,2844	-36,0108
90	SBFN	Fernando De Noronha	Fernando De Noronha	PE	-3,8547	-32,4283
91	SBPL	Senador Nilo Coelho	Petrolina	PE	-9,3675	-40,5636
92	SBRF	Guararapes	Recife	PE	-8,1264	-34,9228
93	SBTE	Senador Petrônio Portella	Teresina	PI	-5,0606	-42,8244
94	SBCA	Adalberto M. Da Silva	Cascavel	PR	-25,0022	-53,5019
95	SBCT	São José Dos Pinhais	Curitiba	PR	-25,5358	-49,1714
96	SBFI	Cataratas (De Iguaçu)	Foz De Iguaçu	PR	-25,6003	-54,485
97	SBGU	Tancredo T. De Faria	Guarapuava	PR	-25,3881	-51,5217
98	SBLO	Governador José Richa	Londrina	PR	-23,3303	-51,1367
99	SBMG	Maringá	Maringá	PR	-23,4794	-52,0122
100	SDAG	Angra Dos Reis	Angra Dos Reis	RJ	-22,9753	-44,3072
101	SBBZ	Umberto Modiano	Armação De Búzios	RJ	-22,7661	-41,9655
102	SBCB	Cabo Frio	Cabo Frio	RJ	-22,9208	-42,0714
103	SBCP	Bartolomeu Lisandro	Campos Dos Goytacazes	RJ	-21,6984	-41,3019
104	SBME	Macaé	Macaé	RJ	-22,3458	-41,7639
105	SBGL	Galeão	Rio De Janeiro	RJ	-22,81	-43,2506
106	SBJR	Jacarepaguá	Rio De Janeiro	RJ	-22,9869	-43,371
107	SBRJ	Santos Dumont	Rio De Janeiro	RJ	-22,9103	-43,1628
108	SBMS	Dix-Sept Rosado	Mossoró	RN	-5,1958	-37,3617
109	SBNT	Augusto Severo	Parnamirim	RN	-5,9083	-35,2492
110	SBJI	Ji-Paraná	Ji-Paraná	RO	-10,8706	-61,8467
111	SBPV	Jorge Teixeira De Oliveira	Porto Velho	RO	-8,7136	-63,9028
112	SBVH	Brigadeiro Camarão	Vilhena	RO	-12,6942	-60,0972
113	SBBV	Atlas Brasil Cantanhede	Boa Vista	RR	2,8414	-60,6922
114	SBCX	Regional Hugo Cantergiani	Caxias Do Sul	RS	-29,1956	-51,1898
115	SSER	Erechim	Erechim	RS	-27,66	-52,2761
116	SSIJ	Aeroporto De Ijuí	Ijuí	RS	-28,37	-53,8455
117	SBPF	Lauro Kurtz	Passo Fundo	RS	-28,2453	-52,3286
118	SBPK	Pelotas	Pelotas	RS	-31,7161	-52,3311
119	SBPA	Salgado Filho	Porto Alegre	RS	-29,9939	-51,1711
120	SJRG	Rio Grande	Rio Grande	RS	-32,0817	-52,1633
121	SBSM	Santa Maria	Santa Maria	RS	-29,7108	-53,6922
122	SSZR	Santa Rosa	Santa Rosa	RS	-27,9089	-54,5222
123	SBNM	Santo Angelo	Santo Angelo	RS	-28,2822	-54,1689
124	SBUG	Internacional Rubem Berta	Uruguaiana	RS	-29,7819	-57,0383
125	SBCD	Carlos A. Da Costa Neves	Caçador	SC	-26,7897	-50,9397
126	SBCH	Serafin Enoss Bertaso	Chapeco	SC	-27,1339	-52,6619
127	SBCM	Diomício Freitas	Forquilhinha	SC	-28,7244	-49,4214
128	SBFL	Hercılio Luz	Florianópolis	SC	-27,6703	-48,5525
129	SSJA	Santa Terezinha	Joaçaba	SC	-27,1728	-51,5517
130	SBJV	Lauro Carneiro De Loyola	Joinville	SC	-26,2231	-48,7975
131	SBNF	Ministro Victor Konder	Navegantes	SC	-26,8786	-48,6508
132	SBAR	Santa Maria	Aracaju	SE	-10,9853	-37,0733

133	SBAU	Estadual Dario Guarita	Araçatuba	SP	-21,1442	-50,4264
134	SBBT	Chafei Amsei	Barretos	SP	-20,5856	-48,5958
135	SBAE	Bauru - Arealva	Arealva	SP	-22,1578	-49,0683
136	SBKP	Viracopos	Campinas	SP	-23,0069	-47,1344
137	SBML	Frank Milove Milenkovich	Marilia	SP	-22,1956	-49,9269
138	SBDN	Presidente Prudente	Presidente Prudente	SP	-22,1783	-51,4189
139	SBRP	Leite Lopes	Ribeirão Preto	SP	-21,1364	-47,7767
140	SDSC	Mário Pereira Lopes	São Carlos	SP	-21,8764	-47,9033
141	SBSR	Eriberto Manoel Reino	São José Do Rio Preto	SP	-20,8161	-49,4047
142	SBSJ	Urbano Ernesto Stumpf	São José Dos Campos	SP	-23,2268	-45,8619
143	SBSP	Congonhas	São Paulo	SP	-23,6261	-46,6539
144	SBGR	André Franco Montoro	Guarulhos	SP	-23,4356	-46,4731
145	SWGN	Araguaína	Araguaína	TO	-7,2283	-48,2408
146	SWGI	Gurupi	Gurupi	TO	-11,7392	-49,1322
147	SBPJ	Brig. Lysias Rodrigues	Palmas	TO	-10,29	-48,3578

A Tabela A2 traz a lista dos aeroportos que atendem as restrições de Siqueira (2008) para atuar como *hub*.

Tabela A2 – Aeroportos que podem atuar como *hub* (SIQUEIRA, 2008).

Nr.	ICAO	Aeroporto	Município	UF	PAX (Ano)
3	SBMO	Zumbi Dos Palmares	Rio Largo	AL	705.350
13	SBEG	Eduardo Gomes	Manaus	AM	1.298.797
30	SBPS	Porto Seguro	Porto Seguro	BA	536.097
31	SBSV	Deputado Luís Eduardo Magalhães	Salvador	BA	3.899.710
34	SBFZ	Pinto Martins	Fortaleza	CE	2.424.499
36	SBBR	Presidente Juscelino Kubitschek	Brasília	DF	7.244.495
33	SBQV	Vitória Da Conquista	Vitória Da Conquista	BA	1.245.178
39	SBGO	Santa Genoveva	Goiânia	GO	1.083.786
43	SBSL	Marechal Cunha Machado	São Luis	MA	670.640
46	SBCF	Tancredo Neves	Confins	MG	3.481.352
58	SBUL	Tenente-Coronel Aviador cesar Bombonato	Uberlândia	MG	374.678
61	SBCG	Campo Grande	Campo Grande	MS	657.984
67	SBCY	Marechal Rondon	Várzea Grande	MT	1.081.287
75	SBBE	Val De Cans -Júlio Cezar	Belém	PA	1.340.418
88	SBJP	Presidente Castro Pinto	Bayeux	PB	462.129
92	SBRF	Guararapes - Gilberto Freyre	Recife	PE	3.276.547
93	SBTE	Senador Petrônio Portella	Teresina	PI	381.907
95	SBCT	São José Dos Pinhais, Alfonso Pena	Curitiba	PR	2.928.096
96	SBFI	Cataratas (De Iguaçu)	Foz De Iguaçu	PR	564.090
98	SBLO	Governador José Richa	Londrina	PR	373.928
105	SBGL	Antônio Carlos Jobim - Galeão	Rio De Janeiro	RJ	4.640.879
107	SBRJ	Santos Dumont	Rio De Janeiro	RJ	3.925.762
109	SBNT	Augusto Severo	Parnamirim	RN	1.131.616
111	SBPV	Governador Jorge Teixeira De Oliveira	Porto Velho	RO	368.090
119	SBPA	Salgado Filho	Porto Alegre	RS	3.048.924
128	SBFL	Hercılio Luz	Florianópolis	SC	1.236.691
131	SBNF	Ministro Victor Konder	Navegantes	SC	409.659
132	SBAR	Santa Maria	Aracaju	SE	472.471
136	SBKP	Viracopos	Campinas	SP	1.897.672
143	SBSP	Congonhas	São Paulo	SP	7.853.742
144	SBGR	Governador André Franco Montoro	Guarulhos	SP	8.244.810

As Figuras de 41 a 44 trazem a representação gráfica de algumas instâncias do PCB.

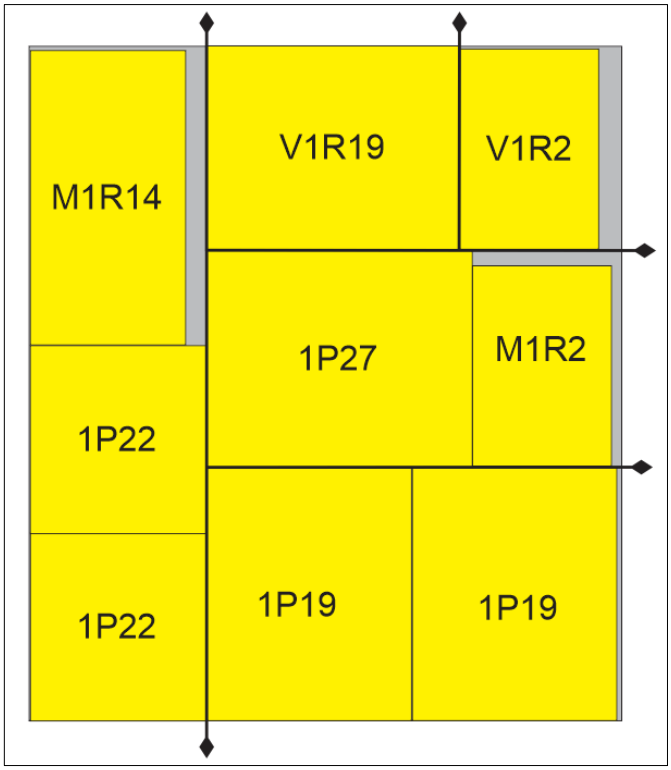


Figura 41 – Padrão de corte para instância CW1.

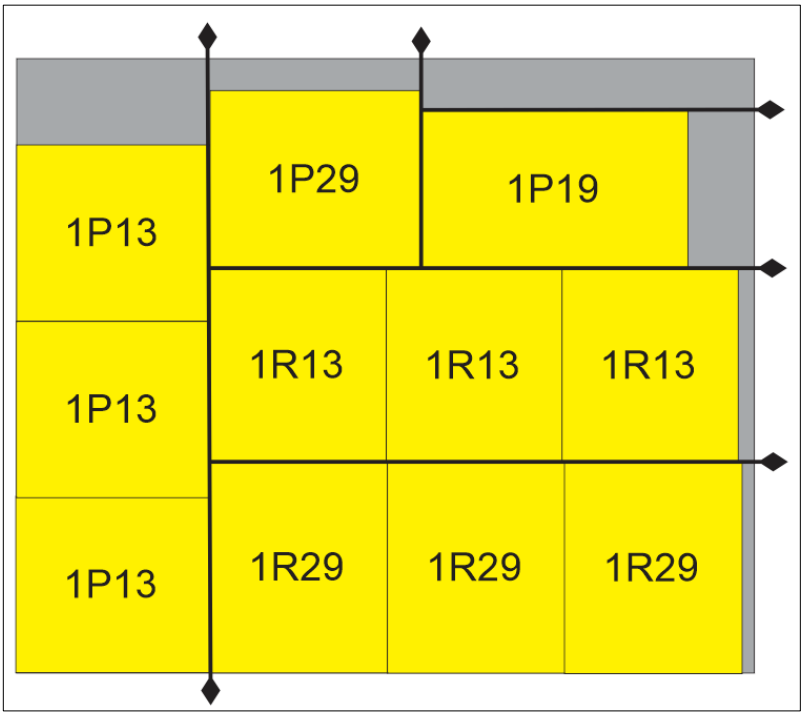


Figura 42 – Padrão de corte para instância CW4.

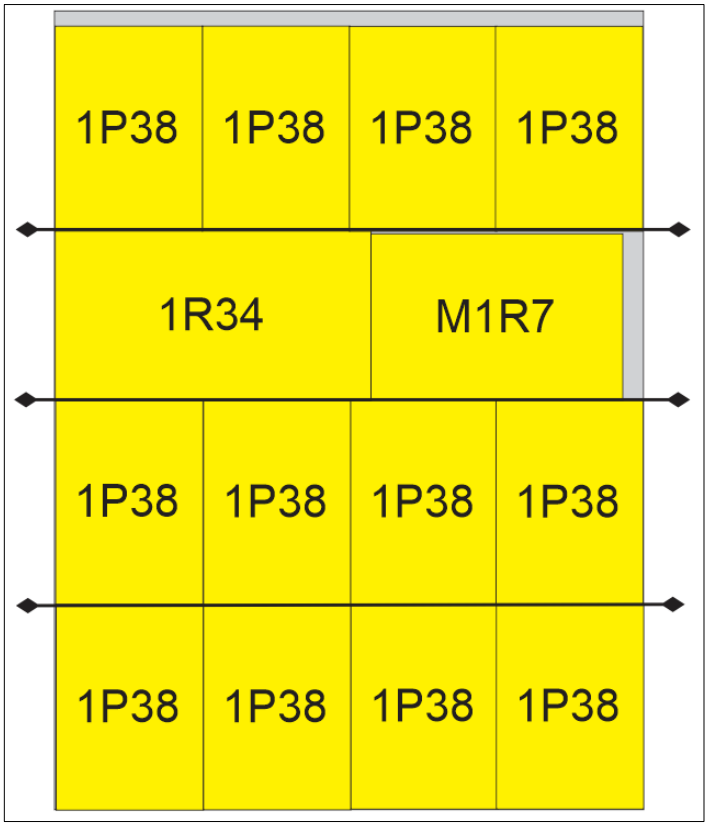


Figura 43 – Padrão de corte para instância CW7.

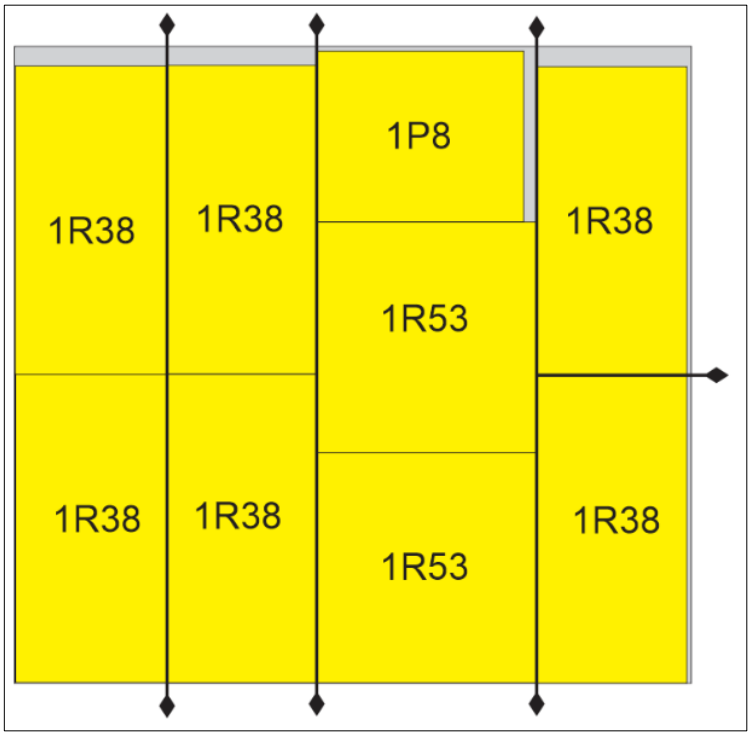


Figura 44 – Padrão de corte para instância CW8.