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

Jeferson Queiroga Pereira

UMA ABORDAGEM VIA METAHEURÍSTICA HÍBRIDA PARA O PROBLEMA DE ATRIBUIÇÃO DE LOCALIDADES A ANÉIS SONET/SDH

Mossoró-RN

2018

Jeferson Queiroga Pereira

UMA ABORDAGEM VIA METAHEURÍSTICA HÍBRIDA PARA O PROBLEMA DE ATRIBUIÇÃO DE LOCALIDADES A ANÉIS SONET/SDH

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 Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Profº Dr. Dario José Aloise

Mossoró-RN

2018

Catalogação da Publicação na Fonte.
Universidade do Estado do Rio Grande do Norte.

P436a Pereira, Jeferson Queiroga
UMA ABORDAGEM VIA METAHEURÍSTICA HÍBRIDA PARA O PROBLEMA DE ATRIBUIÇÃO DE LOCALIDADES A ANÉIS SONET-SDH. / Jeferson Queiroga Pereira. - Mossoró, 2018.
84p.

Orientador(a): Prof. Dr. Dario José Aloise.
Dissertação (Mestrado em Programa de Pós-Graduação em Ciência da Computação). Universidade do Estado do Rio Grande do Norte.

1. Projeto de Redes. 2. BRKGA. 3. PALAS. 4. Vocabulary Building. 5. Q-learning. I. Aloise, Dario José. II. Universidade do Estado do Rio Grande do Norte. III. Título.

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC´s) foi desenvolvido pela Diretoria de Informatização (DINF), sob orientação dos bibliotecários do SIB-UERN, para ser adaptado às necessidades da comunidade acadêmica UERN.

JEFERSON QUEIROGA PEREIRA

Uma Abordagem Via Metaheurística Híbrida para o Problema de Atribuição de Localidades a Anéis SONET/SDH.

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

APROVADA EM: 19 / 04 / 2018

Prof. Dr. Dario José Aloise
Orientador e Presidente

Prof. Dr. Carlos Heitor Pereira Liberalino
Examinador - Universidade do Estado do Rio Grande do Norte

Prof. Dr. Plácido Rogerio Pinheiro
Universidade de Fortaleza - UNIFOR

Prof. Dr. Fabio Francisco da Costa Fontes
Universidade Federal Rural do Semi-Árido

Agradecimentos

Primeiramente a Deus, pela dádiva da vida, e por ter me direcionado por bons caminhos por esta estrada da vida, sendo esses nem sempre fáceis, porém dando-me forças e fé para superar todos os obstáculos encontrados.

À minha família que tanto me apoiou em todas as etapas da minha vida, minha mãe Edilza Custódio, meu pai Júlio Batista e minha irmã Anatália Cristina (in memoriam).

Em especial, agradeço à minha esposa Lorena Holanda, que sempre esteve ao meu lado nesta conquista e superou comigo todas as dificuldades vivências nesse período. Obrigado amor.

Ao meu orientador Dario Aloise pela oportunidade que me concedeu na realização deste trabalho, pela orientação e valiosos ensinamentos dados durante a pesquisa.

Enfim, a todos que de alguma forma me ajudaram, direta ou indiretamente.

Resumo

Os sistemas de telecomunicações estão na fase de grandes transformações e expansões, que tornam os problemas de planejamento de redes de telecomunicações cada vez maiores e mais complexos. Com isso, muitos desses problemas podem ser formulados como modelos de otimização combinatória, e o uso de algoritmos heurísticos podem ajudar a solucionar essas questões da fase de planejamento. Este trabalho propõe uma implementação da metaheurística BRKGA (*Biased Random-Key Genetic Algorithm*) – além de duas implementações híbridas – BRKGA com *Vocabulary Building* (BRKGA+VB) e BRKGA com *Q-learning* (BRKGA+QL) – para o Problema de Atribuição de Localidades a Anéis SONET/SDH (ou abreviadamente, PALAS). Neste problema, cada localidade cliente deve ser atribuída a exatamente um anel SONET, também denominado de anel local e um anel especial, chamado de Anel Federal, que interligam os anéis locais entre si. É imposta sobre cada anel uma restrição de capacidade. O objetivo do problema é encontrar uma atribuição de localidades clientes que minimize o número total de anéis utilizados, pois quanto menos anéis, menor o custo total da rede. Esse problema é NP-difícil e, portanto, não se pode garantir a obtenção dos melhores resultados para todas instâncias utilizando os métodos exatos, em um tempo computacional viável, dessa forma, é proposto para solução desse problema, a utilização dos métodos heurísticos. Os algoritmos foram implementados na linguagem de programação JAVA e utilizaram as instâncias das classes C1, C2, C3 e C4 para realizações dos experimentos computacionais. A análise dos experimentos mostrou a competitividade dos algoritmos propostos frente aos melhores resultados encontrados na literatura.

Palavras-chaves: Projeto de Redes, BRKGA, PALAS, *Vocabulary Building*, *Q-learning*

Abstract

The Telecommunication systems are in the phase of major transformations and expansions, which make telecommunication network planning problems increasingly larger and complex. In this context, many of these problems can be formulated as combinatorial optimization models, and the use of heuristic algorithms can help solve these issues in the planning phase. This paper proposes an implementation of the BRKGA (Biased Random-Key Genetic Algorithm) metaheuristic - in addition to two hybrid implementations - BRKGA with Vocabulary Building (BRKGA + VB) and BRKGA with Q-learning (BRKGA + QL) - to be applied to problem SONET Ring Assignment Problem – SRAP. In this problem, each customer location must be assigned to exactly one SONET ring, also called the local ring and a special ring, called the Federal Ring, which interconnect the local rings with each other. A capacity constraint is imposed on each ring. The solution looking for the problem is to find an allocation of customer locations that minimize the total number of rings used, since the fewer rings the lower the cost of the telecommunication network planning. This problem is the class NP-hard, so it can not be guaranteed to obtain the best results for all instances using the exact algorithms, with a viable computational time. It is proposed to solve this problem with the heuristic methods mentioned above. The algorithms were implemented in the JAVA programming language and used the instances of classes C1, C2, C3 and C4 to perform computational experiments. The analysis of the experiments showed the competitiveness of the proposed algorithms against the best results found in the literature.

Keywords: Network Design, BRKGA, PALAS, Vocabulary Building, Q-learning

Sumário

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ADM	*Add-Drop Multiplexer*
ANSI	*American National Standards Institute*
AR	Aprendizagem por Reforço
APS	*Automatic Protection Switching*
BRKGA	*Biased Random-Key Genetic Algorithms*
DCS	*Digital Cross Connect System*
ECSA	*Exchange Carriers Standards Association*
DMN	*Diversication by Multiple Neighborhhods*
VB	*Vocabulary Build*
PALAS	Problema da Atribuição de Localidades a Anéis em Redes SONET/SDH;
PUC	*Parametrized Uniform Crossover*
PDH	*Plesiochronous Digital Hierarchy*
RKGA	*Random-Key Genetic Algorithms*
RTD	*Random by Traffic Density*
RTNR	*Random with Target Number of Rings*
SONET	*Synchronous Optical NETwork*
SDH	*Synchronous Digital Hierarchy*
TSSO	*Tabu Search with Strategic Oscillation*

1 Introdução

Os sistemas de telecomunicações estão na fase de grandes transformações e expansões, com o crescimento da demanda do tráfego impulsionado pela comercialização e utilização dos recursos de acesso à Internet por banda larga (móvel ou fixa), telefonia e serviços de dados para transmissão de mídias em tempo real, tais como: serviços de *streaming*, transferência de dados, imagens, voz, operações bancárias, compras, entre outros. Essa expansão é requisitada por uma sociedade contemporânea, que está cada vez mais dependente de redes eficientes e confiáveis, pois, mais e mais produtos e serviços são incorporados no mercado frequentemente. Esse crescimento nos serviços de telecomunicações, requer das operadoras maior investimento na ampliação da infraestrutura.

Apesar das tecnologias da telecomunicações estarem sempre evoluindo, atendendo sempre novas demandas, existe a problemática referente à infraestrutura das redes, em virtude do crescimento demasiado da quantidade de equipamento necessário, bem como a grande quantidade de clientes, e isso torna o planejamento de rede de telecomunicações um problema complexo para os profissionais da área de redes de computadores.

Dessa forma, os problemas de planejamento de redes têm despertado bastante interesse entre pesquisadores de várias áreas, principalmente na área de Pesquisa Operacional, pois muitos desses impasses da fase de planejamento, podem ser formulados como problemas de otimização combinatória. Nas pesquisas sobre planejamento de redes encontram-se várias aplicações, como por exemplo, rede de telecomunicações, transporte urbano, água, gás etc. A modelagem mais utilizada para representar as topologias destas redes, são os grafos, onde cada vértice corresponde aos pontos que são interligados entre si, através da aresta (SAMANTA; PAL, 2013).

Este trabalho discute um dos problemas de otimização identificado durante o planejamento de uma rede de telecomunicações, o Problema de Atribuição de Localidades a Anéis SONET (*Synchronous Optical Network*), ou abreviadamente, PALAS. Esse problema surgiu com o uso da tecnologia SONET, que emprega a topologia em anel para atender a capacidade de sobrevivência de uma rede de telecomunicações, ou seja, para a capacidade dessa rede permanecer ativa, caso ocorra falha numa localidade ou numa ligação. A utilização dessa topologia, tornou o planejamento físico um problema de otimização combinatória, pois consiste na determinação das conexões entre um conjunto de localidades (clientes), de modo a satisfazer, um conjunto de restrições e minimizando os custos possíveis. A seguir, uma descrição mais formal do PALAS:

Dado um conjunto de localidades L que pertencem a uma rede de telecomunicações, um conjuntos de demandas d_{uv} entre duas localidades u e v, com u, v $\in$ L, e um inteiro B que

representa a capacidade de tráfego num anel, o PALAS consiste em atribuir as localidades a subconjuntos, que representam os anéis, tal que, a capacidade desses subconjuntos sejam satisfeitas de acordo com o parâmetro B e todas as demandas entre as localidades sejam satisfeitas com diminuição dos custos.

O PALAS é NP-difícil (GOLDSCHMIDT; LAUGIER; OLINICK, 2003), e, portanto, não existe algoritmos polinomiais para resolvê-lo, a menos que P=NP. Dessa forma a aplicação de métodos heurísticos é usada para encontrar soluções sub-ótimas (eventualmente, ótimas) em um tempo computacional viável (polinomial).

O principal objetivo deste trabalho, se baseia em propor novas metaheurísticas híbridas que consistem na junção de duas metaheurísticas ou uma metaheurística com conceitos e técnicas de outras áreas de pesquisas, para a resolução do PALAS de forma satisfatória. O desenvolvimento de metaheurísticas híbridas tem sido uma tendência na pesquisa em otimização combinatória, ganhando um aumento considerável no interesse entre os pesquisadores dessa área (TALBI, 2002). Uma explicação para isso é porque os algoritmos híbridos têm adquirido melhores resultados para muitos problemas de otimização, como pode ser visto em Lima (Lima Júnior, 2009).

Este trabalho utiliza a metaheurística BRKGA (*Biased Random-Key Genetic Algorithm*) a qual é considerada relativamente recente, tendo seus primeiros resultados competitivos frente a literatura da área de Pesquisa Operacional no ano de 2002 (ERICSSON; RESENDE; PARDALOS, 2002). Afim de aprimorar a eficiência do BRKGA, serão apresentadas duas versões híbridas dessa metaheurística: (i) BRKGA junto com Q-learning que é definido como um algoritmo de Aprendizagem por Reforço (AR). Aprendizagem por reforço é considerado um paradigma de aprendizagem de máquina que se baseia na capacidade de um agente obter conhecimento interagindo com o ambiente no qual está inserido. A intenção é melhorar a aptidão da população inicial do BRKGA utilizando o *Q-learning*. (ii) BRKGA mais *Vocabulary Build* (VB). VB é considerado uma técnica de otimização que foi idealizada por Fred Glover no início da década de 90. A técnica VB tem por objetivo armazenar os melhores vocábulos no pool (memória adaptativa). Os melhores indivíduos de uma população podem possuir boas soluções parciais, que, se aplicadas a outras, podem melhorar seus respectivos *fitness*. Geralmente esses vocábulos do VB são gerados de forma aleatória ou selecionando elementos da elite. Então, procurando aprimorar esse processo de geração de vocábulo, esse trabalho propõe a utilização de aprendizado por reforço para geração dos vocábulos utilizado pelo VB. A ideia é se utilizar do algoritmo *Q-learning* para geração de vocábulos com boas qualidades, a fim de melhorar o desempenho do VB junto com os algoritmos proposto.

Ao longo deste trabalho, será detalhado os algoritmos e técnicas empregadas na resolução do problema. A ideia do trabalho final, é realizar uma comparação de resultados adquiridos nos experimentos computacionais, sobre as instâncias disponíveis na literatura,

para que possamos demonstrar a relevância das metaheurísticas híbridas.

O trabalho está organizado na seguinte estrutura: no Capítulo 2, será apresentado de forma sucinta, o Problema de Atribuição de Localidades a Anéis em Redes SONET/SDH, assim como a formulação matemática utilizada na resolução do mesmo, e, revisão da literatura. O Capítulo 3, introduziremos os conceitos de metaheurísticas e apresentação do Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA). Em seguida, no capítulo 4, apresentaremos detalhamento, as estratégias utilizadas para a resolução do problema PALAS, desde as metaheurísticas utilizadas, até a o processo de busca local. No Capítulo 5, serão mostrados os resultados computacionais obtidos, nos experimentos realizado durante o trabalho, onde será feita uma análise comparativa dos resultados encontrados para todas as classes, tanto em relação à qualidade média das soluções, quanto em relação ao desempenho. Além disso, é feito a comparação dos resultados obtidos para as instâncias das classes C1, C2, C3 e C4. Por fim, no capítulo 6, são apresentadas as conclusões do trabalho e sugestões para pesquisas futuras.

2 Fundamentação Teórica

Neste capítulo, são apresentados os fundamentos acerca dos temas abordados no presente trabalho. As próximas seções trazem a definição e os principais conceitos do padrão SONET, formulação matemática e revisão da literatura.

2.1 Apresentação do Problema de Atribuição de Localidades a Anéis SONET

Inúmeros problemas do nosso dia a dia, são capazes de ser modelados como problemas de otimização combinatória (PAPADIMITRIOU; STEIGLITZ, 1982), (COOK et al., 1998). Nestes problemas, não basta ter uma das soluções viáveis, mas, uma que, também otimize os objetivos de interesse. Alguns objetivos típicos dos problemas de otimização combinatória, consistem no geral, em aproveitar melhor os materiais no processo de produção, otimizar o tempo para a realização das ações, transportar mais materiais pelas melhores rotas, aumentar lucros, minimizar custo do planejamento das redes de telecomunicações, etc.

A maioria dos problemas de otimização combinatória, não dispõem de métodos exatos que obtenham soluções ótimas em tempo computacional viável. Muitos estudos vêm demonstrando, por meio de cálculos estatísticos, a inexistência de procedimentos exatos, capazes de resolver eficientemente uma determinada classe de problemas combinatórios, que são denominados de NP-Difícil. Com isso, tornam-se necessários à utilização de algoritmos heurísticos, que, segundo (GAREY; JOHNSON, 2002) são algoritmos polinomiais que buscam minimizar ao máximo, a possível perda de qualidade na solução alcançada pelo métodos exatos, e obtendo o máximo de eficiência computacional, sem a garantia de encontrar a solução ótima. Nas classes de problemas de resoluções difíceis, encontram-se alguns, que abrangem o desenvolvimento de projetos de redes de telecomunicações.

A atribuição de localidades a anéis SONET, é um problema de otimização combinatória, NP-Difícil, presente no projeto de redes de telecomunicações, que é tratado neste trabalho. O planejamento de uma rede SONET é uma tarefa que é considerada complexa, na qual, um conjunto de restrição tecnológica deve ser levada em consideração (OMIDYAR; ALDRIDGE, 1993) e (WASEM; WU; CARDWELL, 1994), este planejamento pode ser dividido em projeto físico (determinação dos que darão origem aos anéis) e projeto lógico (estabelecimento das conexões entre as localidades).

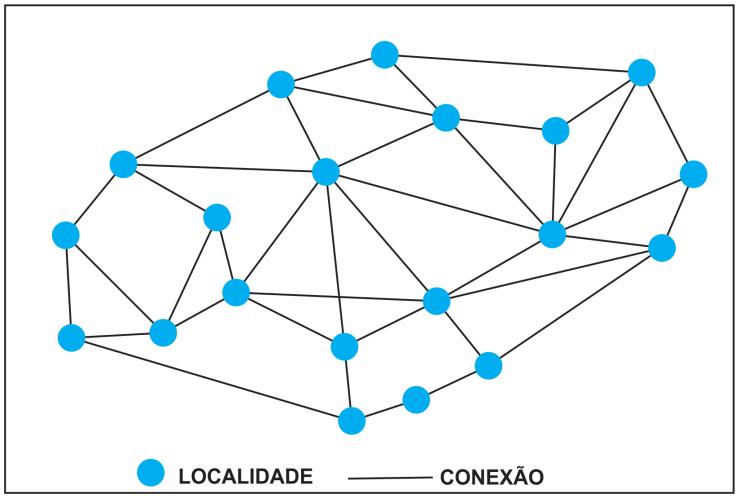
2.2 Planejamento de uma Rede de Telecomunicação

O desenvolvimento de um projeto de infraestrutura de redes, é um desafio em diferentes áreas de negócios e empresas de serviços, tais como: telecomunicações, distribuição de água, gás e energia, tendo sempre como meta: otimizar a instalação de equipamentos em uma região geográfica, com o objetivo de reduzir custos.

As redes de telecomunicações geralmente são compostas de dois níveis: rede *backbone* e redes de acesso local (SORIANO et al., 1999). A primeira busca, facilitar o tráfego de informações entre os usuários, enquanto a segunda, tem a função de concentrar essas informações entre os usuários.

O planejamento de uma rede *backbone* resume-se em determinar uma conexão entre as localidades de redes, respeitando um conjunto de restrições, a um custo mínimo. A figura 1 exemplifica o estado inicial de uma rede, com suas localidades e conexões entre elas.

Figura 1 – Estado inicial de uma rede backbone



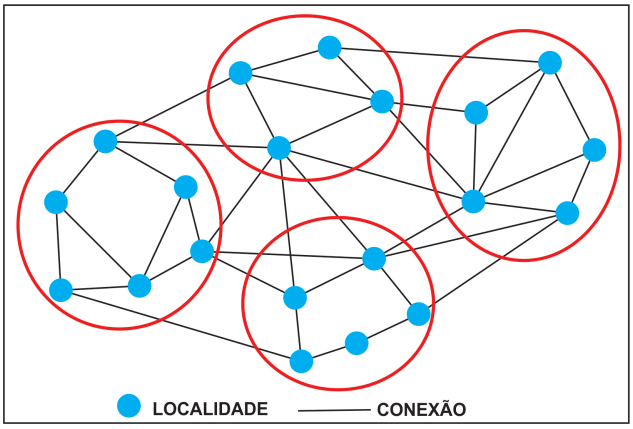
Fonte: Adaptado de (OLIVEIRA, 2010)

Dentre as restrições da rede *backbone*, destaca-se a restrição referente à capacidade de sobrevivência da rede, ou seja, capacidade de continuar funcionando em caso de alguma falha em uma localidade ou uma ligação. Esta capacidade na rede *backbone* pode ser adquirida, com uso de uma das versões da topologia em anel. Neste trabalho, utilizaremos a versão que possui uma estrutura, em que múltiplos anéis disjuntos, são interligados através de um anel denominado de Anel Federal. (GOLDSCHMIDT; LAUGIER; OLINICK, 2003).

O planejamento de uma rede *backbone* com topologia em anel, é uma tarefa difícil, que geralmente é dividida em projeto físico (determinação dos subconjuntos de localidades que darão origem aos anéis) e projeto lógico (estabelecimento das conexões entre as localidades). A Figura 2, exibe o particionamento das localidades da rede *backbone*, em

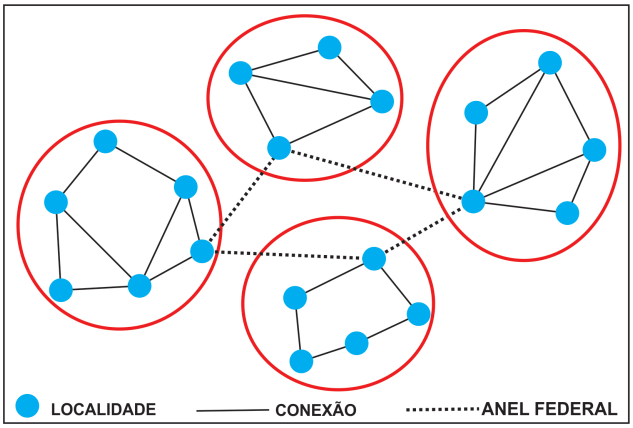
subconjuntos. Enquanto na figura 3, é apresentado um exemplo de conexão para estas localidades, denominado de projeto lógico.

Figura 2 – Projeto físico de uma rede



Fonte: Adaptado de (OLIVEIRA, 2010)

Figura 3 – Projeto lógico de uma rede



Fonte: Adaptado de (OLIVEIRA, 2010)

2.3 O Padrão SONET/SDH

Na década de 1980, a infraestrutura das operadoras, era baseada em PDH (*Plesiochronous Digital Hierarchy*), porém, este padrão possuía vários problemas, como: multiplexação dispendiosa, pouca padronização e baixas taxas de transmissões. Além disso, o crescimento dos serviços de comunicação e o surgimento da fibra ótica, culminaram para o desenvolvimento de uma nova hierarquia de transmissão (RAMASWAMI, 1998).

A partir desse contexto, surgiu a tecnologia SONET (*Synchronous Optical Network*) que pode ser definida como um padrão para transporte de comunicações óticas formuladas pela ECSA (*Exchange Carriers Standards Association* - Associação de Padrões de

Intercâmbio de Portadoras). Posteriormente ao seu desenvolvimento, um trabalho de colaboração entre o ANSI (*American National Standards Institute*) e o CCITT (*Comité consultatif International Téléphonique et Télégraphique*), produziu um padrão denominado SDH (*Synchronous Digital Hierarchy*, em português - Hierarquia Digital Síncrona), que seria o equivalente do SONET mundialmente. Ficando assim, o padrão conhecido como SONET/SDH. Estas tecnologias, são os padrões de transmissão e multiplexação para sinais de alta velocidade, dentro das infraestruturas das operadoras espalhadas pelo mundo.

O Padrão SONET/SDH define níveis de portadora ótica (*Optical Carrier -OC*) e seu equivalente elétrico (*Synchronous Transport Signal – STS*, no SONET, e *Synchronous Transport Module – STM*, no SDH) para a hierarquia de transmissão baseada em fibra ótica.

A maior flexibilidade de configuração e disponibilidade de banda larga do SONET, oferece vantagens significativas sobre o antigo sistema de telecomunicações. Algumas dessas vantagens são: redução dos requisitos dos equipamentos e aumento da confiabilidade da rede, possibilidade de conectar equipamentos de fabricantes diferentes, definição de uma arquitetura flexível capaz de acomodar futuras aplicações, com uma variedade de taxas de transmissão (TEKTRONIX. . . , 2001).

2.3.1 O Sinal SONET/SDH

O Padrão SONET estabelece uma tecnologia de transmissão de sinais com diferentes capacidades através de uma hierarquia ótica síncrona e flexível. Para realizar o envio de sinais em diferentes capacidades, é preciso um esquema de multiplexação por intercalação de *bytes*. Isso simplifica a multiplexação e oferece um gerenciamento de ponta a ponta para as redes. O bloco básico de construção e o primeiro nível da hierarquia do sinal SONET, são denominado de sinal de transporte SONET – nível 1 (STS-1), operando a 51,84 Mb/s. Enquanto os sinais SONET com maior taxa são adquiridos pela intercalação de *bytes* de N desses quadros STS-1 que são codificados e alterados para um sinal de portador óptico de nível N (OC-N). Posteriormente, o sinal OC-N definirá uma transmissão de taxa igual a N vezes a de um sinal OC-1 (KEISER, 2014) .

A Tabela 1 apresenta o equivalente ótico para cada sinal STS-N, chamado de OC-N (OC - *Optical Carrier*, Portadora Ótica), a equivalência entre os sinais do SONET e os sinais do SDH, seguido pela largura de banda da carga útil (a informação transportada) e a taxa total da linha de transmissão.

Tabela 1 – SONET/SDH Designações e Larguras de Banda

Nível de Portadora Ótica para o SONET	Formato do Quadro para o SONET	Formato do Quadro para o SDH	Largura de Banda da "Carga Útil" (Payload) (kbit/s)	Taxa de Linha(kbit/s)
OC-1	STS-1	STM-0	48.960	51.840
OC-3	STS-3	STM-1	150.336	155.520
OC-12	STS-12	STM-4	601.344	622.080
OC-24	STS-24	STM-8	1.202.688	1.244.160
OC-48	STS-48	STM-16	2.405.376	2.488.320
OC-96	STS-96	STM-32	4.810.752	4.967.640
OC-192	STS-192	STM-64	9.621.504	9.953.280
OC-738	STS-768	STM-256	38.486.016	39.813.120
OC-1536	STS-1536	STM-512	76.972.032	79.626.120
OC-3072	STS-3072	STM-1024	153.944.064	159.252.240

2.3.2 Elementos da Rede SONET/SDH

Uma rede SONET poderá ser composta pelos seguintes elementos:

(a) Regeneradores: O processo de regeneração tem a função de tratar o sinal que foi degradado, para isso é necessário adequação nas características de amplitude, formas de onda e de sincronização adequadas aos limites especificados para a sua retransmissão. Geralmente, esse dispositivo é necessário, quando o nível de sinal da fibra ótica se torna muito baixo devido à longa distância.

(b) Multiplexador Terminal: Tem a função de concentrar os sinais digitais de baixo nível. Frequentemente é utilizado em rotas ponto a ponto, ou, nas terminações de rotas em barramento, possui apenas uma interface de agregado e possibilita a inserção (*add*) ou retirada (*drop*) de tributários de diversas hierarquias.

(c) Multiplexador *Add/Drop* (ADM): Tem a capacidade de acessar qualquer um dos sinais formados do sinal agregado STM-N, sem demultiplexar e interromper o sinal completo. Esse multiplexador é utilizado em pontos intermediários de rotas em barramento, ou, nas rotas em anel. Na localidade que possui uma ADM, pode retirar sinais de taxas baixas, para serem transportados em diferentes equipamentos, ou podem ser adicionados sinais de taxas baixas, para dentro de sinais de STS-N de taxas mais altas. O padrão SONET/SDH não restringe fabricantes a um único tipo de produto, nem requer que eles fabriquem todos os tipos. Isso implica que podem existir ADMs diferentes (com níveis de sinais diferentes).

(d) Digital Cross Connect/Digital Cross Connect System (DXC or DCS): É responsável por realizar *grooming*, que é (de)multiplexar tráfegos de mais baixa velocidades em tráfegos de alta velocidade (este tráfego de mais baixa velocidade pode ser tanto fluxo PDH como SDH/SONET). A maior diferença entre um DXC e um ADM, é que um DXC pode ser usado para interconectar um número maior de sinais STS-1. O

DXC pode ser usado para consolidar ou segregar sinais STS-1, ou para gerenciamento de tráfego de banda muito larga.

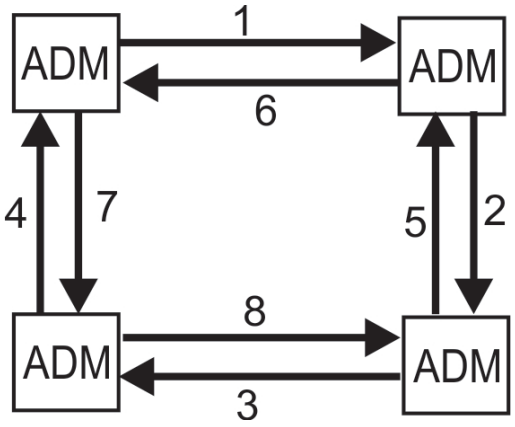
2.3.3 Topologias do Padrão SONET/SDH

Existem várias topologias de redes para serem utilizadas, a escolha depende da necessidade do projeto e da tecnologia utilizada. No caso do padrão SONET/SDH, as topologias disponíveis são: anel, ponto a ponto, ponto multiponto e barramento.

A topologia em anel se destaca no padrão SONET/SDH pela alta confiabilidade. Os anéis são comumente conhecidos como *self-healing* (auto curável), pela capacidade automática de restauração, após uma falha ou deterioração dos sinais da rede. Para que os serviços do anel possam ser restaurados automaticamente, após uma falha, utiliza-se um protocolo chamado APS (*Automatic Protection Switching*), que restaura os serviços com menos 50 milissegundos (PERROS, 2005).

As configurações do anel SONET/SDH, podem consistir em duas ou quatros fibras. Na figura 4, o anel possui duas fibras, nas quais, 1, 2, 3 e 4 são utilizadas para formar o anel de trabalho, que possui o fluxo em sentido horário, enquanto as fibras 5, 6, 7 e 8 são utilizadas para formar o anel de proteção e funciona no sentido anti-horário, ou seja, caso o cabo de fibra ótica romper ou falhar, os multiplexadores enviam automaticamente os serviços afetados por um caminho alternativo. Com isso, não haverá a interrupção dos serviços de comunicação da rede.

Figura 4 – Anel Sonet/SDH



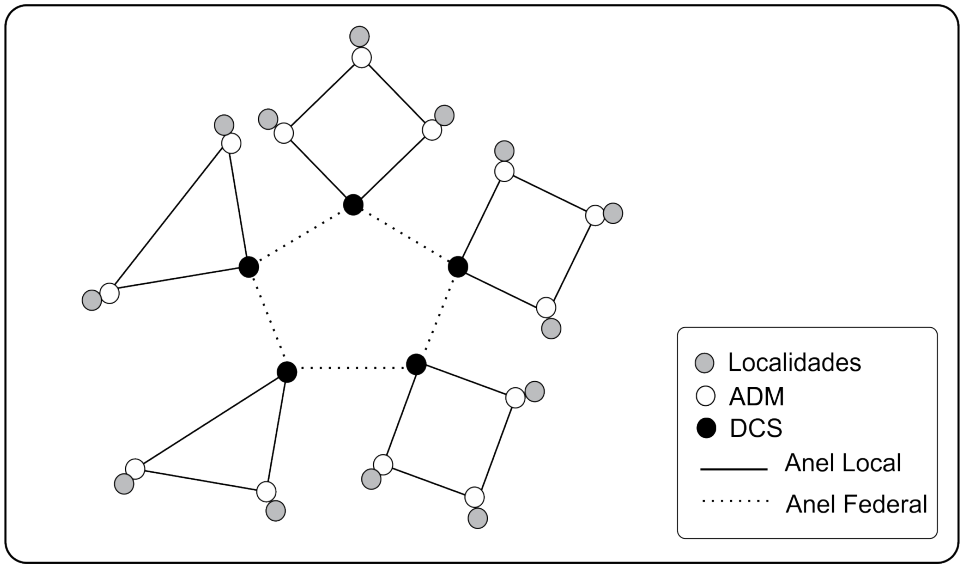
Fonte: Perros (2005)

Os anéis podem ou não realizar uma comunicação entre si. Caso essa comunicação ocorra, um dispositivo *Digital Cross-Connect System* (DCS) liga os anéis locais a um anel especial denominado Anel Federal (AF).

A figura 5 mostra uma rede SONET segundo a topologia em anel. Cada ADM conecta uma localidade à rede, através de um anel local. O tráfego entre os anéis locais, é

feito através dos DCSs, no centro, que compõem o anel federal. As restrições estão melhor detalhadas na seção 2.4.

Figura 5 – Rede SONET/SDH com topologia em anel



Adaptado: Oliveira (2010)

2.4 Descrição do Problema

Nesta seção, descreveremos o problema de Atribuição de Localidades a Anéis em Redes SONET/SDH (PALAS) que surge na etapa do planejamento físico de uma rede *backbone*, ou seja, na etapa em que o projetista determina os subconjuntos de localidades, que constituirão os anéis SONET.

O custo principal de uma rede SONET, está diretamente relacionado ao custo dos componentes ADMs e DCSs, já que o cabos de fibra ótica, possuem graus bem menores de relevância; como os ADMs são consideravelmente mais baratos que os DCSs, a quantidade de DCSs, tornam-se o fator principal para o custo da rede.

É possível projetar algoritmos para a resolução do PALAS com diferentes conjuntos de restrições, cujo objetivo, é minimizar o custo total da rede. Para isso, é necessário considerarmos as seguintes restrições:

(i) Cada localidade tem que ser alocada a um único anel;

(ii) A capacidade máxima de cada anel é limitada por um valor comum, representado por B;

(iii) Minimizar o número de anéis locais, que correspondem ao custo com os DCSs instalados, pois para cada anel local é necessário um DCS.

O PALAS pode ser descrito como um problema de particionamento dos vértices de um grafo, em que, os vértices representam os clientes da rede, e os pesos das arestas indicam as demandas de tráfego entre eles.

2.4.1 Modelo Matemático

Na literatura existem diversas formulações matemáticas propostas para o PALAS. Sendo (GOLDSCHMIDT et al., 2002) e (MACAMBIRA; MACULAN; SOUZA, 2005) as mais difundidas. Esta última, apresentada em seguida, trata-se de um modelo mais compacto, no qual foi desenvolvido com intuito de reduzir o número de variáveis, mantendo os mesmos números de restrições.

A formulação (2.1)-(2.12) utiliza o grafo G = (V, E) completo não direcionado. Tendo as demandas de tráfego $d_{ij} > 0$ entre cada par de vértice i, j $\in$ V. O total do tráfego é dado por D= $\sum_{[i,j]\in E} d_{ij}$. A constante B $\in \mathbb{N}^*$ representa a capacidade máxima de cada anel, inclusive a do anel federal.

Esse modelo utiliza a variável y^r que determina se o anel r é usado ($y^r = 1$) ou não ($y^r = 0$) e a variável x_i^r que estabelece se um vértice i é alocado no anel r (x_i^r=1) ou não (x_i^r=0). A variável de decisão auxiliar z_{ij}^r que indica se pelo menos uma das extremidades da aresta $[i, j]$ pertence ao anel r (z_{ij}^r=1), ou não (z_{ij}^r=0).

A função objetivo (2.1) estabelece a maximização do número de clientes incluídos nos anéis e a minimização da quantidade de anéis.

$$max \sum_{r=1}^{n} \sum_{i \in V} n x_i^r - \sum_{r=1}^{n} y^r \tag{2.1}$$

As funções (2.2) e (2.3), são restrições que asseguram, respectivamente, o respeito a capacidade dos anéis locais e federal.

$$\sum_{[i,j]\in E} d_{ij} z_{ij}^r \leqslant B \quad \forall i = 1...n \tag{2.2}$$

$$\sum_{r=1}^{n} \sum_{[i,j]\in E} d_{ij} z_{ij}^r \leqslant D + B \tag{2.3}$$

A inequação (2.4) garante que cada vértice seja alocado a no máximo um anel

$$\sum_{r=1}^{n} x_i^r \leqslant 1 \quad \forall i \in V \tag{2.4}$$

A restrição (2.5) garante que os vértices não sejam alocados a anéis que não existem.

$$x_i^r \leqslant y^r \quad \forall i \in V, \forall r = 1...n \tag{2.5}$$

As variáveis x e z são conectadas através das restrições (2.6), (2.7), (2.8).

$$z_{ij}^r \geqslant x_i^r \quad \forall [i,j] \in E, \forall r = 1...n \tag{2.6}$$

$$z_{ij}^r \geqslant x_j^r \quad \forall [i,j] \in E, \forall r = 1...n \tag{2.7}$$

$$z_{ij}^r \leqslant x_i^r + x_j^r \quad \forall [i,j] \in E, \forall r = 1...n \tag{2.8}$$

As inequações (2.9) reduzem o fenômeno de simetria das variáveis y, evitando que um anel seja aberto, se seu predecessor encontra-se fechado.

$$y^r \leqslant y^{r-1} \quad \forall r = 2...n \tag{2.9}$$

Para finalizar, são definidas as variáveis x, y, z nas equações (2.10) a (2.12).

$$x_i^r \in \{0,1\} \quad \forall i \in V, \forall r = 1...n \tag{2.10}$$

$$y_r \in \{0,1\} \quad \forall i \in V, \forall r = 1...n \tag{2.11}$$

$$z_{ij}^r \in \{0,1\} \quad \forall [i,j] \in E, \forall r = 1...n \tag{2.12}$$

2.4.2 Revisão de Literatura

Essa seção faz a revisão sobre os trabalhos publicados nos últimos anos que tratam do problema PALAS. Em todos os trabalhos, mencionados nessa revisão, utilizaram as instâncias da literatura conhecidas como: GLO, AD, LSHK e RTNR, referenciadas respectivamente como C1, C2, C3 e C4.

Em 2001, a resolução do PALAS foi realizada pela metaheurística Busca Tabu em (ARINGHIERI; DELL'AMICO, 2001). O algoritmo foi construído com quatro funções objetivo (z1, z2, z3, z4), uma vizinhança *Node Improvement Neighborhood* (N1) e duas listas tabu (*node-list* e *node-from-list*). Nesse mesmo trabalho, foi proposto a utilização do TSSO (*Tabu Search with Strategic Oscillation*), além das técnicas de intensificação e diversificação: *Path Relinking* (GLOVER, 1998), *Exploring Tabu Search* (DELL'AMICO;

TRUBIAN, 1998) e *Scatter Search* (GLOVER, 1999). Nos testes realizados, os autores utilizaram as mesma instâncias do trabalho (GOLDSCHMIDT; LAUGIER; OLINICK, 2003) (as instâncias da classe C1) e obtiveram as soluções ótimas para todas as instâncias. Concluíram, naquele momento, que o TSSO tem superioridade nas comparações com os outros métodos.

Em (GOLDSCHMIDT; LAUGIER; OLINICK, 2003), os autores revelaram que o PALAS é um problema NP-Difícil. Para a resolução do problema, disponibilizaram três heurísticas gulosas: *edge-based, cut-based* e *node-based*. Os testes das heurísticas, foram realizados nas instâncias da classe C1, com no mínimo 15 e no máximo 50 localidades. Os resultados das heurísticas foram comparados com os resultados dos métodos exatos, e, demonstraram que as heurísticas, obtiveram um bom desempenho. Os autores tiveram cerca de 97% das instâncias com soluções ótimas.

Os autores Aringhieri e Dell'Amico continuaram com novas proposta para PALAS, dessa vez, foi no trabalho (ARINGHIERI; DELL'AMICO, 2005) no qual desenvolveram uma técnica de vizinhanças múltiplas, denominada DMN (*Diversication by Multiple Neighborhhods*). O DMN tentando diversificar as soluções utilizou as vizinhanças N1 e N2. A vizinhança N1, move um cliente de um anel para outro anel. Já a vizinhança N2, pratica uma troca entre dois clientes de anéis distintos em dois movimentos, onde ele move um cliente para outro anel, provavelmente se tornando inviável e depois, outro movimento, buscando se tornar viável, o objetivo foi diversificar as soluções. Nesse trabalho, os autores compararam a nova técnica de vizinha, com todas as técnicas desenvolvida no trabalho anterior (ARINGHIERI; DELL'AMICO, 2001). Os resultados mostraram que o procedimento DMN, encontrou a solução ótima em 100% das instâncias das classes C1 e C3, já para classe C2 encontrou a solução ótima em 98,2% das instâncias. Nesse trabalho os autores obtiveram melhores resultados que os outros anteriores.

Na obra de (MACAMBIRA, 2003) são apresentadas várias formulações de programação linear inteira para o problema do PALAS. Além disso, foi proposto um novo modelo matemático para o problema de atribuição de localidades a anéis. Este modelo, é baseado no princípio de empacotamento, em particular, o empacotamento das localidades da rede. O trabalho realizou um estudo, sobre o problema de simetria entre as soluções do problema de atribuição de localidade a anéis. Depois, apresentaram os experimentos computacionais com o algoritmo *branch-and-price*, e, obteve de forma exata, ótimos resultados para instâncias de até 50 localidades. Em (MACAMBIRA; SOUZA, 2003) encontra-se uma metaheurística GRASP, utilizada na fase de inicialização do algoritmo *branch-and-price*, proposto por Macambira em (MACAMBIRA, 2003), para a obtenção de limitantes superiores (iniciais) para o Problema de Atribuição de Localidades a Anéis.

Os autores Bastos, Ochi e Macambira desenvolveram uma nova versão do GRASP denominada Algoritmo Construtivo, baseado em Vizinhança Relativa (BASTOS; OCHI;

MACAMBIRA, 2005a). Já no trabalho (BASTOS; OCHI; MACAMBIRA, 2005b) foram apresentado um estudo, comparando os resultados do GRASP PURO (GP) e GRASP com *Path-Relinking* (PR) no qual os resultados mostram que, em alguns casos, o GRASP com PR convergiu para a solução ótima mais rápido que GP. Os experimentos desses algoritmos utilizaram as instâncias da classe C1 e C2.

Em (SILVA, 2008), foi realizado uma pesquisa para resolução do PALAS com as seguintes metaheurísticas: Algoritmo Genético (AG), Algoritmo Memético (AM) e outra híbrida de AM com *Vocabulary Building* (AM+VB). Para geração das populações iniciais, o algoritmo genético utilizou as versões das heurísticas *Edge-Based e Cut-Based* implementadas por (BASTOS, 2005), o método da Roleta para fazer a seleção, elitismo e BPXcrossover, para reprodução dos indivíduos. Enquanto o AM foi composto através de uma nova versão do AG que empregou busca sobre duas vizinhas: N1 e N3. Verificou-se que o algoritmo AM+VB foi o que apresentou maior número de instâncias resolvidas de forma ótima para as duas classes testadas (C1 e C2).

O trabalho de (SOARES, 2009) foi implementado uma metaheurística híbrida, algoritmo Busca Tabu (BT) com *Vocabulary Building*, com vocábulos sendo gerados de três formas: A primeira, a partir de um conjunto de soluções elite; a segunda, de forma aleatória; e a terceira, a partir das duas anteriores, onde cada uma, produziu metade dos vocábulos. Dos três métodos utilizado pelo vocábulo, os oriundos das soluções elite foram bastante eficientes na melhoria das soluções do conjunto de soluções elite e, por sua vez, da melhor solução encontrada. O algoritmo demonstrou aumento na robustez, além de aumentar a quantidade de ótimos com a utilização da técnica de *Vocabulary Build*. A BT apresentou um bom desempenho, atingindo os ótimos da grande maioria das instâncias C1 e C2.

Posteriormente, em (OLIVEIRA, 2010), foram apresentados algoritmos evolutivos, que juntamente com a técnica Construção de Vocabulário e da programação paralela, foram aplicados na resolução do PALAS. Foram implementadas duas metaheurísticas puras: o Algoritmo Genético (AG) e Algoritmo Memético (AM), e uma híbrida, sendo esta, implementada em uma versão sequencial, o Algoritmo Memético com Construção de Vocabulário, e uma versão paralela, o Algoritmo Memético Paralelo com Construção de Vocabulário. Com os algoritmos propostos, foram realizados extensivos experimentos computacionais sobre as instâncias disponíveis na literatura, as classes C1, C2 e C4. A versão paralela foi a que apresentou os melhores resultados, principalmente em termos de tempo computacional, pois, em relação à qualidade das soluções, a versão sequencial e paralela obtiveram resultados similares.

Mais recente em (OLIVEIRA, 2015), foi utilizado o Algoritmo Genético de Chaves Aleatórias Viciadas e uma técnica de programação dinâmica chamada *Split*, que consiste em uma técnica de decomposição do problema em subproblemas independentes, com

decisões multiestágio, para resolução do PALAS. O trabalho desenvolveu a metaheurística BRKGA, com um decodificador, que também, utilizou técnicas de programação dinâmica para avaliar a viabilidade da solução do cromossomo. Para este trabalho, foram utilizados 3 (três) conjuntos de instâncias C1, C2, C3. O algoritmo de programação dinâmica, chamado de *Multi-Start*, mostrou-se eficaz na obtenção da solução ótima para os conjuntos de instâncias C1 e C2. O trabalho conclui, que, com exceção do conjunto C3, o *Multi-Start* e o BRKGA, conseguiram obter soluções ótimas para todas as instâncias, sendo diferenciados pelo tempo de execução, onde o *Multi-Start* demonstrou vantagem.

3 Abordagens Heurísticas

A Pesquisa Operacional, é definida como uma área de conhecimento que estuda soluções de problemas reais, onde geralmente, são inviáveis a utilização dos métodos exatos para obtenção da solução. Isso acontece porque, conforme aumentam os tamanhos da instância, o tempo de resposta exigido pelos métodos exatos, sobem de modo exponencial. De maneira que há necessidade de se procurar outros métodos não exatos, mas, que, permitem obter uma solução próxima do ótimo em tempo polinomial. Dessa forma, a alternativa é a utilização de heurísticas (heurísticas, metaheurísticas, heurísticas híbridas, matheurísticas e hiper-heurísticas) que possibilitam encontrar soluções sub-ótimas viáveis, eventualmente ótimas. Este capítulo, tem como objetivo apresentar a metaheurística evolucionária BRGKA.

3.1 Metaheurísticas

Incialmente, temos a heurística clássica, que é definida como uma técnica de busca local, que procura boas soluções em um tempo computacional viável, sem garantia de fornecer uma solução no ótimo global. Uma das grandes desvantagens de uma heurística é a sua dificuldade em fugir do ótimo local, pois o objetivo é encontrar o ótimo global.

Nesse contexto, surgiram as metaheurísticas: métodos de solução, que comandam uma interação entre processos de melhoria locais e estratégias inteligentes, para escapar de ótimos locais, realizando uma busca no espaço de soluções do problema, com objetivo de encontrar o ótimo global (GLOVER; KOCHENBERGER, 2006). As metaheurísticas, são usadas para resolver problemas de otimização, que não possuem solução exata em tempo polinomial. De forma resumida, o que diferencia as metaheurísticas das heurísticas convencionais, é que elas são genéricas e podem ser capazes de fugir de um ótimo local. Nesse trabalho, será utilizada a metaheurística BRKGA que terá suas definições nas próximas seções.

3.1.1 Metaheurísticas Híbridas

A junção de metaheurísticas com outras técnicas de otimização, são chamadas de metaheurísticas híbridas, gerando como principais vantagens, um comportamento mais eficiente e uma maior flexibilidade para lidar com os problemas do mundo real e de larga escala. Metaheurísticas híbridas, buscam combinar de modo complementar as forças de cada método heurístico utilizado (BLUM; ROLI, 2008).

Alguns exemplos de trabalhos que utilizaram metaheurísticas híbridas para obtenção

de melhores resultados: (POMARI; CHAVES, 2014), (OLIVEIRA, 2010),(NASCIMENTO; ALOISE, 2014), (OLIVEIRA, 2015)

Propõe-se dessa forma, uma abordagem híbrida, usando o BRKGA, com a técnica Vocabulary Build, além de outra versão que utiliza o *Q-learning* para geração da população inicial, sendo explicadas na próximas seções, para obtenção de melhores resultados.

3.2 Algoritmo Genético de Chaves Aleatórias Viciadas

Este trabalho utiliza uma variante do Algoritmo genético, denominado por Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA, do inglês *Biased Random-Key Genetic Algorithm*), proposta por (GONÇALVES; RESENDE; MENDES, 2011). É bom salientar, que o BRKGA foi derivado do RKGA, do inglês *Random-Key Genetic Algorithm*) (BEAN, 1994).

Os principais fatores que motivaram a escolha desta metaheurística, foram: (i) A flexibilidade que esse método possui para trabalhar com problema de otimização combinatório; (ii) A comprovação que BRKGA é superior aos algoritmos genéticos tradicionais, que utilizam a combinação e a mutação de acordo com a teoria básica dos algoritmos genéticos (GONÇALVES; RESENDE; MENDES, 2011) e os bons resultados da literatura que foram apresentados em alguns trabalhos, como por exemplo: (GONÇALVES; RESENDE, 2015), (ROQUE; FONTES; FONTES, 2011), (CHAN et al., 2013); (iii) Possuir uma documentação detalhada e suporte para implementação através da disponibilidade de *framework* na linguagem C++ e Python (SILVA; RESENDE; PARDALOS, 2015).

O BRKGA tem seu funcionamento semelhante ao Algoritmo Genético tradicional, porém, apresenta particularidades que serão apresentadas. A primeira diferença, é que o BRKGA realiza a codificação de uma possível solução em um vetor de tamanho n, sendo cada vetor de solução, gerado aleatoriamente a partir do intervalo real [0, 1]. Essa etapa é repetida até completar a população inicial. Em seguida, o decodificador realiza o cálculo da função objetivo associando a cada indivíduo um valor. O algoritmo compõe-se das seguintes etapas:

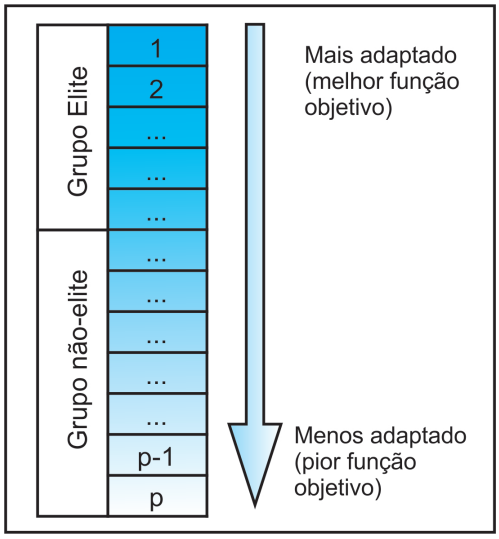
- A população da k-ésima geração, é particionada em dois grupos de indivíduos, ver Figura 6, tendo um grupo de p_e indivíduos elite (conjunto com os melhores valores da função objetivo) e o conjunto restante dos $p - p_e$ indivíduos não-elite.
- O BKKGA utiliza o elitismo para geração da nova população, ou seja, a evolução desta geração k-ésima, terá todos os indivíduos elite da população copiados sem modificações para a população da geração k+1-ésima.

- Para completar a população k+1-ésima, o algoritmo realiza a criação dos mutantes. O BRKGA no processo de mutação realiza a criação de novos elementos p_m, usando a mesma estratégia que foi utilizada na geração da população inicial, porém, com uma quantidade menor. Dessa forma, o algoritmo adiciona os elementos p_m para nova geração k+1-ésima.

- Para completar a população da geração k+1-ésima, são gerados novos indivíduos utilizando os métodos de cruzamento.

Essas etapas mostradas anteriormente são repetidas iterativamente e cada iteração é chamada de uma geração. O processo é repetido até satisfazer um dos critérios de parada a seguir: parar após executar um número de iterações previamente especificado, parar o processo se não melhora após um número de iterações e parar o processo após atingir um limite de tempo de processamento.

Figura 6 – Agrupamento de uma população de p indivíduos em certa geração k



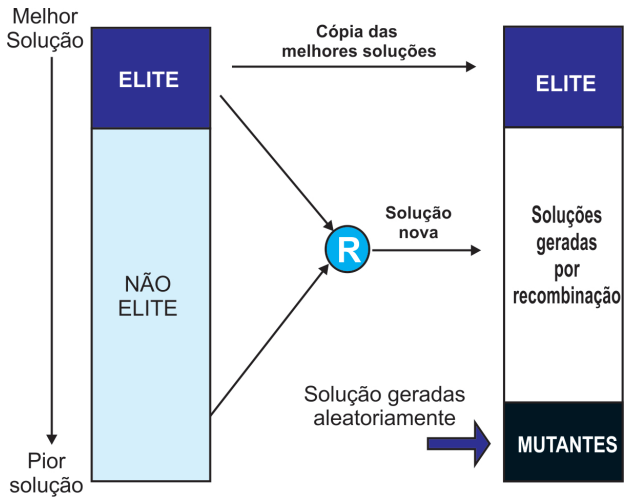
Fonte: Mainieri (2014)

O decodificador é um algoritmo determinístico que é responsável em receber uma possível solução codificada, para encontrar uma solução factível do problema original, ou seja, o decodificador transforma uma vetor codificado para uma solução viável do problema (UZINSKI, 2014).

Para realizar o cruzamento, o RKGA seleciona dois pais aleatoriamente de toda população. Essa estratégia possibilita que soluções de qualidade diversificada, sejam escolhidas aleatoriamente, já que não é usado o valor da função objetivo para escolher os pais da novas soluções. Nesse ponto foi que surgiu a proposta que possibilitou a criação do BRKGA apresentada em (GONÇALVES; RESENDE; MENDES, 2011).

O BRKGA tem um pai que é sempre escolhido aleatoriamente no conjunto elite e outro escolhido do conjunto não-elite. A figura 7 mostra a formação da nova população k+1 do BRKGA a partir da população k. Assim que surgiu a palavra "viciada"(do inglês Biased) no BRKGA, que foi definida pelos autores (GONÇALVES; RESENDE; MENDES, 2011), para definir que a recombinação de novos indivíduos não é puramente aleatórias, ou seja, uma das soluções geradoras deve ser necessariamente da população elite (viciada).

Figura 7 – Montagem de uma geração



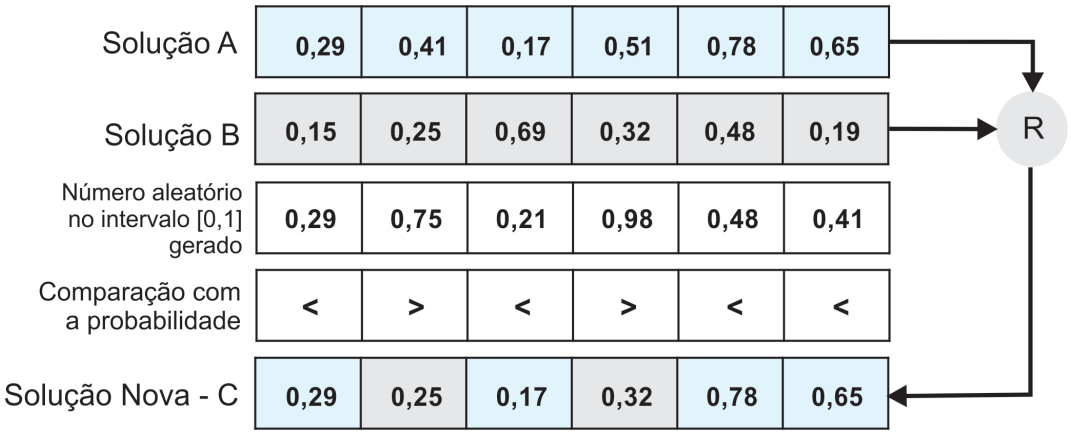
Fonte: Adaptado de Silva et al. (2012)

Ambos os algoritmos, utilizam a recombinação de pais com o cruzamento uniforme parametrizado, mais conhecida como recombinação PUC, do inglês *Parametrized Uniform Crossover*, que foi elaborada por (SPEARS; JONG, 1995) para geração dos descendentes. O funcionamento dessa recombinação PUC para o BRKGA, se inicia com a seleção de um indivíduo A, do conjunto solução elite e o indivíduo B, da população não elite para participar da combinação. O PUC possui um parâmetro ρ_a que representa a probabilidade de uma prole herdar um elemento (alelo) de vetor de seu pai elite, nesse caso, o indivíduo A. Então, para formar uma descendente C, gera-se aleatoriamente, um número $\rho \in [0,1]$ que será comparado com ρ_a. Se $\rho \leq \rho_a$ então, é copiado o valor do alelo armazenado no individuo A, caso contrário, copia-se o valor do alelo do indivíduo B. Geralmente o valor de ρ_a utilizado é $\rho_a = 0,7$. Dessa forma, aumenta a probabilidade do descendente C possuir uma grande quantidade de alelos da solução elite. Na figura 8, é mostrada com mais detalhes como é gerado um descendente usando a combinação PUC.

Nota-se, que a diferença entre ambos os algoritmos é pequena, porém, existe uma grande diferença de desempenho entre ambos algoritmos, sendo que o BRKGA, encontra soluções de qualidade muito melhores quando se considera o mesmo tempo de processamento.

No fluxograma do BRKGA, mostrado na figura 9, pode-se notar que existem

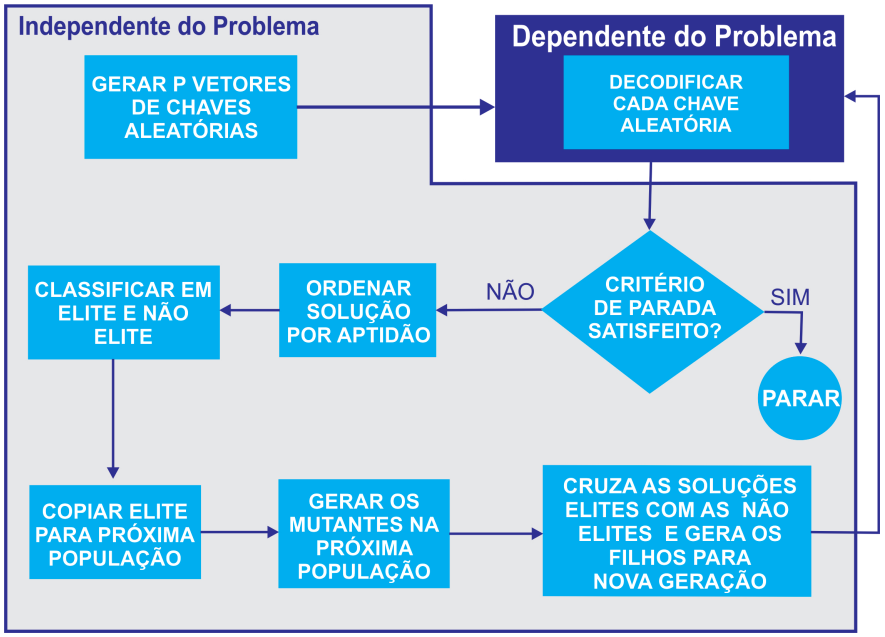
Figura 8 – Recombinação PUC, tendo $\rho_a = 0,7$



Fonte: Adaptado de Gonçalves, Resende e Mendes (2011)

duas partes muito bem diferenciadas, uma parte independente do problema, aplicável a qualquer tipo de problema e outra parte que depende do tipo de problema que se pretende resolver. Assim, a parcela independente do BRKGA pode ser implementada uma única vez e depois reaproveitada para resolver outros problemas. Então, quando pretende-se aplicar o BRKGA para outro tipo de problema será necessário apenas implementar o decodificador para o problema em questão.

Figura 9 – Fluxograma do Algoritmo BRKGA



Fonte: Gonçalves, Resende e Mendes (2011)

3.2.1 Parâmetros do BRKGA

No BRKGA possui alguns parâmetros que precisam ser escolhidos e calibrados. Os parâmetros, são o tamanho do vetor de codificação de chaves aleatórias (n), o tamanho da população (n_{pop}), o número das soluções elite (n_e), o número das soluções mutantes (n_m) e a probabilidade (ρ_e) de um descendente incorporar a informação do pai elite. Na Tabela 2, são apresentados valores recomendados desses parâmetros que são apresentados por (GONÇALVES; RESENDE; MENDES, 2011). Esses valores dos parâmetros foram obtidos através de experimentos em vários tipos de problemas.

Tabela 2 – Parâmetros BRKGA com valores recomendados

Parâmetros	Especificação	Valores recomendados
n_{pop}	Tamanho da população	n_{pop} = a x n onde $1 \leqslant a \in \mathbb{R}$ é uma constante que depende de n comprimento de uma proposta de solução codificada pelo vetor de chaves aleatórias
n_e	Tamanho das soluções de elite	0,10 $n_{pop} \leqslant n_e \leqslant$ 0,25 n_{pop}
n_m	Tamanho das soluções de mutantes	0,10 $n_{pop} \leqslant n_m \leqslant$ 0,30 n_{pop}
ρ_e	Probabilidade de herdara informação da solução do pai elite	0,50 $n_{pop} \leqslant \rho_e \leqslant$ 0,80 n_{pop}

4 Algoritmos Propostos

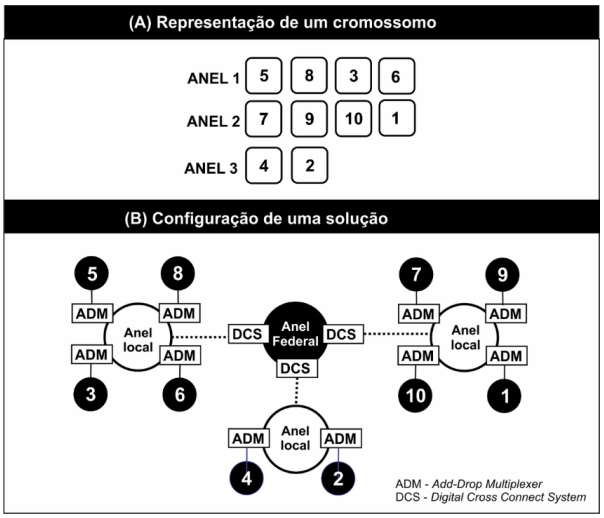
Neste capítulo serão apresentados os detalhes das implementações dos algoritmos propostos para o Problema de Atribuição de Localidades a Anéis SONET/SDH. O trabalho aborda o desenvolvimento dos seguintes algoritmos: Algoritmo Genético de Chave Aleatórias Viciada (BRKGA), BRKGA com *Q-learning* (BR+QL), BRKGA com *Vocabulary Build* (BR+VB), tendo uma versão em que a geração dos vocábulos é feita de forma aleatória e outra versão que utiliza o *Q-learning* para gerar os vocábulos.

4.1 Algoritmo Genético de Chaves Aleatória Viciadas Aplicadas ao PALAS

4.1.1 Representação do Cromossomo

Foram utilizadas listas de vetores para representar os cromossomos dos indivíduos do BRKGA implementado. Conforme visto, uma solução do PALAS é uma partição de um conjunto de n localidades em k subconjuntos disjuntos. Dessa forma, cada vetor da lista que constitui o cromossomo, contém as localidades que formam determinado anel da solução. A figura 10, inicialmente, na parte A, mostra uma representação dos anéis que são armazenados em estrutura agrupadas, enquanto já parte B, mostra na perspectiva da alocação das localidades (clientes) junto com os equipamentos (ADM e DCS) da rede SONET.

Figura 10 – (A) Cromossomo , (B) Configuração de uma solução



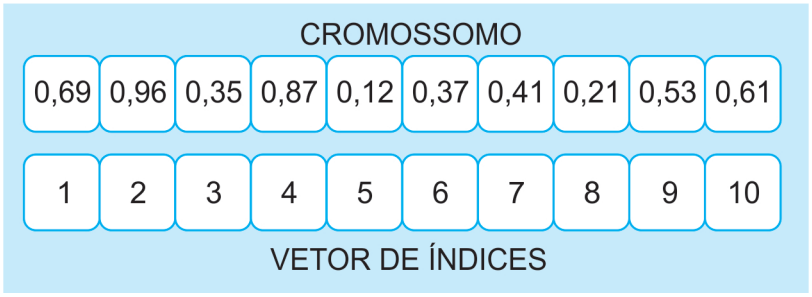
Fonte: Autor

4.1.2 Decodificador

Como foi visto, o BRKGA gera cromossomos, no quais, cada chave do vetor é gerada aleatoriamente no intervalo [0,1]. E para a verificação da qualidade do cromossomo, é necessário realizar a decodificação. Esse processo de decodificação, consiste em um método heurístico que tem como entrada um cromossomo de números aleatórios e como saída, fornece uma solução viável.

Para o nosso algoritmo, a interpretação das chaves aleatórias foi feita com o uso de um vetor auxiliar, associado ao cromossomo. A figura 11, mostra um cromossomo em seu estado natural, sem qualquer alteração, e um vetor de índices associado a ele, sendo que, o primeiro elemento do vetor de índices é ligado ao primeiro elemento do cromossomo, o segundo elemento do vetor de índices é ligado ao segundo elemento do cromossomo e, assim, sucessivamente.

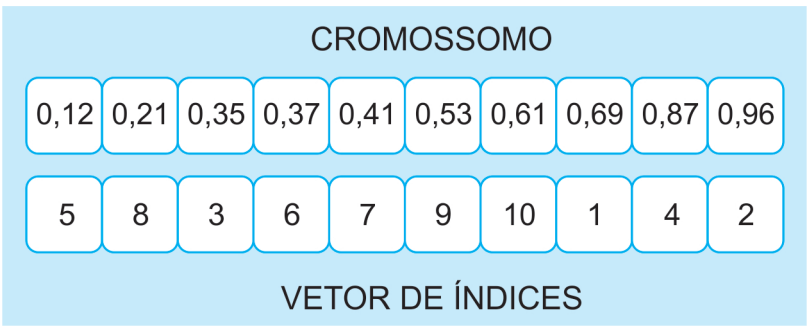
Figura 11 – Cromossomo e Vetor de Índice



Fonte: Autor

Na figura 12, é exemplificado a ordenação do cromossomo de forma crescente, com base em seus valores numéricos e o vetor de índices foi permutado por seguir o comportamento do cromossomo.

Figura 12 – Cromossomo ordenado e Vetor de Índice permutado



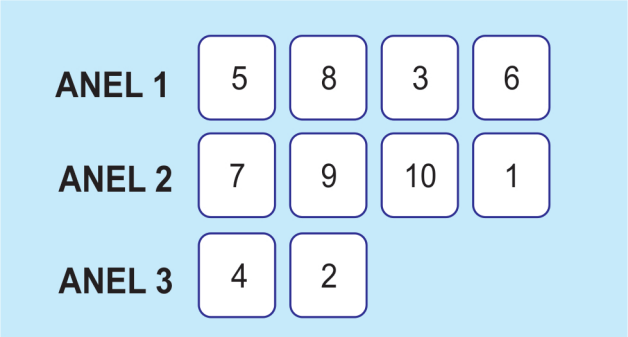
Fonte: Autor

Em seguida, o decodificador vai distribuindo para os vetores as localidades, na sequência que foi ordenado. Sempre verificando as restrições imposta pelo problema do

PALAS. Caso uma localidade não seja mais suportada por um anel, será criado outro vetor (representando o anel) para adicionar a localidade, isso se repete até que todas as localidades estejam adicionadas em algum vetor.

A figura 13, monstra a distribuição do cromossomo, verifica-se que foi preciso de 3 vetores para adicionar todas as localidades, de forma que todos os anéis sejam viáveis. Nota-se assim, que o anel federal não precisa ser representado por um vetor, uma vez que não existem localidades atribuídas a ele.

Figura 13 – Representação de um Cromossomo após a decodificação



Fonte: Autor

4.1.3 Função Aptidão

Para se ter uma solução viável no PALAS, é preciso que todos os anéis locais e anel federal sejam viáveis, ou seja, se a demanda total de cada anel for menor ou igual a B. Percebe-se, que, podemos ter uma solução viável com o mesmo número de anéis de uma solução inviável. Logo, é necessário desenvolver uma função de aptidão para diferenciar soluções viáveis de soluções inviáveis. Como uma solução viável é sempre melhor que uma solução inviável, acrescentou-se um fator de penalidade nas soluções inviáveis, esse fator é a demanda do anel federal. Desta forma, seja S uma solução qualquer para o PALAS, a função de aptidão é dada por:

$$F(S) = \begin{cases} B \cdot K(S) & \text{Se S for viável} \\ B \cdot K(S) + DemAF(S) & \text{Se S for inviável} \end{cases} \tag{4.1}$$

A fórmula 4.1 tem $K(S)$ que representa uma função que retorna o número de anéis locais de uma solução S, $DemAF(S)$ uma função que retorna a demanda no anel federal de uma solução S e B, é a capacidade máxima de cada anel, inclusive a do anel federal.

4.2 Algoritmo BRKGA com *Q-learning* Aplicado ao PALAS

Nesta seção, será apresentado a utilização do BRKGA agregado com o algoritmo *Q-learning*, afim de utilizar os conceitos de aprendizagem por reforço, para construir melhores soluções do Problema de Atribuição de Localidades a Anéis SONET.

A proposta, é utilizar o algoritmo *Q-learning* apenas como construtor da população inicial de alta aptidão e com boa taxa de diversidade para o algoritmo BRKGA. Para realizar a aprendizagem do algoritmo *Q-learning*, é necessário usar uma política para escolha de ação, que pondere entre usar o conhecimento já obtido e escolher novos estados ainda não explorados, ou seja, o algoritmo tem característica guloso e aleatório, porque usa a ação com valor máximo obtido pela função Q(s,a) e aleatório, ao usar a política de escolha de ações, que possibilite visitar a todos os demais estados do ambiente.

O agente aprendiz utiliza os conhecimento de aprendizagem por reforço, para buscar maximizar o total de recompensa recebida durante o processo. Para isso, é necessário desenvolver uma função de recompensa, que possa mensurar quão bom é escolher uma determinada ação. Para o PALAS, é necessário criar um função de reforço que possa equilibrar o fluxo entre os anéis e diminuir o fluxo do anel federal. Para isso, no momento da formação do anéis, terão maiores recompensas os clientes que possuem maiores ligações internas, pois diminuirá o fluxo externo, e, consequentemente deixará o fluxo do anel federal menor.

Uma ideia semelhante foi aplicada em (Lima Júnior, 2009) que utilizou o *Q-learning* como heurística guloso-aleatória, como gerador de população inicial para o Algoritmo Genético aplicado ao Problema do Cacheiro Viajante. Outra inovação proposta nesse trabalho, foi a atuação cooperativa entre o algoritmo *Q-learning* e os operadores genéticos, dentro os quais, apresentaram ótimos resultados.

4.2.1 Aprendizado por reforço

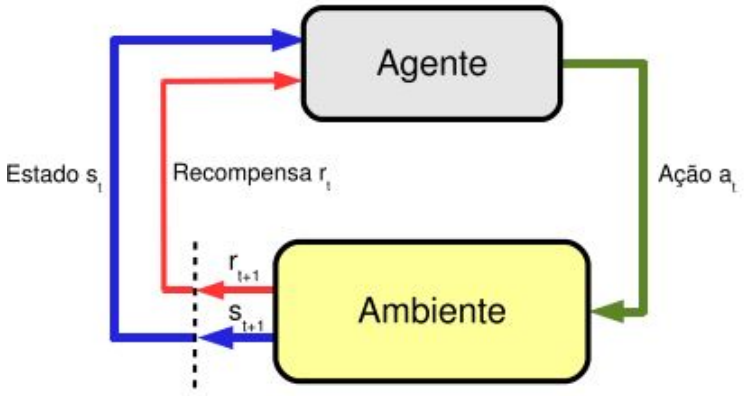
Grande parte do conhecimento adquirido pelos seres vivos, ao longo da vida, origina-se da aprendizagem. O aprendizado através das interações com o ambiente, é a ideia central que está na base da Aprendizagem por Reforço (AR). Um exemplo, é o treinamento de cães para encontrar armas e drogas. Para que isto seja possível, os cães são treinados em simulações para encontrar objetos com odores específicos. Sempre que eles encontram um objeto são recompensados com comida. Com base nesta recompensa, o cão adapta seu aprendizado gradativamente, para encontrar o objeto que retorna a maior recompensa.

Segundo (SUTTON; BARTO, 1998), a Aprendizagem por Reforço resume-se no aprendizado do mapeamento de estados em ações, tendo como objetivo, maximizar o valor de recompensa. O agente aprendiz não é informado quais ações escolher, tendo que descobrir por conta própria, quais ações têm as maiores recompensas através de

tentativas. Para alguns casos, as ações podem não apenas influir na recompensa imediata, mas também, nas recompensas seguintes. O agente aprende de forma autônoma, qual é a melhor política de atuação, se aprimorando através da experiência de suas ações e, dessa forma, interagindo com o ambiente.

A Figura 14, representa o modo como um agente de AR, interage com o ambiente durante processo de aprendizagem. A cada instante t de tempo, o agente percebe o ambiente, e, com base nas informações adquiridas, escolhe e executa uma ação. Essa ação modifica o estado do ambiente, e uma medida dessa modificação de estado, é informada ao agente por meio de um valor de sinal de reforço, r, este processo interativo permite ao agente adquirir, após um determinado número de iterações, conhecimentos para definir qual a melhor ação a ser executada em cada estado (Lima Júnior, 2009).

Figura 14 – Interação entre um agente de aprendizado por reforço e o ambiente.



Fonte: Lima Júnior (2009)

Uma das característica de AR é não necessitar de modelos do ambiente, isto é, sua adaptação é automática a ambientes desconhecidos e dinâmicos. Dessa forma, o aprendizado por reforço, é indicado quando não se conhece a priori o ambiente, ou as informações sobre as situações a serem enfrentadas pelo agente, não estão acessíveis ou são imprecisas. O agente interage diretamente com o ambiente, cujo o objetivo, é executar uma política ótima de ações, que o levem a encontrar seus objetivos. Outro ponto importante dos problemas em aprendizado por reforço, é o dilema chamado de exploração *versus* explotação. O agente deve ser capaz de decidir de forma autônoma, quando irá obter um novo aprendizado ou utilizar o conhecimento adquirido até o momento. A complexidade está na tarefa de examinar uma série de possibilidades: favorecer progressivamente aquelas que fornecem melhores resultados, ou agir de forma, a explorar ações ainda não selecionadas pelo agente (ALMEIDA, 2014)

Quatro elementos básicos de AR, além do ambiente e o agente, definidos por (SUTTON; BARTO, 1998) são:

- Política: uma política especifica o comportamento do agente aprendiz, ou seja, como ele deve escolher suas ações em um dado instante de tempo. Representa parte de um agente AR que possui o objetivo de determinar seu comportamento através das informações fornecido pelo ambiente. Seja S o conjunto de estados do ambiente e A(s) o conjunto de ações disponíveis para um estado $s \in S$. Uma política π é uma função de mapeamento de estados para ações:

$$\alpha = \pi(s), \forall \alpha \in A(s), s \in S, \tag{4.2}$$

  A partir de um estado s, a política informa qual é ação a que o agente deverá tomar.

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

- Função Valor: Esta função realiza a soma de todas as recompensas obtidas pelo agente, a partir do estado atual e dos estados futuros, ou seja, a função valor indica uma estimativa do ganho total que pode ser acumulado pelo agente durante a aprendizagem. Quando a função valor considera apenas o estado s, ela é denominada função valor-estado, e denotada por V(s):

$$V(S) = E\{r_{\mathrm{t+1}} + r_{\mathrm{t+2}} + ... + r_{\mathrm{t+n}} | S_t\} = s, \tag{4.3}$$

  sendo E 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:

$$Q(s, a) = E\{r_{\mathrm{t+1}} + r_{\mathrm{t+2}} + ... + r_{\mathrm{t+n}} | S_t = s, a_t = a\} \tag{4.4}$$

  A diferença da equação 4.4 para 4.3 é o fato de que o ganho esperado depende da ação escolhida no instante t.

- Modelo do ambiente: Tem a função de demonstra o comportamento do ambiente, ou seja, dados o estado e a ação, o modelo pode prever o próximo estado e a próxima recompensa de acordo com a probabilidade de transição.

4.2.2 Algoritmo *Q-learning*

O algoritmo Q-learning é uma técnica desenvolvido por (WATKINS, 1989) é considerado uma das mais importantes contribuições em Aprendizagem por reforço. O

algoritmo *Q-learning* foi classificado como um método de diferença temporal *off-policy*, não necessita de uma modelagem completa do ambiente, pois a sua convergência para ótimos de Q, não depende da política que estão sendo utilizada, ou seja, a função ação-valor Q se aproxima da função ação valor ótima Q*, a partir de atualizações dos pares-ação, que são adquiridas a medida que os pares são visitados (Lima Júnior, 2009). Abaixo, a expressão responsável em atualizar a matriz dos Q-valores no algoritmo *Q-learning*:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{\mathrm{t+1}}, a) - Q(s_t, a_t)] \tag{4.5}$$

onde s_t é o estado atual, a_t é a ação realizada no estado s_t, e r_t é o sinal de reforço (retorno ou recompensa) recebido após executar a_t em s_t, s $_{\mathrm{t+1}}$ é o estado seguinte, γ é o fator de desconto ($0 \leqslant \gamma < 1$) e $\alpha(0 \leqslant \alpha < 1)$ é o coeficiente de aprendizagem. A função Q(s,a) é o valor agregado ao par estado-ação (s,a) e representa quão boa é a escolha dessa ação na otimização do retorno da equação 4.5. Um fator bastante relevante para este algoritmo é a escolha das ações a serem executadas durante o processo de aproximação iterativa da função Q, que poderá ser feita através de qualquer critério de exploração/explotação. Uma das técnicas bastante utilizada, para escolha das ações, é a política de decisão ε-gulosa. Tal política pode ser definida como:

$$a^* = arg \max_a Q(s, a) \tag{4.6}$$

a ação, escolhida na equação acima, corresponde ao maior valor de Q no estado s. Dessa forma, o par estado-ação a ter seu valor atualizado será associado à ação escolhida conforme a política ε-gulosa:

$$\pi(s) = \begin{cases} a^*, \text{com probabilidade 1-}\varepsilon, \\ \text{qualquer outra, com probabilidade } \varepsilon \end{cases} \tag{4.7}$$

O algoritmo *Q-leaning* é considerado o primeiro método de AR a possuir fortes provas de convergências. Em (WATKINS, 1989), cada par estado-ação (s,a) é visitado um número infinito de vezes, então a função Q(s,a) terá sua convergência com probabilidade 1 para Q*, desde que α seja pequeno.

O Algoritmo 1 apresenta o pseudocódigo do *Q-learning* com base em (WATKINS,

1989):

Algoritmo 1: Algoritmo Q-learning

início
  Entrada: $\alpha, \varepsilon, \gamma, NumEpis$
1 Inicializar R(s,a) Matriz de recompensas;
2 Inicializar Q(s,a) Matriz dos Q-valores incialmente zerada;
3 **enquanto** *não atingir NumEsp episódios* **faça**
4 Inicialize s;
5 **enquanto** *não encontrou estado final* **faça**
6 Seleciona a de acordo com a política ε-gulosa;
7 Receber a recompensa r(s,a) e observar o próximo estado s';
8 $Q(s,a) \leftarrow Q(s,a) + \alpha[r_t + \gamma \max_{a \in A} Q(s',a) - Q(s,a)]$;
9 $s \leftarrow s'$;
10 **fim**
11 **fim**
fim

Mais informações sobre o algoritmo Q-learning podem obtidas em (SUTTON; BARTO, 1998) e Barto(1998),(WATKINS, 1989) e (Lima Júnior, 2009).

4.2.3 Detalhe da Implementação do algoritmo *Q-learning*

Nessa seção, iremos mostrar os detalhes relacionados a política de seleção e função de aptidão que foi utilizado pelo algoritmo *Q-learning*.

4.2.3.1 Política de Seleção

Durante o processo de resolução de um problema, utilizando aprendizagem por reforço, é necessário uma política de seleção de ações. Essa política estabelece o comportamento do agente aprendiz, para que o mesmo consiga optar entre o uso de conhecimento já adquirido e a aquisição de conhecimento novo, de forma que possa otimizar o processo de explotação/exploração do espaço de busca. Através de experimento entre ϵ-gulosa e ϵ-gulosa adaptativa, obteve melhor resultados nessa última.

Política $\epsilon-$Gulosa Adaptativa

A política ϵ-gulosa adaptativa é bem semelhante a política $\epsilon-$gulosa. A ϵ-gulosa utiliza a expressão (1-ϵ) para definir a probabilidade durante toda a execução para definir a ação aleatória, com probabilidade e. Já a política ϵ-gulosa adaptativa tem modificações

do valor de ϵ sofrendo decaimento exponencial calculado por

$$\epsilon = max\{v_i, v_f * b^k\} \tag{4.8}$$

tendo k como o contador de episódios do *Q-learning*, b é um valor próximo de 1 e $v_i < v_f \in [0, 1]$. Com isso, inicialmente o algoritmo utilizará valores grandes para ϵ e a medida que o valor de k cresce, a escolha de ϵ é convergida para valores menores (mais próxima a v_i). Dessa forma, o algoritmo realiza escolhas mais aleatórias, nas primeiras iterações, e a medida que o número de episódios aumentam, o aspecto guloso torna-se mais utilizado (Lima Júnior, 2009).

O algoritmo 2, mostra os passos da metaheurística híbrida que tem a população inicial do BRKGA gerada pelo algoritmo *Q-learning*. Ao se examinar o algoritmo 2, percebe-se que o BRKGA teve modificações na linha 2 e 3. Na linha 2 tem a execução do *Q-learning* que retorna a matriz Q (matriz dos q-valores) após executar uma quantidade de episódios (NEp). Na linha 3, o método *gerar_populacao_inicial* utiliza a matriz Q e inicia-se a geração de cromossomos. Para realizar a criação de cada cromossomo, será utilizado Q(s,a) associado a cada gene, de forma que, os genes que têm maiores valores de Q(s,a) serão escolhidos prioritariamente na composição do cromossomo até formar uma solução.

Algoritmo 2: *Brkga_com_Q_learning*$(\mid P \mid, \mid P_e \mid, \mid P_m \mid, n, \rho_a)$

Entrada: Matriz de Demandas entre as localidades
Saída: Melhor Solução Encontrada

```
1  início
2      Q ← QLearning(R, α, ε, ω, NEp);
3      pop_atual ← gerar_população_inicial(Q, | P |);
4      enquanto condições de paradas não satisfeitos faça
5          nova_pop ← ∅;
6          avaliar_populacao(pop_atual);
7          Particione pop_atual em dois conjuntos: P_elite e P_naoElite;
8          nova_pop ← P_elite;
9          Gere o conjunto de mutantes P_m ;
10         nova_pop ← nova_pop ∪ P_m;
11         Copiar as soluções recém-modificadas em nova_pop;
12         enquanto nova_pop não estiver completa faça
13             c ← cruzarPaiElite_Nao_Elite(P_elite P_naoElite ) ;
14             nova_pop ← nova_pop ∪ c;
15         fim
16         avaliar_populacao(nova_pop);
17         pop_atual ← nova_pop;
18     fim
19 fim
```

4.3 Algoritmo BRKGA com *Vocabulary Building* Aplicado ao PALAS

Nesta seção, propõe-se apresentar os conceitos e implementação do *Vocabulary Build*, além de apresentar a agregação do BRKGA com VB, afim de obter melhores resultados. Outro ponto que foi abortado nessa seção, foi a junção do VB com o *Q-learning*, para que o algoritmo de aprendizado por reforço possa disponibilizar bons vocábulos e consequentemente bons resultados.

4.3.1 Vocabulary Building

A técnica *Vocabulary Building* (VB) foi desenvolvida por Fred Glover, dentro do contexto do *Path Relinking* (GLOVER, 1992). No ano de 1993, essa técnica foi implementa dentro da metaheurística Busca Tabu na obra de (GLOVER; LAGUNA, 1993).

Outras publicações de destaque que utilizam o VB: (GLOVER, 1997); (GLOVER, 1999); (GLOVER; LAGUNA; MARTÍ, 2000); (GLOVER; LAGUNA; MARTI, 2003). Trabalho mais recentes de VB: Em 2006, essa técnica foi utilizada no algoritmo memético para problema do caixeiro viajante (GUEDES; ALOISE, 2006); em (SOARES, 2009), que utilizou VB, tendo uma solução híbrida junto com algoritmo Busca Tabu para o Problema de Atribuição de Localidades em Anéis SONET; em (NASCIMENTO; ALOISE, 2014) aplicou VB no Algoritmo Memético, para o solucionar o Problema de Roteamento de Unidades Móveis de Pistoneio; e em (FERREIRA, 2011) foi aplicado VB junto Algoritmo Memético para problema Roteamento de veículos com Frotas heterogêneas.

O funcionamento do VB consiste, basicamente, em identificar boas soluções parciais (trechos de soluções completa) no conjunto dos indivíduos elite, onde é considerada uma boa solução parcial, quando um trecho está presente vários indivíduos da soluções-elite. Então, o VB opera, criando um conjunto de fragmentos de soluções chamados de vocábulos, que são montados, desmontados e modificados para produzir novos fragmentos.

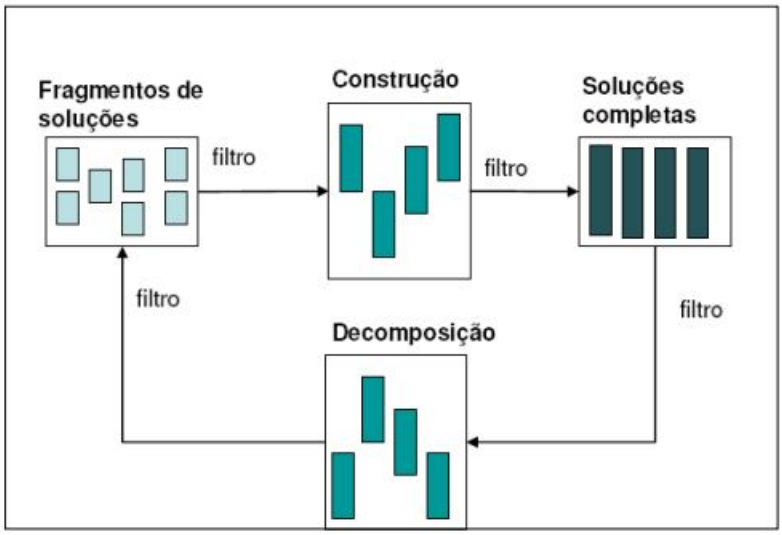
A principal motivação encontrada no VB, é tirar vantagem desses contextos, onde certas configurações parciais de soluções, frequentemente ocorrem como componentes de boas soluções completas. Esta estratégia de busca de "boas configurações parciais" podem ajudar a evitar a explosão combinatória, resultante da manipulação de apenas elementos primitivos. O processo evita a necessidade de reinventar a estrutura de uma configuração parcial como uma base para construir uma boa solução completa (GLOVER, 1999).

Um conjunto de vocábulos recebe o nome de "Pool de vocábulos", que funciona como uma espécie de memória adaptativa. Pois a cada momento, soluções parciais distintas podem ser de diferentes qualidades. Dessa forma, é interessante que o Pool mantenha somente as que forem mais promissoras dentro do processo de busca(GUEDES; ALOISE,

2006).

A figura 15 ilustra o mecanismo utilizado no processo para construção do VB. Nesse processo, são utilizados os métodos construtivos (constrói as novas soluções) e destrutivos (descompõe as soluções, modificando para fragmentos), outro instrumento é o filtro que fica responsável em escolher os melhores fragmentos do pool e as melhores soluções das populações.

Figura 15 – Processo da construção do vocabulário

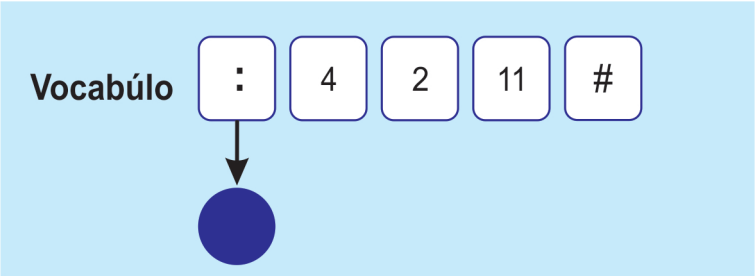


Fonte: Leite (2006)

Essa estratégia é altamente aplicável a uma variedade de procedimentos metaheurísticos, tanto para os procedimentos com memória adaptativa, como para modelos evolucionários e populacionais.

Neste trabalho, será utilizado uma proposta da *Vocabulary Build* que foi apresenta por (GUEDES; ALOISE, 2006). A ideia da proposta, é trabalhar com um pool de vocábulo (memória), que será usado e modificado durante processo de busca, além de um conjunto de solução completa. A figura 16 apresenta uma representação de um vocábulo do PALAS.

Figura 16 – Representação de um Vocábulo



Fonte: Autor

Na representação do vocábulo, é exibido um trecho de solução contendo as localidades {4, 2, 11 } que indicam uma sequência de localidades, no qual, serão utilizadas para tentar melhorar alguma solução completa. No campo que contém "#" será utilizada para armazenar a *fitness* do vocábulo, ou seja, é utilizado para indicar a qualidade do trecho. A partir desse valor, são tomadas decisões sobre o que fazer com o vocábulo, por exemplo, alterar ou apagar. Enquanto o campo que apresenta o ":" é utilizado para indicar como foi realizada a construção do vocábulo; o círculo apontado por esse campo foi denominado de "árvore de quebra". Essa estrutura será melhor entendida a partir da explicação do funcionamento da técnica do VB.

De forma resumida, o processo é iniciado com a geração de vocábulos que são adicionados no *pool* de vocábulo, gerados aleatoriamente ou utilizando *Q-learning*, e que podem ser combinados, a fim obter trechos mais úteis. Em seguida, passa-se a realizar aplicação do *pool* nas soluções elites e vai verificando quais trechos produzem melhorias e quais não produzem. Os vocábulos serão constantemente modificados, pois a intenção, é manter apenas os de alta qualidade e descartar/alterar os de baixas qualidades.

4.3.2 Aplicação do Pool de Vocábulos

Após possuir um *pool* de vocábulos preenchidos, o algoritmo aplica cada vocábulo do *pool* sobre uma solução completa do PALAS, afim de melhorar a solução. O algoritmo 3, realiza a inserção de cada vocábulo do *pool*, separadamente, na solução fornecida como entrada. No final da execução, retorna-se a melhor solução encontrada como resultado de uma inserção. Essa função, mantém, a inserção do vocábulo na solução, mesmo que essa não tenha melhorado sua *fitness*.

Algoritmo 3: Aplicar Pool em solução (P,S)

Entrada: Pool de vocábulos P, Solução completa S
Saída: Melhor solução encontrada, das inserções realizadas

```
1  início
2      melhor.fitness ← ∞;
3      para cada vocábulo V em P faça
4          auxiliar ← S;
5          atual ← inserir_vocabulo_solucao(V, auxiliar, P);
6          se atual.fitness < melhor.fitness então
7              melhor ← atual;
8          fim
9      fim
10 fim
```

O algoritmo 4 descreve os passos para inserção de um vocábulo em um solução. Este algoritmo, inicia realizando a transcrição do trecho, ou seja, vai retirar as localidades de S que estão contidas em V, para evitar repetições, com isso teremos uma solução

incompleta S'. Em seguida, é verificado qual a melhor posição para a inserção do vocábulo em S'. Depois de adicionar o trecho na solução S', retorna-se a nova solução completa obtida. Esses passos são ilustrados na figura 17.

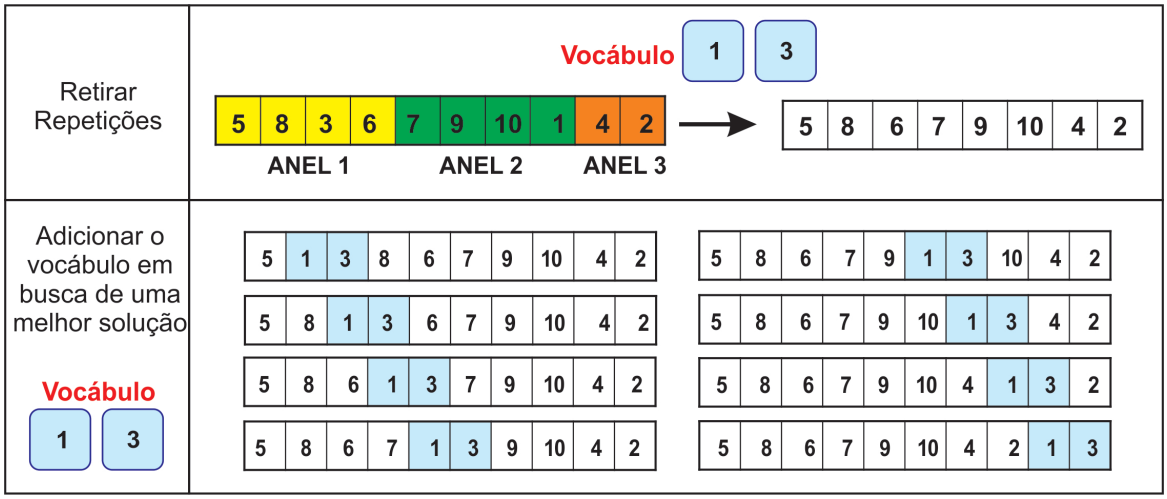
Algoritmo 4: *inserir_vocabulo_em_solucao*(V, S, P)

Entrada: Vocábulo V, Solução completa S, Pool de vocábulos P
Saída: Solução completa após V ser inserido em S

```
1  início
2      res ← transcrever(V, S);
3      melhor.fitness ← ∞;
4      se res.fitness < S.fitness então
5          V.contador ← V.contador + 1;
6      senão
7          V.contador ← V.contador − 1;
8      fim
9      se V.contador ⩽ 0 então
10         se V.arvore_quebra == Nulo então
11             se rand() < PROB então
12                 alterar_vocabulo(V, S);
13                 V.contador ← CONT_INI;
14             senão
15                 V ← novo_vocabulo_aleatorio();
16                 V.contador ← 0;
17             fim
18         senão
19             desnaturar_vocabulo(V, X, Y);
20             X.contador ← CONT_MAX;
21             Y.contador ← CONT_MAX;
22             P ← P − {V};
23             P ← P ∪ {X} ∪ {Y};
24         fim
25     fim
26     se V.contador ≥ CONT_MAX então
27         combinar_vocabulo(V, P);
28         V.contador ← CONT_INI;
29     fim
30 fim
```

Realizada a transcrição, voltando para o algoritmo 4, verifica-se que, a operação causou uma melhoria na solução que foi fornecida como entrada. Caso positivo, incrementa-se o contador do vocábulo; caso negativo, seu contador é decrementado. Depois do incremento/decremento do contador, é necessário verificar a qualidade do vocábulo: Se o contador do vocábulo chegou a um valor muito alto, significa que esse vocábulo tem alta qualidade, então, realiza-se a combinação desse vocábulo com algum outro do *pool*; caso contrário, significa que esse vocábulo não vem melhorando as soluções, todavia,

Figura 17 – inserção do vocábulo {1, 3} na solução completa {5,8,6,7,9,10,4,2}

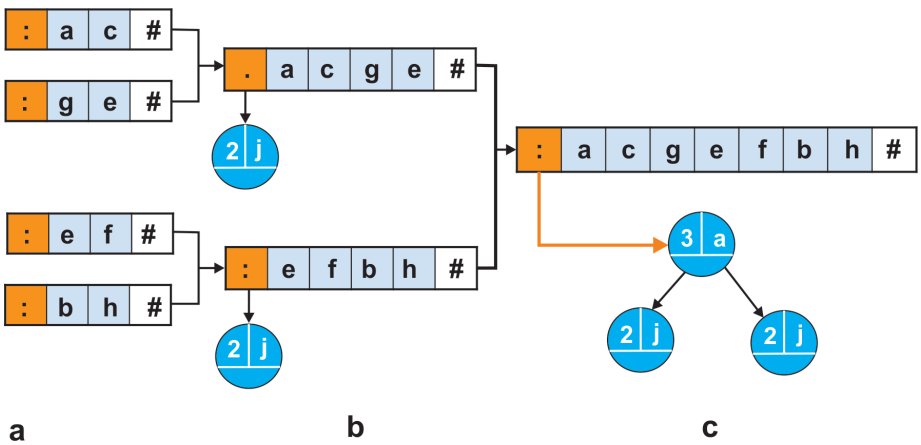


Fonte: Autor

é necessário saber se o vocábulo é composto (originado por dois outros vocábulos) ou elementar (apenas um fragmento). Se for um vocábulo composto, ele é retirado do *pool* e quebrado em dois. As partes resultantes da quebra são realocadas ao *pool*, com seus contadores reinicializados no valor máximo. Para o caso de o vocábulo ser elementar, gera-se um número aleatório e, dependendo de uma probabilidade PROB, pode-se realizar a alteração ou apaga o vocábulo.

Quando um vocábulo X atinge um alto valor em seu contador, é iniciado um processo de combinação, isso indica que esse vocábulo tem melhorado as soluções em que foram inseridas. Então, busca-se um vocábulo Y, que seja compatível e que tenha o maior contador possível. Para dois vocábulos serem considerados compatíveis, é necessário não possuírem nenhuma localidade em comum. Na figura 18, demonstra o funcionamento do processo de combinação, exibindo a construção da árvore de quebra de um vocábulo.

Figura 18 – Combinação de vocábulos



Fonte: Adaptado de (GUEDES; ALOISE, 2006)

Inicialmente em “a”, é exibido um conjunto de vocábulo contendo 4 vocábulos independentes. De “a” para “b”, foi realizado uma combinação dos dois vocábulo de cima e dos dois outros vocábulo de baixo. É necessário guardar informações para realizar as separações desses vocábulo: o número 2 (terceira posição) indica a posição que inicia o segundo vocábulo; enquanto a letra j (de justaposição) indica que os vocábulos originais não possuíam localidades em comum; Na combinação realizada entre 18.b para 18.c existe a localidade “E” se repetindo nos dois vocábulos, como eles possuem uma localidade da extremidade em comum, esse fato está indicado pela letra a (de Aglutinação) da árvore de quebra do vocábulo resultante.

Como foi visto, o numeral do nó-raiz indica a posição, a partir da qual, o vetor armazena o segundo vocábulo da combinação. Com a utilização da árvore de quebra, é possível realizar o processo inverso que foi denominado de desnaturação, no procedimento inserir vocábulo em solução.

4.3.3 Geração de vocábulos

Como foi visto na seção 4.3, o *Vocabulary Build* realiza no início de sua execução, a geração de vocábulos para preenchimento do *pool*, em seguida, esses vocábulos são avaliados e modificados. Na literatura existem várias formas de gerar a população inicial do *pool* do VB, porém, as mais encontradas são: i) A geração de vocábulo de forma aleatória que foi utilizada em Guedes e Aloise (GUEDES; ALOISE, 2006); ii) Formação de vocábulo a partir de solução já encontradas, ou seja, é uma formação de vocábulos através de busca de fragmentos de boa qualidade em soluções disponíveis, por exemplo, encontrar bons fragmentos na população elite do algoritmo genético, esse exemplo foi utilizado por Nascimento e Aloise em (NASCIMENTO; ALOISE, 2014).

Então, tentando aprimorar esse processo e encontrar novas fórmulas para geração de vocábulos, esse trabalho, propõe a utilização de aprendizado por reforço para a geração dos vocábulos utilizados pelo VB. A ideia é utilizar do algoritmo *Q-learning* para geração de vocábulos com boas qualidades, tentando o melhor desempenho do VB, junto com os algoritmos proposto por esse trabalho.

No algoritmo 5 é descrita uma versão do BRKGA hibridizado com *Vocabulary Build*. Na linha 3, consta a função responsável pela geração dos vocábulos, que podem ocorrer de forma aleatória ou com auxílio do *Q-learning*. Na linha 12, destaca-se a aplicação do VB utilizando o *pool* de vocábulos criados anteriormente, e na linha 13, é aplicação de buscais locais para melhoramento das soluções. As buscas locais estão mais detalhadas na seção seguintes (4.3.4).

Algoritmo 5: *Brkga_com_vocabulary_build*($| P |, | P_e |, | P_m |, n, \rho_a$)

Entrada: Matriz de Demandas entre as localidades
Saída: Melhor Solução Encontrada

```
1  início
2      pop_atual ← gerar_população_inicial();
3      pool ← gerar_pool_inicial();
4      enquanto condições de paradas não satisfeitos faça
5          nova_pop ← ∅;
6          avaliar_populacao(pop_atual);
7          Particione pop_atual em dois conjuntos: P_elite e P_naoElite;
8          nova_pop ← P_elite;
9          Gere o conjunto de mutantes P_m ;
10         nova_pop ← nova_pop ∪ P_m;
11         para cada solução S do conjunto elite faça
12             aplicar_pool_em_solucao(pool, S);
13             aplicar_Buscar_Local(S)
14         fim
15         Copiar as soluções recém-modificadas em nova_pop;
16         enquanto nova_pop não estiver completa faça
17             c ← cruzarPaiElite_Nao_Elite(P_elite P_naoElite ) ;
18             nova_pop ← nova_pop ∪ c;
19         fim
20         avaliar_populacao(nova_pop);
21         pop_atual ← nova_pop;
22     fim
23 fim
```

4.3.4 Buscas Locais

A função principal da utilização da busca local junto com VB, é explorar a vizinhança das soluções obtidas e caminhar em busca do ótimo local (para cada solução), antes de continuar a execução do BRKGA. Tendo uma solução S, uma vizinhança de busca N, é considerada um subconjunto de soluções N(S). Logo, a solução S é denominada localmente ótima, se não houver nenhuma solução em N(S) melhor que S.

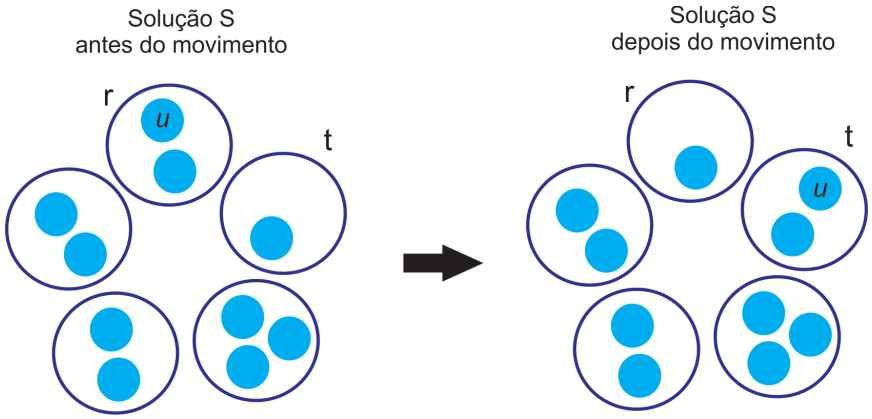
Afim de melhorar a eficiência da metaheurística, foram testadas quatro vizinhanças, presentes em Bastos(BASTOS; OCHI; MACAMBIRA, 2005b), que são a N_1, N_2, N_3 e N_4. Através dos resultados experimentais, percebeu-se que as vizinhanças N_1, N_2 e N_3 foram as que apresentaram melhores resultados quando aplicadas nos algoritmos propostos.

4.3.4.1 Vizinhança N_1

A vizinhança N_1 tem como objetivo, procurar um vértice u pertencente a um anel r e tentar realocá-lo para outro anel t, de modo que, a demanda no anel federal seja reduzida, sem que nenhum anel local seja inviabilizado, ou seja, sejam r e t dois anéis da

solução S, uma localidade $u \in r$ é movida para o anel t caso $demT_{t \cup \{u\}}$ continue menor ou igual a B e $demAF$ (demanda do anel federal) seja reduzida. A Figura 19 mostra o movimento associado à vizinhança N_1.

Figura 19 – Ilustração do movimento de vizinhança N_1

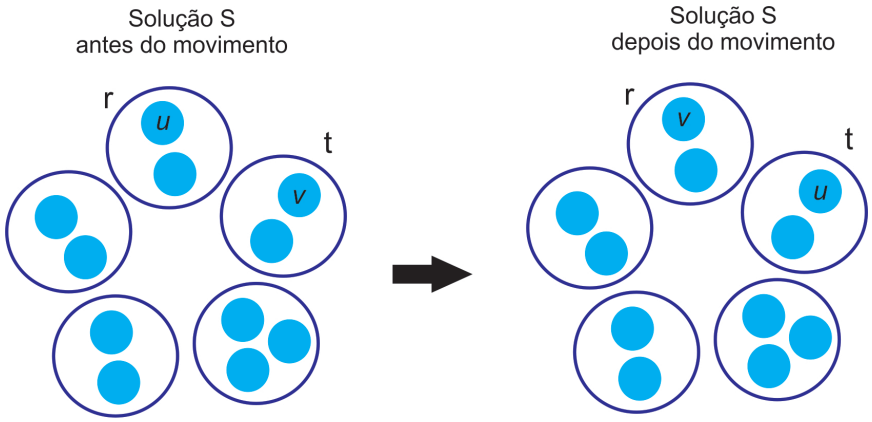


Fonte: Adaptado de Silva (2008)

4.3.4.2 Vizinhança N_2

A vizinhança N_2 tem como objetivo, procurar duas localidades u e v pertencentes a anéis distintos r e t, para efetuar a troca de anéis. Tendo $u \in r$ e $v \in t$, após a troca, teremos $u \in t$ e $v \in r$. Da mesma forma, como na vizinhança N_1, a troca da vizinha N_2 só será realizada caso os anéis locais permaneçam viáveis e a demanda do anel federal seja reduzido. A Figura 20 mostra o movimento associado à vizinhança N_2.

Figura 20 – Ilustração do movimento de vizinhança N_2

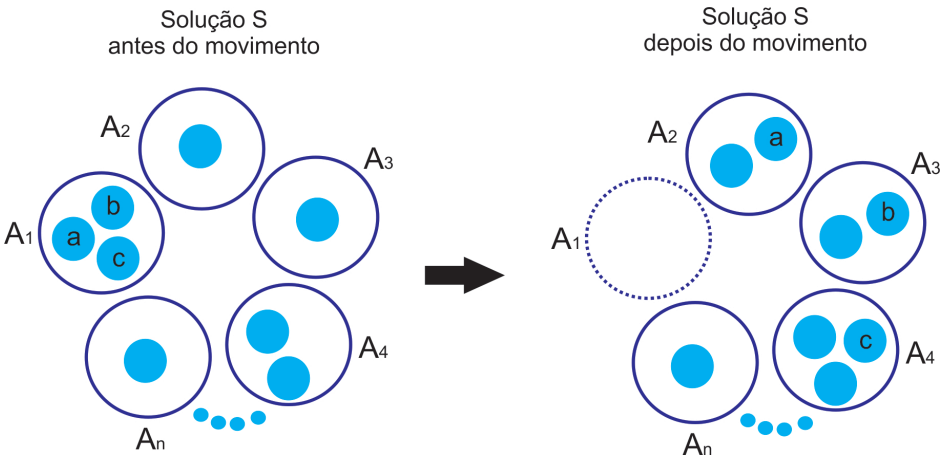


Fonte: Adaptado de Silva (2008)

4.3.4.3 Vizinhança N_3

A vizinhança N_3 tem como propósito, o esvaziamento do anel, o de menor demanda total em S. Essa vizinhança tenta redistribuir as localidades de um anel t, entre os demais

Figura 21 – Ilustração do movimento de vizinhança N_3



Fonte: Adaptado de Silva (2008)

anéis da solução. Muitas soluções que são geradas, têm anéis que apresentam um ou mais anéis com pouco tráfego, que podem ser esvaziados sem inviabilizar outros anéis. Essa vizinhança, procura o anel t mais promissor para receber o vértice $u \in r$, de modo que, t continue sendo um anel viável. Essa operação se repete, até que o anel r seja esvaziado ou até que nenhuma outra retirada de r seja possível. É importante destacar, que esse procedimento de busca associado à V_3, não inviabiliza soluções viáveis, como também, não garante que o anel de menor demanda poderá ser esvaziado através da busca local associada à N_3. A Figura 21 ilustra o movimento associado à vizinhança N_3.

5 Resultados Computacionais

Este capítulo, tem como objetivo mostrar os resultados computacionais obtidos na realização da pesquisa utilizando os algoritmos mencionados no capítulo 4. Além disso, será feito uma comparação com alguns trabalhos disponíveis na literatura.

5.1 Instâncias do problema

Os experimentos utilizando os algoritmos, foram executados em quatro classes de instâncias. As duas primeiras classes (C1 e C2) foram definidas originalmente em dois trabalhos: (GOLDSCHMIDT; LAUGIER; OLINICK, 2003) e (ARINGHIERI; DELL'AMICO, 2001) . O grupo de instâncias C3, conhecido por LSHK foi introduzido por (LEE et al., 2000), esse conjunto tem apenas duas instâncias viáveis para o problema do PALAS. Enquanto a quarta classe de instâncias (C4) foi definida por Bastos em (BASTOS, 2005)

5.1.1 Classes C1 e C2

A classe C1 possui 118 instâncias viáveis e a classe C2 possui 230 instâncias viáveis. Essas duas classes, são formadas por dois grupos de instâncias: geométricas e aleatórias. As geométricas têm as localidades (clientes) mais próximas se comunicando com mais frequência do que as localidades mais distantes. Já as instâncias do grupo das aleatórias, foram produzidas a partir de grafos completos com retiradas aleatórias de arestas de acordo com a probabilidade p $\in$ (0,1). As instâncias possuem dois tipos de capacidades diferentes para os anéis, ou seja, existem instâncias com a capacidade de B=155 MB/s e instâncias com B=622 MB/s. Além disso, existem instâncias com os seguintes valores para número de vértices: 15, 25, 30 ou 50. A tabela 3 descreve a nomenclatura e a quantidade de cada grupo de instâncias. Em Macambira (MACAMBIRA, 2003), encontra-se a comprovação da otimalidade e os valores das soluções ótimas dessas instâncias.

5.1.2 Classe C3

O grupo de instâncias C3, mais conhecido como instâncias LSHK, foi apresentado por (LEE et al., 2000). Esse grupo possui 40 instâncias, com capacidade de tráfego do anel federal limitado em 48 Mbps, porém, apenas 2 instâncias são consideradas viáveis para o problema do PALAS. A tabela 4, lista as instâncias desta classe.

Tabela 3 – Capacidade e Nomenclatura das instâncias das classes C1 e C2

Classe	Grupo	Capacidade (B)	Nome	Qtd. instâncias
C1	Geométricas	155 MB/s	GS (Geometric Short)	23
	Geométricas	622 MB/s	GL (Geometric Large)	27
	Aleatórias	155 MB/s	RS (Random Short)	31
	Aleatórias	622 MB/s	RL (Random Large)	37
			Total	118
C2	Geométricas	155 MB/s	new.GL (Geometric Low)	70
	Geométricas	622 MB/s	new.GH (Geometric High)	70
	Aleatórias	155 MB/s	new.RL (Random Low)	70
	Aleatórias	622 MB/s	new.RH (Random High)	20
			Total	230

Tabela 4 – Instâncias da Classe C3

Nome da instância	Capacidade (B)	Número de Localidades
LSHK_25_35_02	48 MB/s	25
LSHk_25_35_08	48 MB/s	25

5.1.3 Classe C4

A classe C4 foi criada utilizando dois algoritmos para geração das instâncias, sendo eles: o algoritmo RTD (*Random by Traffic Density*), que consiste em gerar um grafo aleatório dado um percentual de densidade de tráfego D_t desejado; o outro algoritmo é o RWNR (*Random with Target Number of Rings*), que tem como propósito, a geração de instâncias viáveis e fornecimento de uma solução viável para essas instância. Através desses algoritmos foram geradas 105 instâncias com soluções conhecidas. Os valores de B utilizados na geração dessas instâncias, variaram entre 51, 84 MB/s (STS- 1) e 39813,12 MB/s (STS-768). A tabela 5 descreve a nomenclatura e as capacidades das instâncias desta classe. As instâncias C4 são maiores e mais densas do que aquelas das classes C1, C2 e C3; tornando-se a classe mais difícil desse problema. Para a classe C4 não se conhece as soluções ótimas, dessa forma, é necessário utilizar o limite inferior para avaliar as soluções. (SOARES, 2009)

Tabela 5 – Capacidade e Nomenclatura das instâncias da classe C4

Grupo	Número da instância	Capacidade (B)	Qtd. Instâncias
RWNR_100	[01...10]	2488 MB/s	46
	[12 ... 16]	9953,28 MB/s	
	[11 e 17 ... 26]	51,84 MB/s	
	[27 ... 31]	155,52 MB/s	
	[32 ... 36]	622,08 MB/s	
	[37 ... 41]	2488,32 MB/s	
	[42 ... 46]	39813,32 MB/s	
RWNR_150	[01 ... 10]	2488 MB/s	10
RWNR_200	[01 ...10]	9953 MB/s	31
	[11 ...15]	51,84 MB/s	
	[16 ... 21]	155,52 MB/s	
	[22 ... 26]	2488,32 MB/s	
	[27 ... 31]	39813,32 MB/s	
RWNR_250	[01 e 02]	622 MB/s	6
	[03, 05 e 06]	9953,28 MB/s	
	4	622,08 MB/s	
RWNR_300	[01 ... 06]	622,08 MB/s	6
RWNR_400	[01 ... 06]	2488,32 MB/s	6
		Total	105

5.2 Resultados e Análises do Experimentos

Nesta seção, apresentamos os resultados dos algoritmos propostos neste trabalho. Inicialmente, será descrito o ambiente de desenvolvimento (Seção 5.2.2), no qual os algoritmos foram desenvolvidos. Em seguida, serão apresentados os resultados para as classes C1, C2, C3, C4 e, por fim, comparação dos resultados com outras heurísticas da literatura.

5.2.1 Ambiente de Desenvolvimento

Os algoritmos deste trabalho, foram implementados na linguagem Java, utilizando a IDE Eclipse Neon 2017. Os testes foram executados sobre as quatro classes de instâncias C1, C2, C3 e C4 em computador do tipo PC com o sistema operacional Windows 10 64 bits, com 8 Gb de memória RAM com frequência de 1333 Mhz e processador Intel Core i7 de 2.5 Ghz com 2 núcleos.

5.2.2 Experimentos

Cada uma das instâncias, das classes C1, C2, C3 e C4, foi executada dez vezes, e os parâmetros utilizados foram os mesmos para os quatros algoritmos: BRKGA (BR), BRKGA com *Q-Learning* (BR + QL), BRKGA com *Vocabulary Build* (BR + VB) e BRKGA com *Vocabulary Build* auxiliado pelo *Q-Learning* (BR+VB(QL)). Os parâmetros utilizados pelo BRKGA estão descrito na tabela 2 . O critério de parada aplicado, foi encontrar uma solução viável com o número de anéis igual ao da solução ótima de cada instância. Caso não fosse encontrada, o algoritmo era parado após 150 iterações.

Da tabela 6 até a tabela 8 são mostrados os resultados de forma resumida, os resultados mais detalhados encontram-se no apêndice A. Cada tabela, mostra o percentual geral das instâncias, em que a solução ótima foi encontrada pelo menos uma vez *(%), a convergência percentual média C(%) e o tempo computacional médio em (minutos:segundos:milissegundos), para encontrar uma solução correspondente ao alvo (solução ótima) ou até completar o máximo de iterações propostas.

5.2.2.1 Resultados das Classes C1

A tabela 6, contém os resultados da classe C1 para todos os algoritmos utilizado nesse trabalho e podemos verificar que os algoritmos BR+VB e BR+VB(QL) chegaram às soluções ótimas de todas as 118 instância disponíveis. O BR obteve as soluções ótimas de 109 das 118 (92,37 %) e BR+QL obteve as soluções ótimas de 114 das 118 (96,61%). BR e BR+QL foram os algoritmos que não utilizaram busca locais. Comparando os resultados, já é possível notar uma pequena vantagem do BR+QL em relação ao BR.

Tabela 6 – Quadro geral dos resultados da classe C1

Instâncias		BR			BR + QL			BR + VB			BR + VB(QL)		
Tipo	#IV	*(%)	C(%)	T. Médio	*(%)	C(%)	T. Médio	*(%)	C(%)	T. Médio	*(%)	C(%)	T. Médio
GS	23	100	94,35	(0:0:181)	100	95,22	(0:11:794)	100	97,00	(0:1:234)	100	100	(0:7:713)
GL	27	100	88,50	(0:0:816)	100	93,30	(0:7:203)	100	99,60	(0:2:259)	100	99,63	(0:6:99)
RS	31	83,87	74,52	(0:0:564)	100,00	72,58	(0:6:180)	100	95,81	(0:11:504)	100	97,42	(0:20:424)
RL	37	89,19	76,49	(0:0:611)	89,19	82,97	(0:6:185)	100	91,62	(0:15:887)	100	96,76	(0:22:842)
TOTAL	**118**	**92,37**	**82,20**	**(0:0:547)**	**96,61**	**83,64**	**(0:7:508)**	**100**	**98,89**	**(0:7:387)**	**100**	**94,83**	**(0:14:669)**

Analisando os resultados da classe C1, observamos que o BR+VB e BR+VB(QL) encontraram os melhores resultados dentre os algoritmos, porém, em relação à taxa de convergência, o BR+VB teve uma pequena vantagem em relação ao BR+VB(QL). Além disso, o BR+VB obteve a menor média de tempo computacional.

5.2.2.2 Resultados das Classes C2

Verificamos, na Tabela 7, que o BR+VB(QL) foi o único algoritmo que chegou à solução ótima de todas as 230 instâncias da classe C2. O BR obteve o pior resultado,

pois, obteve as soluções ótimas de 108 das 230 (46,96%). O BR+QL obteve as soluções ótimas de 216 das 230 (93,71 %) e o BR+VB obteve as soluções ótimas de 225 das 230 (97,83%). Ainda pela tabela, Verifica-se também, que o BR+VB(QL) encontrou os melhores resultados dentre os algoritmos, porém, teve a maior média de tempo computacional.

Tabela 7 – Quadro geral dos resultados da classe C2

Instâncias		BR			BR + QL			BR + VB			BR + VB(QL)		
Tipo	#IV	*(%)	C(%)	T. Médio	*(%)	C(%)	T. Médio	*(%)	C(%)	T. Médio	*(%)	C(%)	T. Médio
GL	70	70	39,86	(0:1:280)	100	85,57	(0:36:444)	95,71	82,71	(0:18:937)	100	89	(0:57:417)
GH	70	34	13,43	(0:2:521)	99	77,57	(0:38:16)	100	73,57	(0:25:622)	100	88,57	(0:53:398)
RL	70	51,43	28,43	(0:1:296)	81,43	64,43	(0:30:950)	97,14	61,71	(0:25:622)	100	81,29	(0:50:772)
RH	20	65,00	40,00	(0:0:714)	100	68,00	(0:3:508)	100	71,00	(0:25:558)	100	80,00	(0:13:269)
TOTAL	**230**	**46,96**	**27,17**	**(0:1:614)**	**93,91**	**75,17**	**(0:32:386)**	**R$97,83**	**72,52**	**(0:21:625)**	**100**	**85,83**	**(0:50:333)**

Na classe C2 já é possível notar a qualidade que foi obtida pela metaheurística híbrida, pois BR+QL conseguiu encontrar as soluções ótimas para 93,9% das instâncias, enquanto o BR, só encontrou para 46.9% das instâncias. Porém, o BR+QL possui uma maior média de tempo computacional, ou seja, é mais lento, devido o treinamento que o algoritmo *Q-learning* necessita durante seu processamento. Esse aumento de tempo computacional, também pode ser verificado no BR+VB(QL), que teve o melhor desempenho, porém, foi o que gastou mais tempo para executar todas as instâncias.

5.2.2.3 Resultados da Classe C3

Como já foi mencionado, a classe C3 possui apenas 2 instâncias viáveis que são consideradas difíceis, pois alguns trabalhos encontrados na literatura, não conseguiram obter as soluções ótimas. A tabela 8, contém os resultados detalhados para classe C3, verifica-se que o BR foi o algoritmo com o pior desempenho, pois só conseguiu encontrar a solução ótima para a instância LSHK_25_35_08. Os demais algoritmos, conseguiram encontrar bons resultados. O BR+VB(QL) e BR+VB tiveram desempenho parecido, mas o primeiro, teve a melhor convergência média (70%), por outro lado, teve o maior tempo médio computacional.

Tabela 8 – Quadro geral dos resultados da classe C3

Instâncias		BR			BR+QL			BR+VB			BR+VB(QL)		
Nome	*	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)
LSHK_25_35_02	6	(0:0:529)	-	0	(0:5:183)	6	50	(0:9:44)	6	20	(0:17:30)	6	40
LSHK_25_35_08	6	(0:0:451)	6	40	(0:4:802)	6	80	(0:0:218)	6	100	(0:0:856)	6	100
Médias		**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**
		(0:0:490)	**50**	**20**	**(0:4:993)**	**100**	**65**	**(0:4:631)**	**100**	**60**	**(0:8:943)**	**100**	**70**

5.2.2.4 Resultados da Classe C4

Para esta classe foi realizado os mesmos experimentos descritos nas seções anteriores, porém, devido à complexidade da instâncias C4, como mencionado na seção 5.1.3, foi necessário realizar mudanças nas condições de parada: a primeira mudança, foi que nos experimentos para esta classe foi fixado o número máximo de 50 iterações, isso por causa do elevado tempo computacional demandado para sua execução; a segunda mudança, como o resultado ótimo para as instâncias C4 não é conhecido, a outra estratégia para parada do algoritmo, além da quantidade iterações, foi se o algoritmo encontrasse um resultado igual ao limite inferior (k_{lb}) subtraindo de 1. Para realizar o cálculo do k_{lb}, utilizamos a seguinte fórmula 5.1 (GOLDSCHMIDT; LAUGIER; OLINICK, 2003):

$$k_{lb} = \frac{\sum_{(u,v)\in E|u<v} d_{uv}}{B} \tag{5.1}$$

A partir da fórmula 5.1 utilizada, vemos que o tráfego total entre as localidades (u,v), é somado e então dividido pela capacidade B da rede. É necessário subtrair a quantidade 1 (um) do k_{lb}, pois, pela fórmula acima, não se leva em consideração que parte do tráfego está passando pelo anel federal e, com isso, não conta como anel local. O PALAS utiliza o número de anéis locais para analisar os resultados dos algoritmos. Então, pode ser que a fórmula encontre um valor x, mas o verdadeiro limitante inferior deve ser x - 1, pois parte do tráfego, está no anel federal, e por isso, não se criou um anel local. Por exemplo, se existe um tráfego total entre as localidades igual a 1000 e uma capacidade B igual a 100, tem-se um k_{lb} igual a 10. Entretanto, imagine-se que o tráfego no anel federal seja de 100, dessa maneira, um dos anéis é o anel federal, que não contém localidades, e o limitante inferior correto seriam 9 anéis locais e o anel federal (SOARES, 2009).

A tabela 9, apresenta os resultados obtidos para classe de instância C4. É possível ver que para cada algoritmo, temos o percentual médio de soluções viáveis que foram encontradas SV(%), em seguida, a convergência percentual média C(%) e o tempo computacional médio em (minutos:segundos:milissegundos). A última linha da tabela, descreve a média dos resultados para o SV(%), C(%) e o tempo médio para cada algoritmo.

Tabela 9 – Resumo dos resultados da classe C4

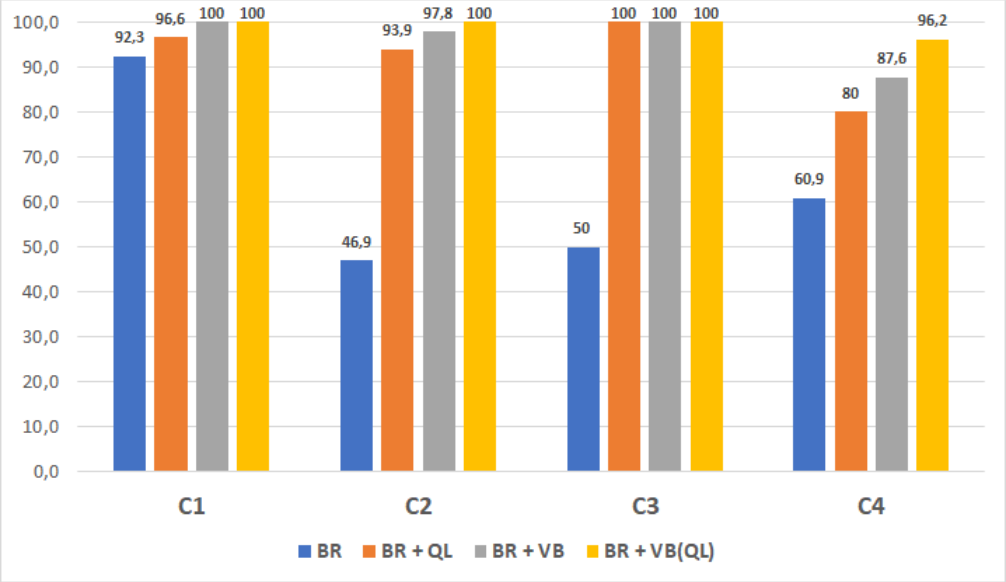
Instâncias		BR			BR + QL			BR + VB			BR + VB(QL)		
Nós	Qt	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio
100	46	84,78	55,22	(0:45:831)	91,30	79,78	(1:11:689)	97,83	83,48	(2:26:982)	100	97,39	(1:56:715)
150	10	20	5,0	(1:30:712)	20,0	5	(4:41:905)	70,00	23	(17:46:91)	100	61	(22:19:617)
200	31	77,42	49,03	(3:37:216)	90,32	71,61	(9:48:801)	87,10	72,90	(19:46:735)	96,77	80,65	(25:16:108)
250	6	16,67	8,33	(8:40:216)	100	85	(12:46:714)	100	88,33	(29:28:299)	100	96,67	(17:30:58)
300	6	0	0	(11:16:29)	50,0	16,67	(90:1:644)	50,00	45	(50:47:863)	83,33	55	(83:16:810)
400	6	0	0	(13:41:543)	50,0	10	(118:25:97)	66,67	56,67	(91:51:607)	66,67	63,33	(119:18:703)
Médias		**60,95**	**39,23**	**(3:28:150)**	**80**	**62,95**	**(16:30:574)**	**87,62**	**71,14**	**(18:26:452)**	**96,13**	**84,57**	**(23:0:932)**

Os resultados da tabela, permitem observar que o BR+VB(QL) foi o que encontrou a maior quantidade de soluções ótimas para classe C4, pois, conseguiu encontrar soluções ótimas para 101 das 105 instâncias (96,1%) das instâncias, e também, foi o que teve a melhor convergência média (84,5%). O segundo melhor algoritmo, foi o BR+VB que obteve as soluções ótimas de 92 das 105 (87,6%). Todos os resultados detalhados para cada uma das instâncias da classe C4, se encontram no Apêndice A deste trabalho.

5.2.3 Análises dos Resultados

Nesta seção, para melhor avaliar o desempenho dos algoritmos, criou-se dois gráfico: o primeiro com percentual de soluções ótimas e outro com percentual de convergência. A figura 22, ilustra um gráfico comparativo do percentual de instância, em que as soluções ótimas/viáveis foram encontradas e a figura 23, ilustra um gráfico comparativo do percentual da convergência média C(%), para os algoritmos desenvolvidos.

Figura 22 – Gráfico com percentual soluções ótimas / viáveis



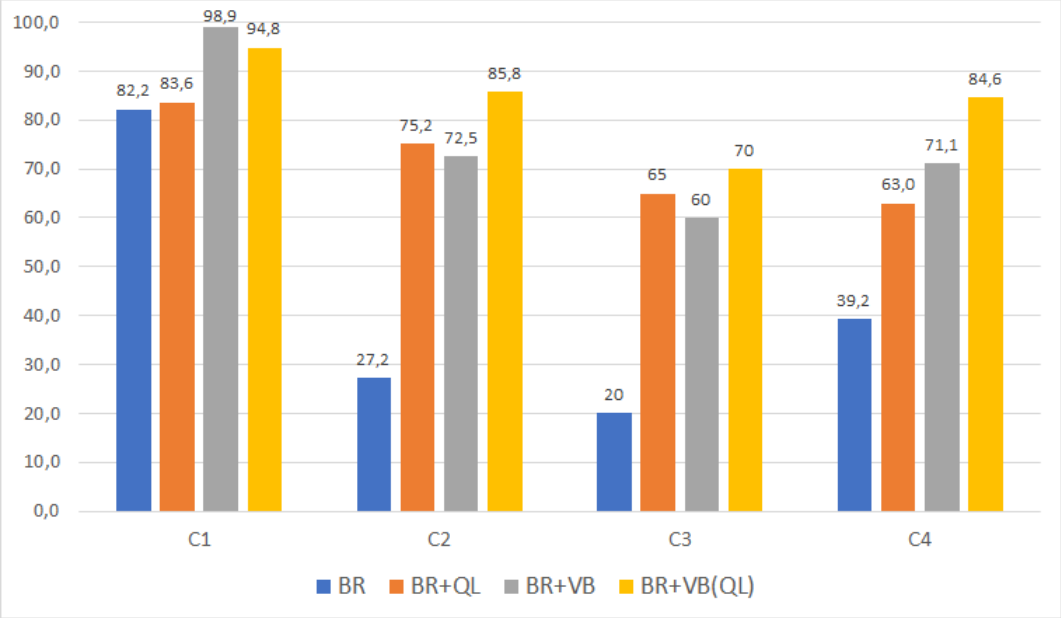
Fonte: Próprio autor

Observando os dados dos gráficos acima, percebemos que para as classes C1, C2, C3, o algoritmo BR+VB(QL) foi o teve o melhor desempenho, encontrou 100,0% das soluções ótimas e para a classe C4 encontrou 96,2% das soluções viáveis. Ainda através do gráfico, verifica-se que o segundo melhor desempenho, foi adquirido pelo BR+VB, que obtive 100% das soluções ótimas para classe C1 e C3. Já o BR, foi o algoritmo que teve o pior desempenho em relação à quantidade de soluções ótimas encontradas.

A figura 23, é possível perceber que para a maioria as classes das instâncias, o BR+VB(QL) teve o melhor percentual de convergência para as soluções encontradas

durantes as dez execuções, exceto para classe C1, que teve percentual de 98,9% no BR+VB *versus* 94,8 do BR+VB(QL).

Figura 23 – Gráfico com percentual de Convergência - C(%)



Fonte: Próprio autor

A partir dos gráficos e demais informações vista até aqui, é possível perceber que os algoritmos BR+VB e BR+VB(QL), se comportaram de maneira bastante semelhantes, obtendo taxas de convergências parecidas, porém, o BR+VB(QL) tem uma vantagem em relação a quantidade de soluções ótimas encontradas para as instâncias do PALAS. Apesar do BR+VB(QL) gastar um maior tempo computacional em relação aos outros algoritmos propostos, os trabalhos sobre o PALAS, estabelecem a quantidade de ótimos encontrados, como o principal fator de comparação com outros trabalhos. Dessa forma, o BR+VB(QL) será escolhida para ser comparado com outros trabalhos da literatura na próxima seção.

5.2.4 Comparação com Heurísticas da Literatura

Na seção anterior, realizou-se uma análise dos algoritmos tanto em relação à qualidade das soluções, quanto à convergência, ficando evidenciada a superioridade do BR+VB(QL) sobre os outros algoritmos propostos. Em vista disso, comparou-se o BR+VB(QL), a resultados existentes na literatura. Como já foi mencionado, a comparação será feita em relação à qualidade média das soluções encontradas, pois, com relação ao desempenho, há questões que prejudicam a análise dos dados, como por exemplo, o tipo de linguagem de programação, equipamentos computacionais utilizados e diferentes modelos computacionais.

A tabela 10, realiza uma comparação dos resultados obtidos para as classes C1, C2, C3 utilizando o algoritmo BR+VB(QL), com os trabalhos mais relevantes disponíveis na literatura: As heurísticas edge-based, cut-based e node-based propostas por Goldschmidt, Laugier e Olinick (GOLDSCHMIDT; LAUGIER; OLINICK, 2003); Em seguida, os resultados encontrados pelo GRASP+PR, proposto por Bastos (BASTOS, 2005); A técnica DMN proposta por Aringhieri e Dell'Amico (ARINGHIERI; DELL'AMICO, 2005); O Algoritmo Memético com Construção de Vocabulário utilizando paralelismo (AM+VB) proposto por Oliveira (OLIVEIRA, 2010); O algoritmo híbrido HBMO (*Honey Bees Mating Optimisation*) proposto por Bernardino (BERNARDINO et al., 2012); e para finalizar, teremos o Multi-Start (MULTI) que foi pesquisado por Oliveira (OLIVEIRA, 2015)

Tabela 10 – Comparação dos resultados obtidos pelo BR-VB(QL)

Instâncias		**GRASP**	**GRASP-PR**	**DMN**	**AM-VB**	**MULT**	**HBMO**	**BR+VB(QL)**
Classes	Qtd.	*(%)	*(%)	*(%)	*(%)	*(%)	*(%)	*(%)
C1	118	100	100	100	100	100	100	100
C2	230	98,69	99,57	98,3	99,57	100	99,57	100
C3	2	-	-	100	-	50	100	100

Analisando os dados da tabela 10, podemos verificar que todos os algoritmos encontraram 100% das soluções ótimas para classe C1. Já em relação à classe C2, o BR+VB(QL) junto com o MULTI, foram os únicos que obtiveram 100% das soluções ótimas das 230 instâncias dessa classe. Para a classe C3, existem poucos trabalhos que pesquisaram essa classe, o algoritmo BR+VB(QL) encontrou 100% das soluções ótimas desta classe, juntamente com o DMN e HDMO. Apesar de alguns trabalhos não ter pesquisados todas as classes (C1, C2, C3 e C4), podemos verificar que o BR+VB(QL) foi o único, dentre os apresentados, que encontrou 100% das soluções ótimas.

A classe C4 como já foi mencionado, não possui o ótimo para suas instâncias, será feito a comparação em termos da quantidade de soluções, em que uma solução viável foi encontrada pelo menos uma vez SV(%). Dessa forma, a tabela 11 traz a comparação dos trabalhos disponíveis na literatura.

Tabela 11 – Comparação dos resultados do C4 obtidos pelo BR+VB(QL) com resultados presentes na literatura.

Instâncias		**GRASP**	**GRASP-PR**	**DMN**	**AM-VB**	**MULTI**	**HBMO**	**BR+VB(QL)**
Classes	Qtd.	SV(%)	SV(%)	SV(%)	SV(%)	SV(%)	SV(%)	SV(%)
C4	105	-	100*	-	100*	-	100	96,16

Existem poucos trabalhos na literatura que utilizaram a classe C4, sendo que, dos trabalhos que foram encontrados, destacamos (*) na tabela 11, significa que só utilizaram parte das instâncias, ou seja, não utilizaram as instâncias com 400 localidades. Analisando os dados da tabela 11, podemos verificar que o BR+VB(QL), não conseguiu encontrar

soluções viáveis para todas as instâncias, achando-o soluções ótimas de 101 das 105 instâncias.

6 Conclusões e Trabalhos Futuros

Neste trabalho, apresentamos quatro implementações de metaheurísticas, nas quais, tiveram sempre como base, a utilização do algoritmo BRKGA, que juntamente com a técnica do *Vocabulary Build* (VB) e o algoritmo *Q-learning*, possibilitaram o desenvolvimento de metaheurísticas híbridas, aplicadas e analisadas para a resolução do Problema de Atribuição de Localidades a Anéis em Redes SONET/SDH. Além da implementação pura da metaheurística BRKGA (BR), desenvolveu-se três metaheurísticas híbridas: O BRKGA com *Q-learning* (BR+QL), tendo o algoritmo de aprendizagem reforço a responsabilidade de criar bons indivíduos da população inicial, em seguida, o BRKGA foi desenvolvido com a técnica do *Vocabulary Build* (BR+VB) e, para finalizar, apresentou-se uma variação do BRKGA com *Vocabulary Build*, que teve a geração dos vocábulos, o auxílio do *Q-learning*, ao invés de geração aleatórias proposta pelo *Vocabulary Build* tradicional, denominado de BR+VB(QL).

O BRGKA (BR) seguiu a mesma implementação disponibilizada pela literatura sem modificações. O BR+QL, possui as mesma etapas do BR, que se modificou apenas na geração da população inicial, pois no início de cada execução, o algoritmo *Q-learning* gera a população inicial. Essas duas implementações, iniciais não utilizaram buscas locais, para facilitar a comparação do resultados gerados pela agregação do *Q-learning* no BRKGA. Nos resultados obtidos, foi possível observar que a agregação do *Q-learning* no algoritmo BRKGA, trouxe melhores resultados para o problema estudado. Outro algoritmo estudado, foi BR+VB, onde consiste na junção da técnica *Vocabulary Build* no Algoritmo BRKGA. Como foi visto, a técnica do VB tem por objetivo, adquirir bons fragmentos (vocábulos) e guarda-los no pool (memória), sendo que esses fragmentos, são adicionados em solução, afim de obter melhores resultados. Os fragmentos são retirados da população elite e gerados de forma aleatória. O BR+VB(QL) seguiu as mesma etapas do algoritmo anterior, mudando apenas na geração do vocábulos, pois essa versão utilizou uma ideia inovadora que foi a utilização do algoritmo *Q-learning* para geração dos vocábulos. Então, foi possível analisar que os resultados entre BR+VB e BR+VB(QL) são parecidos, porém, é notório a vantagem causada pela modificação, com a utilização do Q-learning para geração do vocábulos, do VB para as resoluções da maioria das instâncias trabalhadas.

Para os quatros algoritmos propostos, extensivos experimentos computacionais foram realizados sobre as classes de instâncias C1, C2, C3 e C4. Os algoritmos foram executados 10 vezes sobre cada uma dessas instâncias, utilizando os mesmos parâmetros. Os resultados computacionais foram coletados e avaliados, sob o critério da qualidade das soluções, a média de tempo e quantidade de ótimos encontrados durante as 10 execuções de cada umas das 455 instâncias.

Podemos verificar que o algoritmo BR+VB(QL), foi o que apresentou maior número de instâncias resolvidas de forma ótima para as quatros classes testadas, seguido pelo BR+VB, BR+QL e BR, respectivamente. Em compensação, o BR mostrou-se bem mais rápido que os demais algoritmos.

Os resultados mostraram também que, frente aos algoritmos encontrados na literatura que trabalharam apenas as instâncias as classes C1, C2 e C3, o BR+VB(QL) foi o que conseguiu resultados melhores em termos da qualidade das soluções obtidas. O destaque foi para a classe C2, o BR+VB(QL) foi superior aos algoritmos GRASP, GRASP-PR, DMN, AM-VB, MULT e HBMO. Já para a classe C4, os algoritmos AM-VB e HBMO encontrou 100% das soluções viáveis, enquanto o BR+VB(QL) encontrou 96,16% de soluções viáveis.

Dessa forma, o BR+VB(QL) se mostrou um algoritmo promissor, conseguindo atingir soluções de boa qualidade e em tempo computacional, podendo ser expandido para outros problemas de partição de grafos. Com as instâncias trabalhadas ele conseguiu achar o ótimo em todas, com exceção do conjunto C4.

Para trabalhos futuros, podemos citar as seguintes possibilidades:

- Comparação de outras vizinhanças dentro do VB para que permitam, por exemplo, sair de soluções viáveis para inviáveis.
- Realização de um estudo comparativo com várias políticas de seleção no algoritmo *Q-learning*.
- Desenvolver uma versão paralela do *Vocabulary Build* juntamente com o *Q-learning* (utilizado para geração dos vocábulos). Dessa forma, o *Q-learning* poderá ficar treinando enquanto não for solicitado pelo VB.

Referências

ALMEIDA, I. I. d. *Metaheuristica Híbrida utilizando GRASP Realivo e Aprendizagem por Reforço: Uma Aplicação na Seguranca*. Dissertação (Mestrado) — Universidade do Estado do Rio Grande do Norte, 2014. Citado na página 38.

ARINGHIERI, R.; DELL'AMICO, M. Solution of the sonet ring assignment problem with capacity constraints. In: *Metaheuristic Optimization via Memory and Evolution*. [S.l.]: Springer, 2001. p. 93–116. Citado 3 vezes nas páginas 24, 25 e 52.

ARINGHIERI, R.; DELL'AMICO, M. Comparing metaheuristic algorithms for sonet network design problems. *Journal of Heuristics*, Springer, v. 11, n. 1, p. 35–57, 2005. Citado 2 vezes nas páginas 25 e 60.

BASTOS, L. *Soluções Heurísticas para o Problema de Atribuição de Localidades a Anéis em Redes SONET*. Tese (Doutorado), 2005. Citado 3 vezes nas páginas 26, 52 e 60.

BASTOS, L. O.; OCHI, L. S.; MACAMBIRA, E. M. Grasp problem. In: NETWORK OPTIMIZATION CONFERENCE (INOC 2005 COSPONSORED BY INFORMS - TELECOM SECTION). LISBOA - PORTUGAL. *A relative neighbourhood GRASP for the SONET Ring Assignment Problem.* [S.l.], 2005. Citado na página 26.

BASTOS, L. O.; OCHI, L. S.; MACAMBIRA, E. M. Grasp with path-relinking for the sonet ring assignment problem. In: IEEE. *Hybrid Intelligent Systems, 2005. HIS'05. Fifth International Conference on*. [S.l.], 2005. p. 6–pp. Citado 2 vezes nas páginas 26 e 49.

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, INFORMS, v. 6, n. 2, p. 154–160, 1994. Citado na página 29.

BERNARDINO, E. M.; BERNARDINO, A. M.; SÁNCHEZ-PÉREZ, J. M.; GÓMEZ-PULIDO, J. A.; VEGA-RODRÍGUEZ, M. A. Solving large-scale sonet network design problems using bee-inspired algorithms. *Optical Switching and Networking*, Elsevier, v. 9, n. 2, p. 97–117, 2012. Citado na página 60.

BLUM, C.; ROLI, A. Hybrid metaheuristics: an introduction. In: *Hybrid Metaheuristics*. [S.l.]: Springer, 2008. p. 1–30. Citado na página 28.

CHAN, F. T.; TIBREWAL, R.; PRAKASH, A.; TIWARI, M. Inventory based multi-item lot-sizing problem in uncertain environment: Brkga approach. In: *Advances in Sustainable and Competitive Manufacturing Systems*. [S.l.]: Springer, 2013. p. 1197–1206. Citado na página 29.

COOK; J, W.; CUNNINGHAM; H, W.; PULLEYBLANK, W. A. schrijver combinatorial optimization. John Wiley & Sons, 1998. Citado na página 16.

DELL'AMICO, M.; TRUBIAN, M. Solution of large weighted equicut problems. *European Journal of Operational Research*, Elsevier, v. 106, n. 2-3, p. 500–521, 1998. Citado na página 25.

ERICSSON, M.; RESENDE, M. G. C.; PARDALOS, P. M. A genetic algorithm for the weight setting problem in ospf routing. *Journal of combinatorial optimization*, Springer, v. 6, n. 3, p. 299–333, 2002. Citado na página 14.

FERREIRA, V. D. S. *Um algoritmo híbrido para o problema de roteamento de veículos com frotas heterogêneas.* Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2011. Citado na página 43.

GAREY, M. R.; JOHNSON, D. S. *Computers and intractability.* [S.l.]: wh freeman New York, 2002. Citado na página 16.

GLOVER, F. New ejection chain and alternating path methods for traveling salesman problems. *Computer science and operations research*, Elsevier, v. 449, p. 491–507, 1992. Citado na página 43.

GLOVER, F. M. laguna tabu search. *Modern Heuristic Techniques for Combinatorial*, 1997. Citado na página 43.

GLOVER, F. A template for scatter search and path relinking. *Lecture notes in computer science*, Berlin: Springer-Verlag, 1973-, v. 1363, p. 13–54, 1998. Citado na página 24.

GLOVER, F. Scatter search and path relinking. *New ideas in optimization*, v. 297316, 1999. Citado 2 vezes nas páginas 25 e 43.

GLOVER, F.; LAGUNA, M. *Tabu Search. In: REEVES, Modern Heuristic Techniques for Combinatorial Problems.* [S.l.]: Oxford: Blackwell Scientific Publishing, 1993. 71–140 p. Citado na página 43.

GLOVER, F.; LAGUNA, M.; MARTÍ, R. Fundamentals of scatter search and path relinking. *Control and cybernetics*, v. 29, n. 3, p. 653–684, 2000. Citado na página 43.

GLOVER, F.; LAGUNA, M.; MARTI, R. Scatter search and path relinking: Advances and applications. In: *Handbook of metaheuristics.* [S.l.]: Springer, 2003. p. 1–35. Citado na página 43.

GLOVER, F. W.; KOCHENBERGER, G. A. *Handbook of metaheuristics.* [S.l.]: Springer Science & Business Media, 2006. Citado na página 28.

GOLDSCHMIDT, O.; HOCHBAUM, D. S.; LEVIN, A.; OLINICK, E. V. The sonet edge-partition problem. *Networks*, v. 41, p. 13–23, 2002. Citado na página 23.

GOLDSCHMIDT, O.; LAUGIER, A.; OLINICK, E. V. Sonet/sdh ring assignment with capacity constraints. *Discrete Applied Mathematics*, Elsevier, v. 129, n. 1, p. 99–128, 2003. Citado 6 vezes nas páginas 14, 17, 25, 52, 57 e 60.

GONÇALVES, J. F.; RESENDE, M. G. A biased random-key genetic algorithm for the unequal area facility layout problem. *European Journal of Operational Research*, Elsevier, v. 246, n. 1, p. 86–107, 2015. Citado na página 29.

GONÇALVES, J. F.; RESENDE, M. G.; MENDES, J. J. A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. *Journal of Heuristics*, Springer, v. 17, n. 5, p. 467–486, 2011. Citado 5 vezes nas páginas 29, 30, 31, 32 e 33.

GUEDES, A. d. C.; ALOISE, D. J. Um algoritmo memético para o problema do caixeiro viajante assimétrico: Uma abordagem baseada em vocabulary building. *SBPO. Goiânia, GO:[sn]*, 2006. Citado 4 vezes nas páginas 43, 44, 47 e 48.

KEISER, G. *Comunicações por Fibras Ópticas-4*. [S.l.]: AMGH Editora, 2014. Citado na página 19.

LEE, Y.; SHERALI, H. D.; HAN, J.; KIM, S.-i. A branch-and-cut algorithm for solving an intraring synchronous optical network design problem. *Networks*, v. 35, n. 3, p. 223–232, 2000. Citado na página 52.

LEITE, J. d. F. *Algoritmo Memético e Vocabulary Building: Uma Aplicaçao ao Problema do Caixeiro Viajante Assimétrico*. Dissertação (Mestrado) — Departamento de Informática e Matemática Aplicada, UFRN, Natal, 2006. Citado na página 44.

Lima Júnior, F. C. de. *Algoritmo Q-learning como estratégia de exploração e/ou explotação para metaheurísticas GRASP e algoritmo genético*. Tese (Doutorado) — Programa de Pøs-Graduação em Engenharia Elétrica e Computação ,Universidade Federal do Rio Grande do Norte, 2009. Citado 6 vezes nas páginas 14, 37, 38, 40, 41 e 42.

MACAMBIRA, E. M. *Modelos e Algoritmos de Programação Inteira no Projeto de Redes de Telecomunicações*. Tese (Doutorado) — COPPE, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, 2003. Citado 2 vezes nas páginas 25 e 52.

MACAMBIRA, E. M.; MACULAN, N.; SOUZA, C. C. de. A column generation approach for sonet ring assignment. *Networks*, Wiley Subscription Services, Inc., A Wiley Company, v. 47, n. 3, p. 157–171, 2005. ISSN 1097-0037. Disponível em: <http://dx.doi.org/10.1002/net.20102>. Citado na página 23.

MACAMBIRA, E. M.; SOUZA, C. C. de. Projeto de uma rede de telecomunicações usando metaheurísticas. *XXXV Simpósio Brasileiro de Pesquisa Operacional*, v. 4, 2003. Citado na página 25.

MAINIERI, G. B. *Meta-heurística BRKGA aplicada a um problema de programação de tarefas no ambiente flowshop híbrido*. Tese (Doutorado) — Universidade de São Paulo, 2014. Citado na página 30.

NASCIMENTO, J. P. Lima do; ALOISE, D. J. Algoritmo memético com vocabulary building para o problema de roteamento de unidades móveis de pistoneio. *Revista GEPROS*, n. 1, p. 81, 2014. Citado 3 vezes nas páginas 29, 43 e 48.

OLIVEIRA, T. H. F. D. *Algoritmos Híbridos para o Problema de Atribuição de Localidades a Aneis em redes SONET/SDH*. Dissertação (Mestrado) — Universidade do Estado do Rio Grande do Norte, 2015. Citado 3 vezes nas páginas 26, 29 e 60.

OLIVEIRA, W. d. *Algoritmo Evolutivo Paralelo para o Problema de Atribuição de Localidades a Aneis em redes SONET/SDH*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2010. Citado 6 vezes nas páginas 17, 18, 22, 26, 29 e 60.

OMIDYAR, C.; ALDRIDGE, A. Sdh/sonet. *Special Issue, IEEE Commun. Mag*, 1993. Citado na página 16.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial optimization: algorithms and complexity*. [S.l.]: Courier Corporation, 1982. Citado na página 16.

PERROS, H. G. Connection-oriented networks: Sonet. *SDH, ATM, MPLS*, 2005. Citado na página 21.

POMARI, C. Z.; CHAVES, A. A. Algoritmo híbrido com cs e brkga aplicado ao problema de alocação de berços. *SBPO-Simpósio Brasileiro de Pesquisa Operacional. Salvador–BA*, 2014. Citado na página 29.

RAMASWAMI, R. Kumar n sivarajan–optical networks. *A practical perspective*, 1998. Citado na página 18.

ROQUE, L. A.; FONTES, D. B.; FONTES, F. A. A biased random key genetic algorithm approach for unit commitment problem. In: SPRINGER. *International Symposium on Experimental Algorithms*. [S.l.], 2011. p. 327–339. Citado na página 29.

SAMANTA, S.; PAL, M. Telecommunication system based on fuzzy graphs. *J. Telecommun. Syst. Manag*, v. 3, n. 1, p. 1–6, 2013. Citado na página 13.

SILVA, A. C. G. *Busca heurística através de algoritmo genético e memético com construção de vocábulos para o problema de atribuição de localidades a anéis Sonet.* Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2008. Citado 3 vezes nas páginas 26, 50 e 51.

SILVA, R.; RESENDE, M.; PARDALOS, P.; GONÇALVES, J. Biased random-key genetic algorithm for bound-constrained global optimization. In: *GOW 2012: Proceedings of Global Optimization Workshop*. [S.l.: s.n.], 2012. Citado na página 31.

SILVA, R. M.; RESENDE, M. G.; PARDALOS, P. M. A python/c++ library for bound-constrained global optimization using a biased random-key genetic algorithm. *Journal of Combinatorial Optimization*, Springer, v. 30, n. 3, p. 710–728, 2015. Citado na página 29.

SOARES, W. K. d. S. *Heurísticas usando construção de vocabuilário aplicadas ao problema da atribuição de localidades a anéis em redes SONET/SDH*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2009. Citado 4 vezes nas páginas 26, 43, 53 e 57.

SORIANO, P.; WYNANTS, C.; SÉGUIN, R.; LABBÉ, M.; GENDREAU, M.; FORTZ, B. Design and dimensioning of survivable sdh/sonet networks. In: *Telecommunications network planning*. [S.l.]: Springer, 1999. p. 147–167. Citado na página 17.

SPEARS, W. M.; JONG, K. D. D. *On the virtues of parameterized uniform crossover*. [S.l.], 1995. Citado na página 31.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press Cambridge, 1998. Citado 3 vezes nas páginas 37, 38 e 41.

TALBI, E.-G. A taxonomy of hybrid metaheuristics. *Journal of heuristics*, Kluwer Academic Publishers, v. 8, n. 5, p. 541–564, 2002. Citado na página 14.

TEKTRONIX SONET Telecommunications Standard Primer. 2001. <http://www.tek.com/dl/2RW_11407_2.pdf>. Accessed: 2017-03-02. Citado na página 19.

UZINSKI, H. Otimização de problemas multimodais usando meta-heurísticas evolutivas. Universidade Estadual Paulista (UNESP), 2014. Citado na página 30.

WASEM, O. J.; WU, T.-H.; CARDWELL, R. H. Survivable sonet networks/spl mdash/design methodology. *IEEE Journal on Selected Areas in Communications*, IEEE, v. 12, n. 1, p. 205–212, 1994. Citado na página 16.

WATKINS, C. J. C. H. *Learning from delayed rewards*. Tese (Doutorado) — University of Cambridge England, 1989. Citado 3 vezes nas páginas 39, 40 e 41.

Apêndices

APÊNDICE A – Resultados Detalhados

Tabela 12 – Resultados das instâncias geométricas (GS) da classe C1 com B = 155 MB/s.

Instâncias		BR			BR+QL			BR+VB			BR+VB(QL)		
Nome	*	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)
GS_15_1	3	(0:0:9)	3	100	(0:2:579)	3	100	(0:0:91)	3	100	(0:0:325)	3	100
GS_15_4	3	(0:0:4)	3	100	(0:2:994)	3	100	(0:0:50)	3	100	(0:0:218)	3	100
GS_15_7	3	(0:0:3)	3	100	(0:2:925)	3	100	(0:0:60)	3	100	(0:0:213)	3	100
GS_15_9	3	(0:0:2)	3	100	(0:2:881)	3	100	(0:0:5)	3	100	(0:0:5)	3	100
GS_25_1	4	(0:0:3)	4	100	(0:6:330)	4	100	(0:0:209)	4	100	(0:0:665)	4	100
GS_25_3	3	(0:0:21)	3	100	(0:9:269)	3	100	(0:0:254)	3	100	(0:2:533)	3	100
GS_25_4	4	(0:0:20)	4	100	(0:5:756)	4	100	(0:0:236)	4	100	(0:0:423)	4	100
GS_25_7	3	(0:0:16)	3	100	(0:6:120)	3	100	(0:0:215)	3	100	(0:0:779)	3	100
GS_25_8	4	(0:0:357)	4	60	(0:14:919)	4	80	(0:0:385)	4	100	(0:1:667)	4	100
GS_30_1	4	(0:0:343)	4	100	(0:8:358)	4	100	(0:0:521)	4	100	(0:0:827)	4	100
GS_30_2	4	(0:0:284)	4	100	(0:9:816)	4	100	(0:0:381)	4	100	(0:0:856)	4	100
GS_30_3	4	(0:0:219)	4	100	(0:7:678)	4	100	(0:0:362)	4	100	(0:1:38)	4	100
GS_30_4	4	(0:0:271)	4	100	(0:7:840)	4	100	(0:0:405)	4	100	(0:0:644)	4	100
GS_30_5	4	(0:0:875)	7	100	(0:23:80)	4	100	(0:0:379)	4	30	(0:0:668)	4	100
GS_30_6	4	(0:0:298)	4	100	(0:8:119)	4	100	(0:0:361)	4	100	(0:0:806)	4	100
GS_30_7	4	(0:0:378)	4	90	(0:7:265)	4	100	(0:0:368)	4	100	(0:2:759)	4	100
GS_30_8	4	(0:0:225)	4	100	(0:8:114)	4	100	(0:0:361)	4	100	(0:1:683)	4	100
GS_30_9	4	(0:0:434)	4	100	(0:7:725)	4	100	(0:0:354)	4	100	(0:1:44)	4	100
GS_50_1	5	(0:0:47)	5	80	(0:37:639)	5	100	(0:4:639)	5	100	(0:23:185)	5	100
GS_50_4	6	(0:0:86)	6	100	(0:22:582)	6	100	(0:3:25)	6	100	(0:16:936)	6	100
GS_50_7	5	(0:0:51)	5	80	(0:22:117)	5	50	(0:5:3)	5	100	(0:21:710)	5	100
GS_50_8	5	(0:0:157)	5	100	(0:24:504)	5	80	(0:6:684)	5	100	(1:7:585)	5	100
GS_50_9	4	(0:0:64)	4	60	(0:22:653)	4	80	(0:4:34)	4	100	(0:30:841)	4	100
Médias		**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**
		(0:0:181)	**100**	**94,35**	**(0:11:794)**	**100**	**95,22**	**(0:1:234)**	**100**	**97**	**(0:7:713)**	**100**	**100**

Tabela 13 – Resultados das instâncias geométricas (GL) da classe C1 com B = 622 MB/s.

Instâncias		BR			BR+QL			BR+VB			BR+VB(QL)		
Nome	*	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)
GL_15_1	3	(0:0:1)	3	100	(0:1:886)	3	100	(0:0:11)	3	100	(0:0:20)	3	100
GL_15_2	3	(0:0:15)	3	100	(0:1:483)	3	100	(0:0:5)	3	100	(0:0:15)	3	100
GL_15_3	2	(0:0:10)	2	100	(0:1:497)	2	100	(0:0:32)	2	100	(0:0:339)	2	100
GL_15_6	2	(0:0:4)	2	100	(0:1:680)	2	100	(0:0:9)	2	100	(0:0:92)	2	100
GL_15_7	2	(0:0:3)	2	100	(0:1:615)	2	100	(0:0:4)	2	100	(0:0:10)	2	100
GL_15_8	3	(0:0:51)	3	100	(0:1:853)	3	100	(0:0:65)	3	100	(0:0:517)	3	100
GL_15_9	3	(0:0:1)	3	100	(0:1:636)	3	100	(0:0:5)	3	100	(0:0:41)	3	100
GL_15_10	2	(0:0:115)	2	100	(0:1:486)	2	100	(0:0:4)	2	100	(0:0:6)	2	100
GL_25_1	3	(0:0:35)	3	100	(0:4:136)	3	100	(0:0:161)	3	100	(0:0:437)	3	100
GL_25_2	2	(0:0:437)	2	100	(0:3:964)	2	100	(0:0:188)	2	100	(0:14:618)	3	100
GL_25_3	2	(0:0:732)	2	80	(0:4:816)	2	70	(0:0:281)	2	100	(0:14:212)	2	90
GL_25_4	3	(0:0:150)	3	100	(0:4:210)	3	100	(0:0:182)	3	100	(0:1:92)	3	100
GL_25_7	4	(0:0:544)	4	90	(0:4:837)	4	100	(0:0:208)	4	100	(0:0:537)	4	100
GL_25_8	3	(0:0:101)	3	100	(0:3:928)	3	100	(0:0:183)	3	100	(0:0:573)	3	100
GL_25_9	3	(0:0:87)	3	100	(0:4:81)	3	100	(0:0:184)	3	100	(0:0:598)	3	100
GL_30_1	3	(0:0:256)	3	100	(0:5:857)	3	100	(0:0:348)	3	100	(0:4:137)	3	100
GL_30_2	4	(0:0:245)	4	100	(0:8:209)	4	100	(0:0:373)	4	100	(0:1:273)	4	100
GL_30_3	4	(0:0:322)	4	100	(0:9:260)	4	100	(0:0:355)	4	100	(0:1:314)	4	100
GL_30_4	3	(0:0:438)	3	100	(0:6:793)	3	100	(0:0:391)	3	100	(0:2:250)	3	100
GL_30_5	4	(0:0:320)	4	100	(0:6:136)	4	100	(0:0:367)	4	100	(0:0:705)	4	100
GL_30_9	4	(0:0:251)	4	90	(0:5:982)	4	100	(0:0:375)	4	100	(0:0:740)	4	100
GL_30_10	3	(0:0:38)	3	100	(0:5:407)	3	100	(0:0:177)	3	100	(0:0:487)	3	100
GL_50_1	5	(0:3:766)	5	30	(0:20:957)	5	60	(0:3:971)	5	100	(0:14:498)	5	100
GL_50_4	4	(0:4:720)	4	50	(0:21:898)	4	100	(0:38:470)	4	90	(1:15:231)	4	100
GL_50_5	5	(0:2:130)	5	80	(0:19:782)	5	100	(0:3:666)	5	100	(0:7:449)	5	100
GL_50_6	5	(0:4:45)	5	30	(0:21:358)	5	30	(0:7:232)	5	100	(0:15:376)	5	100
GL_50_7	5	(0:3:213)	5	40	(0:19:744)	5	60	(0:3:744)	5	100	(0:8:98)	5	100
Médias		**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**
		(0:0:816)	**100**	**88,5**	**(0:7:203)**	**100**	**93,3**	**(0:2:259)**	**100**	**99,6**	**(0:6:99)**	**100**	**99,6**

Tabela 14 – Resultados das instâncias aleatórias RS da classe C1 com B = 155 MB/s.

Instâncias		BR			BR+QL			BR+VB			BR+VB(QL)		
Nome	*	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)
RS_15_1	3	(0:0:23)	3	100	(0:1:996)	3	100	(0:0:11)	3	100	(0:0:18)	3	100
RS_15_3	2	(0:0:8)	2	100	(0:1:479)	2	100	(0:0:5)	2	100	(0:0:11)	2	100
RS_15_4	3	(0:0:24)	3	100	(0:1:74)	3	100	(0:0:78)	3	100	(0:0:309)	3	100
RS_15_6	3	(0:0:7)	3	100	(0:1:540)	3	100	(0:0:4)	3	100	(0:0:4)	3	100
RS_15_8	3	(0:0:4)	3	100	(0:1:576)	3	100	(0:0:3)	3	100	(0:0:4)	3	100
RS_15_9	3	(0:0:5)	3	100	(0:1:597)	3	100	(0:0:3)	3	100	(0:0:3)	3	100
RS_15_10	3	(0:0:41)	3	100	(0:1:585)	3	100	(0:0:49)	3	100	(0:0:220)	3	100
RS_25_1	4	(0:1:101)	-	0	(0:5:346)	4	20	(0:3:308)	4	80	(0:18:858)	4	100
RS_25_2	3	(0:0:193)	3	100	(0:4:577)	3	90	(0:0:179)	3	100	(0:1:791)	3	100
RS_25_4	4	(0:1:56)	-	0	(0:5:5)	4	10	(0:0:295)	4	100	(0:0:761)	4	100
RS_25_5	4	(0:0:136)	4	100	(0:3:111)	4	100	(0:0:184)	4	100	(0:0:406)	4	100
RS_25_6	4	(0:0:706)	4	70	(0:3:439)	4	50	(0:0:227)	4	100	(0:0:530)	4	100
RS_25_7	4	(0:0:327)	4	100	(0:3:248)	4	100	(0:0:200)	4	100	(0:0:416)	4	100
RS_25_8	3	(0:0:28)	3	100	(0:2:917)	3	100	(0:0:174)	3	100	(0:0:333)	3	100
RS_25_9	4	(0:0:305)	4	90	(0:3:201)	4	90	(0:0:206)	4	100	(0:0:406)	4	100
RS_25_10	4	(0:0:10)	4	100	(0:3:238)	4	100	(0:0:205)	4	100	(0:0:432)	4	100
RS_30_1	3	(0:0:17)	3	100	(0:4:230)	3	100	(0:0:346)	3	100	(0:3:838)	3	100
RS_30_3	4	(0:0:519)	4	10	(0:5:67)	4	20	(0:0:451)	4	100	(0:1:255)	4	100
RS_30_4	4	(0:1:345)	4	20	(0:4:571)	4	70	(0:0:711)	4	100	(0:11:509)	4	90
RS_30_5	4	(0:0:16)	4	100	(0:4:90)	4	100	(0:0:360)	4	100	(0:0:586)	4	100
RS_30_6	4	(0:1:271)	4	20	(0:5:16)	4	10	(0:5:816)	4	90	(0:33:546)	4	70
RS_30_7	3	(0:0:18)	3	100	(0:4:759)	3	80	(0:4:853)	3	100	(0:4:184)	3	100
RS_30_8	4	(0:0:14)	4	100	(0:4:100)	4	100	(0:0:368)	4	100	(0:0:582)	4	100
RS_30_10	4	(0:1:622)	-	0	(0:4:799)	4	40	(0:0:381)	4	100	(0:0:598)	4	100
RS_50_1	5	(0:0:351)	-	0	(0:16:612)	5	20	(0:7:472)	5	100	(0:9:905)	5	100
RS_50_3	4	(0:0:528)	4	100	(0:15:422)	4	80	(0:44:491)	5	80	(2:45:26)	4	60
RS_50_4	4	(0:0:276)	4	100	(0:14:748)	4	100	(0:57:9)	4	90	(3:12:888)	4	100
RS_50_5	4	(0:0:51)	4	100	(0:14:775)	4	80	(0:4:913)	4	100	(0:14:808)	4	100
RS_50_6	4	(0:0:62)	4	100	(0:14:958)	4	60	(0:4:86)	4	50	(0:15:88)	4	100
RS_50_7	5	(0:5:571)	-	0	(0:16:300)	5	10	(0:53:947)	5	80	(0:16:233)	5	100
RS_50_10	4	(0:0:131)	4	100	(0:16:995)	4	20	(0:4:107)	4	100	(0:49:136)	4	100
Médias		**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**
		(0:0:564)	**83,87**	**74,52**	**(0:6:180)**	**100**	**72,58**	**(0:11:504)**	**100**	**95,81**	**(0:20:424)**	**100**	**97,42**

Tabela 15 – Resultados das instâncias aleatórias (RL) da classe C1 com B = 622 MB/s.

Instâncias		BR			BR+QL			BR+VB			BR+VB(QL)		
Nome	*	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)
RL_15_1	3	(0:0:111)	3	100	(0:1:109)	3	100	(0:0:5)	3	100	(0:0:7)	3	100
RL_15_3	3	(0:0:141)	3	100	(0:1:70)	3	100	(0:0:38)	3	100	(0:0:185)	3	100
RL_15_4	2	(0:0:4)	2	100	(0:1:74)	2	100	(0:0:11)	2	100	(0:0:9)	2	100
RL_15_5	3	(0:0:2)	3	100	(0:1:86)	3	100	(0:0:2)	3	100	(0:0:3)	3	100
RL_15_6	3	(0:0:1)	3	100	(0:1:81)	3	100	(0:0:2)	3	100	(0:0:3)	3	100
RL_15_7	2	(0:0:2)	2	100	(0:1:69)	2	100	(0:0:4)	2	100	(0:0:3)	2	100
RL_15_8	2	(0:0:2)	2	100	(0:1:195)	2	100	(0:0:45)	2	100	(0:2:294)	2	100
RL_15_9	3	(0:0:2)	3	100	(0:1:140)	3	100	(0:0:6)	3	100	(0:0:3)	3	100
RL_25_1	3	(0:0:611)	3	50	(0:4:46)	3	90	(0:0:221)	3	100	(0:0:797)	3	100
RL_25_2	3	(0:0:7)	3	90	(0:4:537)	3	100	(0:0:6)	3	100	(0:0:7)	3	100
RL_25_3	3	(0:0:9)	3	100	(0:2:978)	3	100	(0:0:7)	3	100	(0:0:50)	3	100
RL_25_4	3	(0:0:655)	3	100	(0:3:637)	3	100	(0:4:724)	3	100	(0:20:568)	3	70
RL_25_5	4	(0:0:724)	-	0	(0:3:545)	-	0	(0:13:740)	4	40	(0:28:180)	4	80
RL_25_6	4	(0:0:281)	4	100	(0:3:137)	4	100	(0:0:180)	4	60	(0:0:408)	4	100
RL_25_7	3	(0:0:8)	3	100	(0:2:996)	3	100	(0:0:9)	3	100	(0:0:48)	3	100
RL_25_8	4	(0:0:553)	4	80	(0:3:466)	4	100	(0:0:178)	4	100	(0:0:423)	4	100
RL_25_9	2	(0:0:10)	2	100	(0:3:31)	2	100	(0:0:11)	2	100	(0:0:9)	2	100
RL_25_10	3	(0:0:7)	3	100	(0:3:45)	3	100	(0:0:7)	3	100	(0:0:10)	3	100
RL_30_1	3	(0:0:10)	3	100	(0:4:507)	3	100	(0:0:9)	3	100	(0:0:9)	3	100
RL_30_2	4	(0:0:586)	4	70	(0:4:185)	4	90	(0:0:588)	4	100	(0:0:545)	4	100
RL_30_3	3	(0:0:14)	3	100	(0:3:932)	3	100	(0:0:15)	3	100	(0:0:139)	3	100
RL_30_4	3	(0:0:10)	3	100	(0:4:352)	3	100	(0:0:334)	3	100	(0:0:702)	3	100
RL_30_6	4	(0:1:91)	4	10	(0:4:962)	-	0	(0:5:679)	4	90	(0:10:310)	4	100
RL_30_7	4	(0:0:615)	4	80	(0:4:291)	4	100	(0:0:336)	4	100	(0:0:582)	4	100
RL_30_8	4	(0:1:7)	4	100	(0:4:14)	4	100	(0:0:497)	4	50	(0:0:563)	4	90
RL_30_9	4	(0:1:117)	4	100	(0:4:35)	4	100	(0:0:440)	4	100	(0:0:615)	4	100
RL_30_10	4	(0:1:183)	4	100	(0:4:804)	4	100	(0:0:333)	4	100	(0:0:570)	4	100
RL_50_1	4	(0:0:39)	4	100	(0:14:398)	4	100	(0:3:215)	4	100	(0:8:873)	4	100
RL_50_2	3	(0:0:52)	3	100	(0:15:471)	4	100	(0:3:848)	3	100	(2:0:575)	3	80
RL_50_3	4	(0:3:391)	-	0	(0:15:93)	-	0	(4:17:156)	5	50	(4:33:698)	4	70
RL_50_4	4	(0:3:413)	4	10	(0:14:643)	-	0	(1:43:376)	4	70	(2:34:453)	4	90
RL_50_5	3	(0:0:41)	3	100	(0:14:247)	3	70	(0:2:460)	3	100	(0:17:595)	3	100
RL_50_6	3	(0:0:42)	3	100	(0:15:109)	3	100	(0:2:519)	3	100	(0:20:449)	3	100
RL_50_7	5	(0:3:390)	-	0	(0:15:480)	5	10	(2:45:701)	5	30	(2:4:731)	5	100
RL_50_8	4	(0:0:33)	-	0	(0:13:137)	4	100	(0:2:631)	4	100	(0:11:692)	4	100
RL_50_9	4	(0:0:33)	4	20	(0:13:219)	4	100	(0:2:397)	4	100	(0:3:531)	4	100
RL_50_10	4	(0:3:407)	4	20	(0:15:734)	4	10	(0:17:94)	4	100	(0:42:519)	4	100
Médias		**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**
		(0:0:611)	**89,2**	**76,5**	**(0:6:185)**	**89,2**	**83,0**	**(0:15:887)**	**100**	**91,6**	**(0:22:842)**	**100**	**96,8**

Tabela 16 – Resultados das instâncias new.GL da classe C2 com B = 155 MB/s

Instâncias		BR			BR + QL			BRKGA + VB			BRKGA + VB(QL)		
Nome	*	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)
new.GL_15_3.1	4	(0:0:89)	4	100	(0:0:63)	4	100	(0:0:60)	4	100	(0:0:352)	4	100
new.GL_15_3.2	4	(0:0:137)	4	80	(0:0:67)	4	100	(0:0:101)	4	100	(0:0:299)	4	100
new.GL_15_3.3	4	(0:0:70)	4	100	(0:0:79)	4	100	(0:0:55)	4	100	(0:3:806)	4	90
new.GL_15_3.4	4	(0:0:46)	4	100	(0:0:48)	4	100	(0:0:28)	4	100	(0:0:226)	4	100
new.GL_15_3.5	4	(0:0:104)	4	100	(0:0:157)	4	100	(0:0:147)	4	100	(0:0:252)	4	100
new.GL_15_3.6	4	(0:0:166)	4	80	(0:0:172)	4	100	(0:0:44)	4	100	(0:0:228)	4	100
new.GL_15_3.7	4	(0:0:27)	4	100	(0:0:72)	4	100	(0:0:33)	4	100	(0:0:216)	4	100
new.GL_15_3.8	4	(0:0:83)	4	90	(0:0:70)	4	100	(0:0:68)	4	100	(0:0:271)	4	100
new.GL_15_3.9	4	(0:0:168)	4	90	(0:0:182)	4	100	(0:0:52)	4	100	(0:0:272)	4	100
new.GL_15_3.10	4	(0:0:70)	4	100	(0:0:68)	4	100	(0:0:26)	4	100	(0:0:222)	4	100
new.GL_15_6.1	3	(0:0:68)	3	100	(0:0:89)	3	100	(0:0:233)	3	100	(0:0:269)	3	100
new.GL_15_6.2	3	(0:0:21)	3	100	(0:0:241)	3	100	(0:1:61)	3	90	(0:6:567)	3	90
new.GL_15_6.3	3	(0:0:99)	3	100	(0:0:40)	3	100	(0:0:54)	3	100	(0:0:265)	3	100
new.GL_15_6.4	3	(0:0:84)	3	90	(0:0:468)	3	100	(0:0:545)	3	90	(0:19:44)	3	50
new.GL_15_6.5	3	(0:0:139)	3	80	(0:0:864)	3	100	(0:1:154)	3	70	(0:1:50)	3	100
new.GL_15_6.6	3	(0:0:62)	3	100	(0:0:190)	3	100	(0:0:144)	3	100	(0:3:762)	3	100
new.GL_15_6.7	3	(0:0:88)	3	100	(0:0:89)	3	100	(0:0:345)	3	100	(0:6:331)	3	90
new.GL_15_6.8	3	(0:0:83)	3	100	(0:0:229)	3	100	(0:0:206)	3	100	(0:4:212)	3	100
new.GL_15_6.9	3	(0:0:71)	3	100	(0:0:180)	3	100	(0:0:73)	3	100	(0:0:215)	3	100
new.GL_15_6.10	3	(0:0:27)	3	100	(0:0:39)	3	100	(0:0:25)	3	100	(0:0:230)	3	100
new.GL_25_9.1	4	(0:0:671)	4	40	(0:5:389)	4	80	(0:1:439)	4	90	(0:28:468)	4	80
new.GL_25_9.2	4	(0:0:633)	4	20	(0:2:763)	4	90	(0:0:832)	4	100	(0:7:376)	4	90
new.GL_25_9.3	4	(0:0:606)	4	30	(0:0:295)	4	100	(0:0:724)	4	100	(0:1:473)	4	100
new.GL_25_9.4	4	(0:0:617)	4	20	(0:0:393)	4	100	(0:1:25)	4	100	(0:1:180)	4	100
new.GL_25_9.5	4	(0:0:703)	4	10	(0:0:359)	4	100	(0:1:479)	4	100	(0:0:745)	4	100
new.GL_25_9.6	4	(0:0:178)	4	100	(0:0:210)	4	100	(0:0:130)	4	100	(0:0:450)	4	100
new.GL_25_9.7	4	(0:0:592)	4	10	(0:2:790)	4	90	(0:0:251)	4	100	(0:0:929)	4	100
new.GL_25_9.8	4	(0:0:603)	4	50	(0:0:284)	4	100	(0:0:280)	4	100	(0:0:908)	4	100
new.GL_25_9.9	4	(0:0:490)	4	80	(0:0:376)	4	100	(0:0:210)	4	100	(0:0:570)	4	100
new.GL_25_9.10	4	(0:0:402)	4	90	(0:0:347)	4	100	(0:0:169)	4	100	(0:0:647)	4	100
new.GL_25_10.1	5	(0:0:565)	5	50	(0:13:767)	5	40	(0:4:257)	5	60	(0:13:33)	5	80
new.GL_25_10.2	5	(0:0:564)	5	10	(0:5:79)	5	80	(0:0:664)	5	100	(0:0:587)	5	100
new.GL_25_10.3	5	(0:0:489)	5	30	(0:0:241)	5	100	(0:0:372)	5	100	(0:1:736)	5	100
new.GL_25_10.4	5	(0:0:560)	5	20	(0:4:582)	5	80	(0:0:151)	5	100	(0:0:418)	5	100
new.GL_25_10.5	5	(0:0:608)	5	10	(0:6:492)	5	70	(0:0:422)	5	100	(0:1:281)	5	100

Tabela 17 – Resultados das instâncias new.GL da classe C2 com B = 155 MB/s (continuação)

Instâncias		BR			BR + QL			BRKGA + VB			BRKGA + VB(QL)		
Nome	*	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)
new.GL_25_10.6	5	(0:0:586)	5	20	(0:0:259)	5	100	(0:0:191)	5	100	(0:0:499)	5	100
new.GL_25_10.7	5	(0:0:594)	-	0	(0:0:314)	5	100	(0:1:338)	5	100	(0:1:86)	5	100
new.GL_25_10.8	5	(0:0:657)	-	0	(0:0:269)	5	100	(0:0:690)	5	100	(0:0:531)	5	100
new.GL_25_10.9	5	(0:0:613)	5	20	(0:0:383)	5	100	(0:0:319)	5	100	(0:0:716)	5	100
new.GL_25_10.10	5	(0:0:565)	5	30	(0:0:302)	5	100	(0:0:409)	5	100	(0:0:797)	5	100
new.GL_30_10.1	5	(0:0:865)	5	10	(0:2:839)	5	100	(0:3:812)	5	90	(0:1:418)	5	100
new.GL_30_10.2	5	(0:0:833)	5	20	(0:0:448)	5	100	(0:0:944)	5	100	(0:0:744)	5	100
new.GL_30_10.3	5	(0:0:746)	5	20	(0:0:738)	5	100	(0:1:924)	5	100	(0:0:989)	5	100
new.GL_30_10.4	5	(0:0:946)	-	0	(0:0:836)	5	100	(0:0:422)	5	100	(0:0:935)	5	100
new.GL_30_10.5	5	(0:0:968)	-	0	(0:1:49)	5	100	(0:0:901)	5	100	(0:0:874)	5	100
new.GL_30_10.6	5	(0:0:855)	5	10	(0:0:479)	5	100	(0:0:441)	5	100	(0:0:707)	5	100
new.GL_30_10.7	5	(0:0:852)	5	20	(0:0:643)	5	100	(0:3:8)	5	90	(0:1:328)	5	100
new.GL_30_10.8	5	(0:0:783)	5	10	(0:0:436)	5	100	(0:0:569)	5	100	(0:0:645)	5	100
new.GL_30_10.9	5	(0:0:834)	5	10	(0:0:891)	5	100	(0:2:407)	5	100	(0:0:878)	5	100
new.GL_30_10.10	5	(0:0:841)	-	0	(0:0:470)	5	100	(0:0:754)	5	100	(0:1:79)	5	100
new.GL_50_6.1	6	(0:2:640)	-	0	(1:55:528)	6	60	(1:9:228)	-	0	(2:0:213)	6	100
new.GL_50_6.2	6	(0:3:955)	-	0	(2:2:487)	6	50	(1:5:796)	6	20	(3:36:281)	6	20
new.GL_50_6.3	6	(0:2:918)	-	0	(1:46:328)	6	60	(1:3:832)	6	10	(3:8:355)	6	30
new.GL_50_6.4	6	(0:2:984)	-	0	(2:32:131)	6	30	(0:50:571)	6	20	(3:29:815)	6	30
new.GL_50_6.5	6	(0:4:14)	6	10	(0:45:679)	6	80	(0:55:484)	6	30	(2:9:313)	6	60
new.GL_50_6.6	6	(0:3:692)	-	0	(0:5:854)	6	100	(0:21:899)	6	80	(0:52:216)	6	80
new.GL_50_6.7	6	(0:2:512)	-	0	(1:37:715)	6	60	(0:52:278)	6	30	(3:27:480)	6	40
new.GL_50_6.8	6	(0:3:350)	-	0	(1:7:796)	6	70	(0:43:974)	6	40	(2:5:707)	6	70
new.GL_50_6.9	6	(0:3:351)	-	0	(0:8:280)	6	100	(0:25:86)	6	80	(0:18:8)	6	100
new.GL_50_6.10	6	(0:3:126)	6	20	(0:7:181)	6	100	(0:16:635)	6	100	(0:51:371)	6	100
new.GL_50_10.1	5	(0:3:864)	6	0	(3:55:685)	6	100	(1:32:216)	6	100	(6:2:524)	6	100
new.GL_50_10.2	5	(0:3:768)	-	0	(3:55:883)	5	20	(1:29:230)	6	50	(5:45:53)	6	90
new.GL_50_10.3	6	(0:3:768)	-	0	(3:36:490)	6	30	(1:24:859)	-	0	(4:27:461)	6	20
new.GL_50_10.4	5	(0:3:780)	6	0	(4:1:418)	6	100	(1:22:225)	5	20	(5:25:175)	6	100
new.GL_50_10.5	5	(0:2:479)	5	60	(3:9:530)	5	30	(1:18:921)	6	100	(4:51:839)	6	100
new.GL_50_10.6	5	(0:3:821)	-	0	(2:43:59)	5	10	(1:30:901)	6	60	(4:57:914)	5	10
new.GL_50_10.7	6	(0:3:970)	-	0	(2:28:834)	6	30	(1:11:699)	6	60	(1:51:920)	6	80
new.GL_50_10.8	5	(0:3:854)	6	0	(2:26:772)	5	30	(1:32:357)	6	60	(5:28:380)	6	100
new.GL_50_10.9	6	(0:2:482)	6	50	(0:27:960)	6	90	(0:55:638)	6	50	(0:29:190)	6	100
new.GL_50_10.10	6	(0:3:778)	-	0	(2:39:367)	6	10	(1:27:769)	-	0	(3:39:832)	6	50
Médias		**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**
		(0:1:280)	**70**	**39,86**	**(0:36:444)**	**100**	**85,57**	**(0:18:937)**	**95,71**	**82,71**	**(0:57:417)**	**100**	**89,29**

Tabela 18 – Resultados das instâncias geométricas new.GH da classe C2 com B = 622 MB/s.

Instâncias		BRKGA			BRKGA + Q-LEARNING			BRKGA + VB			BRKGA + VB(QL)		
Nome	*	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)
new.GH_25_5.1	4	(0:0:867)	4	20	(0:0:382)	4	100	(0:0:513)	4	100	(0:0:612)	4	100
new.GH_25_5.2	4	(0:0:505)	4	60	(0:0:265)	4	100	(0:0:256)	4	100	(0:0:428)	4	100
new.GH_25_5.3	4	(0:0:727)	4	60	(0:0:343)	4	100	(0:3:0)	4	100	(0:0:881)	4	100
new.GH_25_5.4	4	(0:0:719)	4	20	(0:0:349)	4	100	(0:0:308)	4	100	(0:0:530)	4	100
new.GH_25_5.5	4	(0:0:765)	4	0	(0:0:867)	4	100	(0:0:307)	4	100	(0:7:310)	4	90
new.GH_25_5.6	4	(0:0:681)	4	50	(0:0:208)	4	100	(0:0:146)	4	100	(0:0:494)	4	100
new.GH_25_5.7	4	(0:0:772)	5	0	(0:12:524)	4	40	(0:1:515)	4	100	(0:8:422)	4	90
new.GH_25_5.8	4	(0:0:671)	4	30	(0:0:702)	4	100	(0:0:688)	4	100	(0:0:728)	4	100
new.GH_25_5.9	4	(0:0:709)	4	10	(0:4:989)	4	70	(0:3:638)	4	80	(0:1:646)	4	100
new.GH_25_5.10	4	(0:0:760)	5	0	(0:0:382)	4	100	(0:1:803)	4	100	(0:0:770)	4	100
new.GH_30_8.1	4	(0:1:173)	5	0	(0:11:7)	4	70	(0:4:22)	4	80	(0:11:767)	4	100
new.GH_30_8.2	4	(0:1:188)	4	10	(0:3:239)	4	80	(0:0:662)	4	100	(0:2:345)	4	100
new.GH_30_8.3	4	(0:1:769)	5	0	(0:6:999)	4	90	(0:8:57)	4	50	(0:32:604)	4	100
new.GH_30_8.4	4	(0:1:800)	4	40	(0:6:808)	4	80	(0:8:255)	4	40	(0:2:468)	4	100
new.GH_30_8.5	4	(0:1:140)	4	40	(0:1:824)	4	100	(0:3:797)	4	80	(0:2:995)	4	100
new.GH_30_8.6	4	(0:1:328)	4	20	(0:0:426)	4	100	(0:0:455)	4	100	(0:1:160)	4	100
new.GH_30_8.7	4	(0:1:1)	4	30	(0:0:305)	4	100	(0:0:597)	4	100	(0:1:187)	4	100
new.GH_30_8.8	4	(0:1:301)	-	0	(0:6:45)	4	80	(0:3:908)	4	80	(0:14:738)	4	100
new.GH_30_8.9	4	(0:1:257)	-	0	(0:0:715)	4	100	(0:0:926)	4	100	(0:0:946)	4	100
new.GH_30_8.10	4	(0:0:943)	4	40	(0:6:761)	4	90	(0:0:680)	4	100	(0:1:66)	4	100
new.GH_50_2.1	6	(0:2:880)	7	0	(1:0:971)	6	80	(0:42:604)	6	60	(2:1:696)	6	60
new.GH_50_2.2	6	(0:2:656)	7	0	(1:16:174)	6	60	(0:33:537)	6	70	(0:30:210)	6	100
new.GH_50_2.3	6	(0:2:548)	7	0	(0:4:492)	6	100	(0:6:182)	6	100	(0:16:110)	6	100
new.GH_50_2.4	6	(0:2:524)	-	0	(0:4:846)	6	100	(0:6:664)	6	100	(0:13:614)	6	100
new.GH_50_2.5	6	(0:2:540)	7	0	(0:20:998)	6	90	(0:20:488)	6	80	(0:13:91)	6	100
new.GH_50_2.6	6	(0:2:559)	-	0	(0:3:670)	6	100	(0:5:334)	6	100	(0:9:276)	6	100
new.GH_50_2.7	6	(0:2:540)	-	0	(0:7:345)	6	100	(0:29:387)	6	80	(0:18:60)	6	100
new.GH_50_2.8	6	(0:2:578)	7	0	(0:3:306)	6	100	(0:13:974)	6	90	(0:13:835)	6	100
new.GH_50_2.9	6	(0:2:550)	-	0	(0:5:653)	6	100	(0:6:435)	6	100	(0:26:437)	6	100
new.GH_50_2.10	6	(0:2:415)	7	0	(0:4:357)	6	100	(0:26:16)	6	80	(0:15:639)	6	100
new.GH_50_3.1	5	(0:3:214)	6	0	(0:15:903)	5	100	(0:44:123)	5	80	(1:7:937)	5	90
new.GH_50_3.2	5	(0:3:596)	-	0	(0:49:766)	5	80	(0:51:336)	5	40	(0:15:792)	5	100
new.GH_50_3.3	5	(0:0:282)	5	100	(0:21:16)	5	90	(0:15:504)	5	90	(0:12:158)	5	100
new.GH_50_3.4	5	(0:3:866)	6	0	(2:47:763)	6	60	(1:28:655)	6	40	(5:56:354)	6	100
new.GH_50_3.5	5	(0:4:580)	-	0	(0:55:737)	5	70	(0:48:610)	5	50	(1:10:320)	5	100
new.GH_50_3.6	5	(0:0:616)	5	100	(0:31:282)	5	90	(0:44:834)	5	40	(0:26:442)	5	100

Tabela 19 – Resultados das instâncias new.GH da classe C2 com B = 622 MB/s. (continuação)

Instâncias		BR			BR + QL			BRKGA + VB			BRKGA + VB(QL)		
Nome	*	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)
new.GH_50_3.7	5	(0:3:466)	-	0	(0:20:875)	5	100	(0:42:543)	5	80	(1:40:471)	5	70
new.GH_50_3.8	5	(0:3:423)	-	0	(0:8:121)	5	100	(0:7:944)	5	100	(1:43:251)	5	80
new.GH_50_3.9	5	(0:3:423)	-	0	(0:4:137)	5	100	(0:8:243)	5	100	(0:13:713)	5	100
new.GH_50_3.10	6	(0:3:365)	7	0	(1:33:157)	6	40	(0:38:465)	6	50	(0:8:440)	6	100
new.GH_50_8.1	6	(0:4:714)	7	0	(1:1:173)	6	70	(0:42:970)	6	50	(0:24:264)	6	100
new.GH_50_8.2	6	(0:4:52)	-	0	(0:21:194)	6	90	(0:30:112)	6	100	(0:12:520)	6	100
new.GH_50_8.3	6	(0:4:36)	-	0	(0:20:168)	6	90	(0:7:366)	6	100	(0:14:518)	6	100
new.GH_50_8.4	6	(0:4:718)	-	0	(0:20:148)	6	90	(0:7:662)	6	100	(0:10:268)	6	100
new.GH_50_8.5	6	(0:3:845)	-	0	(0:37:631)	6	80	(0:53:299)	6	60	(0:12:505)	6	100
new.GH_50_8.6	6	(0:4:434)	7	0	(0:21:573)	6	90	(0:26:377)	6	80	(0:8:500)	6	100
new.GH_50_8.7	6	(0:3:786)	6	10	(0:47:436)	6	80	(0:54:498)	6	30	(1:11:652)	6	10
new.GH_50_8.8	6	(0:3:332)	6	20	(1:9:784)	6	60	(0:46:481)	6	50	(0:31:464)	6	100
new.GH_50_8.9	6	(0:3:243)	-	0	(1:15:722)	6	70	(1:6:957)	6	20	(1:8:628)	6	90
new.GH_50_8.10	6	(0:3:321)	7	0	(1:11:115)	6	60	(0:49:565)	6	40	(0:18:583)	6	100
new.GH_50_9.1	5	(0:3:440)	6	0	(0:4:968)	5	100	(0:6:866)	5	100	(0:16:495)	5	100
new.GH_50_9.2	5	(0:3:703)	-	0	(0:58:819)	5	70	(0:20:901)	5	80	(0:28:918)	5	100
new.GH_50_9.3	5	(0:3:549)	6	0	(0:16:479)	5	90	(0:11:971)	5	100	(0:46:958)	5	90
new.GH_50_9.4	5	(0:3:543)	5	10	(2:4:525)	5	10	(0:47:851)	5	50	(2:54:8)	5	50
new.GH_50_9.5	5	(0:3:541)	-	0	(1:22:530)	5	50	(0:52:273)	5	40	(0:52:225)	5	90
new.GH_50_9.6	5	(0:3:647)	6	0	(0:38:728)	5	80	(0:29:566)	5	70	(2:16:841)	5	60
new.GH_50_9.7	5	(0:3:576)	6	0	(2:4:69)	5	30	(1:3:127)	5	30	(2:36:174)	5	70
new.GH_50_9.8	5	(0:0:926)	5	90	(0:45:924)	5	80	(0:56:113)	5	30	(1:40:982)	5	70
new.GH_50_9.9	5	(0:3:813)	-	0	(0:5:895)	5	100	(0:15:13)	5	90	(0:9:16)	5	100
new.GH_50_9.10	5	(0:0:274)	5	100	(0:18:635)	5	90	(0:15:140)	5	90	(0:25:666)	5	100
new.GH_50_10.1	5	(0:3:93)	6	0	(0:58:298)	5	70	(0:20:239)	5	90	(2:22:571)	5	50
new.GH_50_10.2	5	(0:3:161)	5	10	(2:42:947)	0	0	(1:16:905)	5	10	(2:4:536)	5	60
new.GH_50_10.3	5	(0:3:92)	5	30	(1:28:992)	5	40	(0:21:962)	5	80	(1:53:632)	5	90
new.GH_50_10.4	5	(0:3:37)	-	0	(0:46:767)	5	70	(0:29:673)	5	60	(1:54:777)	5	80
new.GH_50_10.5	5	(0:3:220)	6	0	(2:5:783)	5	20	(0:42:607)	5	50	(3:31:784)	5	30
new.GH_50_10.6	5	(0:3:43)	-	0	(0:40:713)	5	80	(0:46:616)	5	40	(3:51:917)	5	50
new.GH_50_10.7	5	(0:3:123)	6	0	(2:12:462)	5	10	(1:16:608)	5	30	(4:34:747)	5	30
new.GH_50_10.8	5	(0:3:52)	5	10	(1:51:465)	5	30	(1:10:573)	5	30	(2:19:821)	5	60
new.GH_50_10.9	5	(0:3:259)	6	0	(2:21:604)	5	10	(0:48:126)	5	30	(2:22:219)	5	60
new.GH_50_10.10	5	(0:3:205)	5	30	(1:0:852)	5	60	(0:27:222)	5	80	(1:15:760)	5	80
Médias		**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**
		(0:2:521)	**34**	**13,43**	**(0:38:16)**	**99**	**77,57**	**(0:25:622)**	**100**	**73,57**	**(0:53:398)**	**100**	**88,57**

Tabela 20 – Resultados das instâncias new.RL da classe C2 com B = 155 MB/s.

Instâncias		BR			BR + QL			BR + VB			BR + VB(QL)		
Nome	*	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)
new.RL_15_2.1	3	(0:0:110)	3	90	(0:0:49)	3	100	(0:0:187)	3	100	(0:1:67)	3	100
new.RL_15_2.2	4	(0:0:155)	4	100	(0:0:30)	4	100	(0:0:37)	4	100	(0:0:191)	4	100
new.RL_15_2.3	3	(0:0:167)	3	70	(0:0:49)	3	100	(0:0:514)	3	90	(0:10:397)	3	60
new.RL_15_2.4	3	(0:0:75)	3	100	(0:0:36)	3	100	(0:0:43)	3	100	(0:0:585)	3	100
new.RL_15_2.5	3	(0:0:181)	3	80	(0:0:58)	3	100	(0:1:227)	3	60	(0:7:375)	3	70
new.RL_15_2.6	3	(0:0:149)	3	80	(0:0:85)	3	100	(0:0:117)	3	100	(0:1:275)	3	100
new.RL_15_2.7	3	(0:0:272)	4	0	(0:3:168)	4	100	(0:2:524)	4	100	(0:26:201)	4	100
new.RL_15_2.8	3	(0:0:33)	3	100	(0:0:35)	3	100	(0:0:69)	3	100	(0:0:191)	3	100
new.RL_15_2.9	4	(0:0:233)	4	80	(0:0:296)	4	100	(0:0:248)	4	100	(0:0:317)	4	100
new.RL_15_2.10	3	(0:0:172)	3	70	(0:0:62)	3	100	(0:0:311)	3	100	(0:0:364)	3	100
new.RL_15_5.1	3	(0:0:66)	3	100	(0:0:36)	3	100	(0:0:71)	3	100	(0:0:162)	3	100
new.RL_15_5.2	3	(0:0:66)	3	90	(0:0:164)	3	100	(0:0:110)	3	100	(0:0:189)	3	100
new.RL_15_5.3	3	(0:0:47)	3	100	(0:0:27)	3	100	(0:0:91)	3	100	(0:0:178)	3	100
new.RL_15_5.4	3	(0:0:106)	3	80	(0:0:128)	3	100	(0:0:67)	3	100	(0:0:237)	3	100
new.RL_15_5.5	3	(0:0:150)	3	90	(0:0:422)	3	90	(0:0:498)	3	80	(0:2:618)	3	90
new.RL_15_5.6	3	(0:0:50)	3	100	(0:0:38)	3	100	(0:0:363)	3	80	(0:0:348)	3	100
new.RL_15_5.7	3	(0:0:71)	3	100	(0:0:39)	3	100	(0:0:65)	3	80	(0:0:169)	3	100
new.RL_15_5.8	3	(0:0:186)	3	80	(0:0:80)	3	100	(0:0:67)	3	100	(0:0:198)	3	100
new.RL_15_5.9	3	(0:0:70)	3	100	(0:0:36)	3	100	(0:0:58)	3	100	(0:0:164)	3	100
new.RL_15_5.10	3	(0:0:56)	3	100	(0:0:39)	3	100	(0:0:229)	3	90	(0:0:236)	3	100
new.RL_25_3.1	4	(0:0:668)	4	20	(0:0:250)	4	100	(0:1:348)	4	100	(0:0:529)	4	100
new.RL_25_3.2	4	(0:0:691)	4	10	(0:1:434)	4	90	(0:1:504)	4	90	(0:0:554)	4	100
new.RL_25_3.3	4	(0:0:673)	4	20	(0:5:441)	4	70	(0:2:559)	4	80	(0:11:630)	4	80
new.RL_25_3.4	4	(0:0:656)	4	10	(0:1:417)	4	90	(0:0:242)	4	100	(0:0:477)	4	100
new.RL_25_3.5	4	(0:0:628)	4	20	(0:0:210)	4	100	(0:0:229)	4	100	(0:0:439)	4	100
new.RL_25_3.6	4	(0:0:729)	-	0	(0:2:932)	4	80	(0:4:430)	4	50	(0:11:306)	4	80
new.RL_25_3.7	4	(0:0:713)	-	0	(0:0:471)	4	100	(0:0:961)	4	100	(0:1:105)	4	100
new.RL_25_3.8	4	(0:0:662)	4	40	(0:0:217)	4	100	(0:0:398)	4	100	(0:0:436)	4	100
new.RL_25_3.9	4	(0:0:630)	4	20	(0:0:200)	4	100	(0:1:35)	4	100	(0:0:363)	4	100
new.RL_25_3.10	4	(0:0:681)	4	30	(0:0:348)	4	100	(0:0:731)	4	100	(0:0:784)	4	100
new.RL_30_2.1	4	(0:1:34)	4	10	(0:0:562)	4	100	(0:2:416)	4	90	(0:1:614)	4	100
new.RL_30_2.2	4	(0:0:990)	-	0	(0:7:169)	4	80	(0:8:370)	4	20	(0:16:156)	4	80
new.RL_30_2.3	4	(0:0:999)	5	0	(0:14:224)	4	30	(0:9:100)	4	30	(0:45:180)	4	40
new.RL_30_2.4	4	(0:1:98)	4	10	(0:7:666)	4	80	(0:5:49)	4	70	(0:10:34)	4	90
new.RL_30_2.5	4	(0:1:41)	-	0	(0:3:646)	4	90	(0:3:595)	4	80	(0:10:478)	4	90
new.RL_30_2.6	4	(0:0:980)	4	10	(0:0:506)	4	100	(0:0:662)	4	100	(0:1:314)	4	100

Tabela 21 – Resultados das instâncias new.RL da classe C2 com B = 155 MB/s (continuação)

Instâncias		BR			BR + QL			BR+VB			BR+VB(QL)		
Nome	*	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)
new.RL_30_2.7	4	(0:0:982)	5	0	(0:12:212)	4	50	(0:8:944)	4	80	(0:11:574)	4	90
new.RL_30_2.8	4	(0:1:3)	-	0	(0:4:86)	4	90	(0:5:938)	4	40	(0:36:482)	4	50
new.RL_30_2.9	4	(0:0:944)	4	10	(0:7:75)	4	70	(0:2:310)	4	100	(0:2:898)	4	100
new.RL_30_2.10	4	(0:0:923)	4	0	(0:0:729)	4	100	(0:3:707)	4	80	(0:30:433)	4	60
new.RL_30_9.1	4	(0:1:23)	4	20	(0:6:299)	4	80	(0:3:63)	4	80	(0:2:150)	4	100
new.RL_30_9.2	4	(0:1:12)	5	0	(0:3:642)	4	100	(0:4:115)	4	70	(0:4:113)	4	100
new.RL_30_9.3	4	(0:1:18)	-	0	(0:13:785)	4	40	(0:4:102)	4	80	(0:10:92)	4	90
new.RL_30_9.4	4	(0:1:17)	4	10	(0:14:446)	4	30	(0:8:223)	4	20	(0:42:598)	4	50
new.RL_30_9.5	4	(0:1:70)	4	10	(0:2:469)	4	90	(0:4:772)	4	60	(0:1:402)	4	100
new.RL_30_9.6	4	(0:1:102)	4	10	(0:15:884)	4	20	(0:8:760)	4	20	(0:12:596)	4	90
new.RL_30_9.7	4	(0:1:5)	5	0	(0:8:314)	4	60	(0:4:790)	4	60	(0:3:876)	4	100
new.RL_30_9.8	4	(0:1:10)	-	0	(0:11:74)	4	60	(0:5:124)	4	70	(0:9:850)	4	90
new.RL_30_9.9	4	(0:1:25)	5	0	(0:10:758)	4	50	(0:5:795)	4	50	(0:1:643)	4	100
new.RL_30_9.10	4	(0:1:30)	-	0	(0:11:182)	4	50	(0:11:169)	4	10	(0:38:635)	4	50
new.RL_50_2.1	5	(0:2:929)	-	0	(1:44:257)	5	30	(1:23:692)	5	0	(1:18:327)	5	80
new.RL_50_2.2	6	(0:2:965)	7	0	(1:39:35)	6	30	(0:51:179)	6	20	(2:5:646)	6	50
new.RL_50_2.3	5	(0:2:866)	5	20	(2:26:429)	5	0	(1:34:253)	5	10	(2:18:948)	5	60
new.RL_50_2.4	5	(0:2:891)	6	0	(2:18:582)	5	0	(1:51:308)	5	10	(3:15:647)	5	30
new.RL_50_2.5	5	(0:2:946)	-	0	(2:4:693)	5	30	(1:30:762)	5	20	(3:2:269)	5	50
new.RL_50_2.6	5	(0:2:992)	6	0	(2:5:205)	5	0	(1:24:960)	5	10	(3:2:416)	5	40
new.RL_50_2.7	5	(0:3:527)	-	0	(2:3:482)	5	0	(1:15:960)	5	20	(1:33:330)	5	80
new.RL_50_2.8	5	(0:3:147)	-	0	(2:31:706)	5	0	(1:17:982)	5	20	(2:16:943)	5	60
new.RL_50_2.9	5	(0:3:50)	-	0	(2:15:333)	5	0	(1:13:323)	5	10	(4:29:290)	5	10
new.RL_50_2.10	5	(0:3:547)	6	0	(1:11:154)	5	30	(1:28:828)	5	10	(3:10:343)	5	40
new.RL_50_8.1	5	(0:3:70)	-	0	(1:18:481)	5	20	(1:14:360)	5	30	(2:15:989)	5	70
new.RL_50_8.2	6	(0:3:111)	7	0	(1:9:106)	6	0	(1:8:926)	6	20	(1:20:246)	6	90
new.RL_50_8.3	5	(0:3:180)	-	0	(1:26:350)	-	0	(1:18:947)	5	20	(2:1:262)	5	70
new.RL_50_8.4	5	(0:3:316)	-	0	(1:28:343)	-	0	(1:30:412)	5	10	(2:48:25)	5	50
new.RL_50_8.5	6	(0:3:537)	-	0	(1:16:254)	-	0	(1:4:288)	6	40	(1:32:748)	6	80
new.RL_50_8.6	5	(0:3:101)	-	0	(1:23:480)	5	10	(1:37:532)	6	10	(2:38:14)	5	100
new.RL_50_8.7	5	(0:2:758)	6	0	(1:13:350)	-	0	(1:36:562)	-	0	(3:12:74)	5	50
new.RL_50_8.8	5	(0:2:756)	-	0	(1:25:579)	-	0	(1:30:55)	5	20	(5:13:634)	5	10
new.RL_50_8.9	5	(0:2:760)	-	0	(1:22:887)	5	0	(1:22:941)	5	20	(2:31:303)	5	70
new.RL_50_8.10	5	(0:2:851)	6	0	(0:49:274)	5	70	(1:26:954)	5	10	(2:54:397)	5	50
Médias		**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**
		(0:1:296)	**51,43**	**28,43**	**(0:30:950)**	**81,43**	**64,43**	**(0:25:622)**	**97,14**	**61,71**	**(0:50:772)**	**100**	**81,29**

Tabela 22 – Resultados das instâncias new.RH da classe C2 com B = 622 MB/s

Instâncias		BR			BR + QL			BR + VB			BR+VB(QL)		
Nome	*	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)	T. Médio	(*)	C(%)
new.RH_15_10.1	3	(0:0:70)	3	100	(0:0:44)	3	100	(0:0:61)	3	100	(0:0:387)	3	100
new.RH_15_10.2	3	(0:0:60)	3	100	(0:0:50)	3	100	(0:0:67)	3	100	(0:0:274)	3	100
new.RH_15_10.3	3	(0:0:165)	3	50	(0:0:299)	3	100	(0:0:698)	3	80	(0:0:820)	3	100
new.RH_15_10.4	3	(0:0:157)	3	60	(0:1:186)	3	30	(0:1:377)	3	30	(0:14:969)	3	50
new.RH_15_10.5	3	(0:0:165)	3	40	(0:1:205)	3	30	(0:0:941)	3	50	(0:16:872)	3	40
new.RH_15_10.6	3	(0:0:121)	3	100	(0:0:64)	3	100	(0:0:283)	3	100	(0:2:155)	3	100
new.RH_15_10.7	3	(0:0:174)	3	60	(0:0:522)	3	80	(0:0:491)	3	80	(0:0:228)	3	100
new.RH_15_10.8	3	(0:0:236)	3	70	(0:0:743)	3	60	(0:1:68)	3	40	(0:13:548)	3	50
new.RH_15_10.9	3	(0:0:136)	3	80	(0:0:729)	3	50	(0:0:740)	3	60	(0:8:116)	3	70
new.RH_15_10.10	3	(0:0:52)	3	100	(0:0:66)	3	100	(0:0:41)	3	100	(0:0:214)	3	100
new.RH_30_5.1	4	(0:1:301)	-	0	(0:0:943)	4	100	(0:2:180)	4	90	(0:1:371)	4	100
new.RH_30_5.2	4	(0:1:360)	5	0	(0:2:585)	4	90	(0:2:57)	4	100	(0:1:502)	4	100
new.RH_30_5.3	4	(0:1:188)	0	0	(0:0:549)	4	100	(0:0:634)	4	100	(0:0:965)	4	100
new.RH_30_5.4	4	(0:1:137)	4	10	(0:0:842)	4	100	(0:0:698)	4	100	(0:1:179)	4	100
new.RH_30_5.5	4	(0:1:112)	-	0	(0:13:328)	4	20	(0:13:484)	4	10	(1:25:34)	4	10
new.RH_30_5.6	4	(0:1:138)	-	0	(0:6:283)	4	50	(0:6:665)	4	70	(0:2:565)	4	100
new.RH_30_5.7	4	(0:1:407)	4	10	(0:9:298)	4	50	(0:11:798)	4	30	(0:30:640)	4	70
new.RH_30_5.8	4	(0:1:483)	4	20	(0:5:751)	4	70	(0:8:506)	4	60	(0:19:550)	4	80
new.RH_30_5.9	4	(0:1:458)	-	0	(0:11:704)	4	20	(0:11:768)	4	20	(0:37:139)	4	60
new.RH_30_5.10	4	(0:1:360)	-	0	(0:13:969)	4	10	(0:2:57)	4	100	(0:27:861)	4	70
Médias		**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**	**T. Médio**	***(%)**	**C(%)**
		(0:0:714)	**65**	**40**	**(0:3:508)**	**100**	**68**	**(0:25:558)**	**100**	**71**	**(0:13:269)**	**100**	**80**

Tabela 23 – Resultados das instâncias da classe C4 com 100 nós

Instâncias		BR			BR+QL			BR+VB			BR+VB(QL)		
Nome	klb	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)
RWNR_100_01	5	(0:28:504)	6	50	(1:39:943)	6	10	(2:25:110)	6	90	(2:10:362)	6	100
RWNR_100_02	5	(0:42:915)	6	30	(1:26:349)	6	10	(2:27:947)	6	80	(3:17:109)	6	90
RWNR_100_03	5	(0:33:838)	6	80	(1:21:718)	6	80	(1:22:182)	6	100	(1:8:596)	6	100
RWNR_100_04	5	(0:21:318)	-	0	(1:18:238)	6	100	(1:19:754)	6	100	(1:1:944)	6	100
RWNR_100_05	5	(0:23:757)	6	60	(1:12:88)	6	100	(1:20:177)	6	100	(1:2:217)	6	100
RWNR_100_06	5	(0:22:298)	6	80	(1:21:123)	6	90	(1:28:396)	6	100	(1:7:205)	6	100
RWNR_100_07	5	(0:37:165)	9	30	(1:22:329)	7	20	(1:56:554)	6	100	(1:39:113)	6	100
RWNR_100_08	5	(0:14:200)	6	80	(1:20:652)	6	100	(1:18:144)	6	100	(0:59:577)	6	100
RWNR_100_09	5	(0:14:34)	6	80	(1:23:83)	6	50	(1:23:701)	6	100	(0:59:56)	6	100
RWNR_100_10	5	(0:17:39)	6	90	(1:17:255)	6	100	(1:44:346)	6	100	(0:59:431)	6	100
RWNR_100_11	3	(0:7:964)	4	70	(1:3:954)	4	100	(0:54:410)	4	100	(0:44:3)	4	100
RWNR_100_12	4	(0:43:212)	5	20	(1:16:31)	5	40	(3:42:515)	5	90	(3:58:438)	5	100
RWNR_100_13	4	(0:50:787)	5	60	(1:5:795)	5	100	(3:14:662)	5	80	(2:43:238)	5	100
RWNR_100_14	4	(0:40:709)	5	30	(1:17:689)	5	30	(3:33:776)	5	70	(1:55:399)	5	100
RWNR_100_15	4	(1:2:150)	5	70	(1:13:37)	5	70	(3:10:733)	5	90	(2:34:527)	5	100
RWNR_100_16	4	(0:23:148)	5	60	(1:17:576)	5	70	(2:20:778)	5	100	(1:10:994)	5	100
RWNR_100_17	3	(0:18:441)	4	80	(1:6:956)	4	100	(1:17:830)	4	100	(0:55:29)	4	100
RWNR_100_18	3	(0:25:918)	4	100	(0:57:807)	4	100	(1:30:199)	4	100	(0:45:649)	4	100
RWNR_100_19	3	(0:24:154)	4	80	(1:0:518)	4	100	(1:27:752)	4	100	(1:6:569)	4	100
RWNR_100_20	3	(0:25:882)	-	0	(1:2:656)	4	100	(1:48:929)	4	100	(1:10:961)	4	100
RWNR_100_21	4	(1:0:786)	5	60	(1:5:125)	5	100	(2:28:189)	5	100	(1:27:181)	5	100
RWNR_100_22	4	(0:38:106)	-	0	(1:4:100)	5	100	(2:22:410)	5	100	(1:33:620)	5	100
RWNR_100_23	5	(1:51:801)	-	0	(1:7:970)	6	100	(5:51:638)	6	10	(5:48:11)	6	100
RWNR_100_24	5	(1:52:222)	-	0	(1:3:653)	6	100	(5:56:977)	7	40	(3:59:914)	6	90
RWNR_100_25	5	(1:30:385)	7	80	(1:11:637)	6	100	(4:0:41)	6	100	(2:23:907)	6	100
RWNR_100_26	5	(1:47:705)	-	0	(1:9:144)	6	100	(5:46:254)	7	70	(3:32:186)	6	90
RWNR_100_27	3	(1:57:816)	4	20	(1:18:335)	4	100	(4:52:980)	4	20	(4:41:356)	4	90
RWNR_100_28	3	(0:20:169)	4	100	(1:2:503)	4	100	(1:2:455)	4	100	(0:35:374)	4	100
RWNR_100_29	3	(1:42:161)	4	100	(1:0:734)	4	100	(2:47:399)	4	90	(2:6:582)	4	100
RWNR_100_30	3	(0:51:29)	4	80	(0:56:58)	4	100	(1:17:487)	4	100	(0:43:867)	4	100
RWNR_100_31	3	(0:22:246)	4	80	(1:2:611)	4	100	(1:9:946)	4	100	(0:54:517)	4	100
RWNR_100_32	4	(0:20:375)	5	90	(1:6:495)	5	100	(1:18:420)	5	100	(0:51:129)	5	100
RWNR_100_33	4	(1:45:716)	-	0	(1:21:36)	-	0	(4:40:370)	5	20	(2:21:717)	5	100
RWNR_100_34	4	(0:39:131)	5	90	(1:3:116)	5	100	(2:6:342)	5	90	(2:2:945)	5	100
RWNR_100_35	4	(1:44:570)	-	0	(1:8:238)	5	100	(4:21:844)	5	50	(4:24:321)	5	90

Tabela 24 – Resultados das instâncias da classe C4 com 100 nós (continuação)

Instâncias		BR			BR+QL			BR+VB			BR+VB(QL)		
Nome	klb	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)
RWNR_100_36	4	(0:39:365)	-	0	(1:21:526)	-	0	(4:33:673)	0	0	(6:0:745)	5	60
RWNR_100_37	4	(0:20:848)	5	50	(1:24:872)	-	0	(1:24:593)	5	100	(0:52:864)	5	100
RWNR_100_38	4	(0:22:512)	5	70	(1:25:933)	-	0	(1:28:974)	5	90	(2:1:698)	5	100
RWNR_100_39	4	(0:24:873)	5	80	(1:3:32)	5	100	(1:17:123)	5	70	(0:49:138)	5	100
RWNR_100_40	4	(0:33:417)	5	40	(1:3:731)	5	100	(2:2:741)	5	100	(1:13:728)	5	100
RWNR_100_41	4	(0:27:874)	5	90	(1:14:729)	5	100	(1:13:254)	5	100	(0:53:13)	5	100
RWNR_100_42	4	(0:20:877)	5	80	(1:4:261)	5	100	(1:14:451)	5	100	(0:50:876)	5	100
RWNR_100_43	4	(1:21:22)	5	40	(1:4:263)	5	100	(2:54:317)	5	80	(1:26:106)	5	100
RWNR_100_44	4	(0:25:983)	5	80	(1:9:430)	5	100	(1:29:736)	5	100	(0:49:652)	5	100
RWNR_100_45	4	(0:31:960)	5	90	(1:15:169)	5	100	(1:16:362)	5	100	(0:52:680)	5	100
RWNR_100_46	4	(1:37:821)	5	30	(1:5:178)	5	100	(3:55:299)	5	10	(4:42:355)	5	70
Médias		T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)
		(0:45:831)	80,4	54,3	(1:11:689)	91,3	79,78	(2:26:982)	97,8	83,5	(1:56:715)	100	97,39

Tabela 25 – Resultados das instâncias da classe C4 com 150 nós

Instâncias		BR			BR+QL			BR+VB			BR+VB(QL)		
Nome	klb	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)
RWNR_150_01	5	(1:54:671)	-	0	(4:37:49)	-	0	(23:55:463)	-	0	(36:21:167)	7	20
RWNR_150_02	5	(1:32:44)	-	0	(4:43:498)	-	0	(19:5:799)	3	30	(13:19:758)	6	80
RWNR_150_03	5	(1:31:279)	-	0	(5:8:380)	-	0	(21:25:299)	-	0	(24:10:749)	6	80
RWNR_150_04	5	(1:28:581)	-	0	(4:16:296)	-	0	(17:53:800)	2	20	(29:47:513)	8	10
RWNR_150_05	5	(1:32:481)	-	0	(4:44:77)	6	10	(17:24:978)	2	20	(24:56:558)	6	70
RWNR_150_06	6	(1:20:583)	-	0	(4:37:708)	-	0	(17:54:985)	7	40	(12:28:548)	7	90
RWNR_150_07	6	(1:34:548)	-	0	(4:49:398)	-	0	(18:9:391)	7	30	(13:47:26)	7	100
RWNR_150_08	6	(1:25:956)	-	0	(4:27:753)	-	0	(18:14:587)	7	10	(23:47:495)	8	10
RWNR_150_09	6	(1:23:147)	7	20	(4:35:817)	7	40	(12:34:198)	-	0	(23:42:409)	7	100
RWNR_150_10	6	(1:23:832)	7	30	(4:59:72)	-	0	(11:2:406)	8	80	(20:54:944)	7	50
Médias		T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)
		(1:30:712)	20,00	5,00	(4:41:905)	20,00	5,00	(17:46:91)	70,00	23,00	(22:19:617)	100	61,00

Tabela 26 – Resultados das instâncias da classe C4 com 200 nós

Instâncias		BR			BR+QL			BR+VB			BR+VB(QL)		
Nome	klb	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)
RWNR_200_01	5	(9:28:407)	7	60	(14:17:20)	6	90	(58:26:639)	-	0	(43:42:390)	6	90
RWNR_200_02	6	(8:13:804)	7	30	(13:27:480)	7	20	(38:19:950)	7	20	(31:57:715)	7	90
RWNR_200_03	6	(6:35:616)	7	30	(6:31:299)	7	90	(16:6:725)	7	100	(28:7:178)	7	70
RWNR_200_04	6	(5:24:208)	7	90	(11:15:550)	7	90	(14:9:519)	7	100	(27:44:307)	7	100
RWNR_200_05	5	(6:4:213)	6	40	(12:52:198)	6	60	(14:4:209)	6	100	(36:31:613)	6	100
RWNR_200_06	4	(7:35:397)	-	0	(12:23:534)	-	0	(37:4:758)	-	0	(61:5:422)	8	10
RWNR_200_07	6	(4:49:819)	-	0	(11:27:881)	7	100	(19:2:974)	7	100	(31:59:645)	7	100
RWNR_200_08	7	(6:54:258)	8	20	(13:39:656)	8	20	(23:34:897)	8	90	(56:5:879)	8	20
RWNR_200_09	6	(7:20:631)	-	0	(14:29:37)	7	20	(38:48:818)	7	10	(45:44:579)	7	20
RWNR_200_10	5	(7:29:378)	-	0	(13:38:238)	8	10	(31:6:212)	6	40	(43:59:731)	8	20
RWNR_200_11	4	(0:3:156)	4	80	(7:44:565)	4	100	(0:4:957)	4	100	(68:15:959)	5	100
RWNR_200_12	4	(0:3:168)	4	70	(7:18:92)	4	100	(0:4:759)	4	100	(70:58:762)	5	100
RWNR_200_13	5	(0:1:759)	6	70	(6:46:316)	6	100	(0:2:957)	6	90	(17:41:745)	6	100
RWNR_200_14	5	(0:1:896)	6	60	(7:15:827)	6	100	(0:3:375)	6	100	(0:3:435)	6	100
RWNR_200_15	5	(0:2:642)	6	100	(8:2:240)	6	100	(0:4:338)	6	100	(0:4:344)	6	100
RWNR_200_16	5	(0:2:607)	6	100	(7:20:594)	6	100	(0:4:782)	6	100	(0:4:782)	6	100
RWNR_200_17	4	(0:2:746)	5	100	(7:1:846)	5	100	(0:29:322)	5	90	(0:5:158)	5	100
RWNR_200_18	4	(0:52:400)	4	100	(7:25:711)	4	100	(0:12:156)	4	100	(0:9:986)	4	100
RWNR_200_19	4	(0:2:433)	5	90	(6:22:462)	5	100	(0:3:703)	5	100	(0:4:514)	5	100
RWNR_200_20	4	(0:2:772)	5	70	(13:43:901)	5	100	(0:4:16)	5	100	(0:5:161)	5	100
RWNR_200_21	4	(0:2:503)	5	80	(7:17:885)	5	100	(0:3:578)	5	100	(0:4:557)	5	100
RWNR_200_22	4	(6:55:897)	-	0	(13:6:235)	-	0	(29:38:407)	-	0	(74:52:528)	-	0
RWNR_200_23	4	(7:31:643)	5	30	(12:56:637)	-	0	(31:13:98)	5	100	(23:15:202)	5	90
RWNR_200_24	4	(7:50:710)	-	0	(12:57:199)	7	10	(37:5:35)	6	20	(46:20:461)	8	10
RWNR_200_25	4	(2:19:706)	5	30	(6:21:804)	5	100	(7:46:833)	5	100	(6:59:809)	5	100
RWNR_200_26	4	(0:16:223)	5	90	(6:16:161)	5	100	(60:2:607)	5	100	(0:5:234)	5	100
RWNR_200_27	6	(1:13:864)	7	70	(6:20:882)	7	100	(43:49:320)	7	60	(4:3:55)	7	100
RWNR_200_28	6	(1:18:251)	7	50	(6:25:604)	7	100	(44:13:96)	7	70	(12:30:158)	7	80
RWNR_200_29	6	(4:59:962)	7	40	(10:5:713)	7	40	(33:57:1)	7	70	(20:33:934)	7	100
RWNR_200_30	6	(4:45:890)	7	20	(10:2:330)	7	70	(22:23:695)	-	0	(15:34:628)	7	100
RWNR_200_31	6	(3:47:727)	-	0	(9:18:936)	7	100	(10:57:36)	7	100	(14:27:465)	7	100
Médias		T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)
		(3:37:216)	77,4	49,03	(9:48:801)	90,32	71,61	(19:46:735)	87,10	72,90	(25:16:108)	96,77	80,65

Tabela 27 – Resultados das instâncias da classe C4 com 250 nós

Instâncias		BR			BR+QL			BR+VB			BR+VB(QL)		
Nome	klb	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)
RWNR_250_01	4	(0:4:961)	-	0	(8:19:617)	5	100	(0:8:839)	5	100	(0:6:652)	5	100
RWNR_250_02	4	(6:42:54)	5	50	(9:15:638)	5	100	(3:34:696)	5	100	(14:14:951)	5	100
RWNR_250_03	4	(9:22:665)	-	0	(11:29:684)	5	80	(18:51:549)	5	100	(13:26:61)	5	100
RWNR_250_04	5	(15:42:439)	-	0	(20:3:355)	6	90	(120:19:31)	6	70	(38:2:295)	6	100
RWNR_250_05	4	(6:14:745)	-	0	(8:35:997)	5	100	(5:11:535)	5	70	(12:44:23)	5	100
RWNR_250_06	4	(13:54:430)	-	0	(18:55:991)	5	40	(28:44:143)	5	90	(26:26:364)	5	80
Médias		T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)
		(8:40:216)	16,67	8,33	(12:46:714)	100	85	(29:28:299)	100	88,3	(17:30:58)	100	96,67

Tabela 28 – Resultados das instâncias da classe C4 com 300 nós

Instâncias		BR			BR+QL			BR+VB			BR+VB(QL)		
Nome	klb	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)
RWNR_300_01	4	(26:23:417)	-	0	(71:55:665)	-	0	(75:4:677)	-	0	(160:34:608)	6	10
RWNR_300_02	4	(19:27:440)	-	0	(103:49:646)	-	0	(79:2:106)	-	0	(176:31:492)	6	20
RWNR_300_03	5	(3:47:836)	-	0	(100:26:981)	6	40	(44:40:685)	6	80	(36:51:817)	6	100
RWNR_300_04	5	(1:9:272)	-	0	(104:50:587)	6	50	(30:48:555)	6	90	(32:36:681)	6	100
RWNR_300_05	5	(16:40:506)	-	0	(72:53:456)	8	10	(75:2:174)	-	0	(92:54:432)	-	0
RWNR_300_06	5	(0:7:702)	-	0	(86:13:526)	-	0	(0:8:979)	6	100	(0:11:828)	6	100
Médias		T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)
		(11:16:29)	0	0,00	(90:1:644)	50	16,67	(50:47:863)	50	45	(83:16:810)	83,33	55,00

Tabela 29 – Resultados das instâncias da classe C4 com 400 nós

Instâncias		BR			BR+QL			BR+VB			BR+VB(QL)		
Nome	klb	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)	T. Médio	SV	C(%)
RWNR_400_01	4	(4:48:95)	-	0	(172:59:469)	-	0	(135:39:34)	5	70	(74:25:531)	5	100
RWNR_400_02	4	(12:5:582)	-	0	(42:32:137)	7	10	(53:50:846)	5	80	(75:30:382)	5	100
RWNR_400_03	5	(5:39:345)	-	0	(79:9:21)	6	30	(57:8:233)	6	90	(114:18:500)	6	90
RWNR_400_04	5	(12:47:544)	-	0	(120:36:899)	6	20	(57:42:748)	6	100	(159:4:316)	6	90
RWNR_400_05	5	(20:16:875)	-	0	(70:51:937)	-	0	(37:23:791)	-	0	(157:41:672)	-	0
RWNR_400_06	5	(26:31:817)	-	0	(224:21:122)	-	0	(209:24:992)	-	0	(134:51:817)	-	0
Médias		T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)	T. Médio	SV(%)	C(%)
		(13:41:543)	0	0	(118:25:97)	50	10	(91:51:607)	66,67	56,67	(119:18:703)	66,67	63,33