UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE ENGENHARIA DE PRODUÇÃO

BUSCA HEURÍSTICA ATRAVÉS DE ALGORITMO GENÉTICO E MEMÉTICO COM CONSTRUÇÃO DE VOCÁBULOS PARA O PROBLEMA DE ATRIBUIÇÃO DE LOCALIDADES A ANÉIS SONET

por

ANA CRISTINA GIRÃO E SILVA

MATEMÁTICA, UFRN, 2004

DISSERTAÇÃO SUBMETIDA AO PROGRAMA DE PÓS-GRADUAÇÃO DE ENGENHARIA DE PRODUÇÃO DA UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE

MESTRE EM CIÊNCIAS EM ENGENHARIA DE PRODUÇÃO

DEZEMBRO, 2008

Assinatura do Autor:

_____

APROVADO POR:

_____
Prof. Dario José Aloise, Dr. – Orientador, Presidente

_____
Prof. José Alfredo Ferreira Costa, Dr. – Examinador

_____
Christophe Didier Duhamel, PhD. – Examinador Externo

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

ANA CRISTINA GIRÃO E SILVA

Busca Heurística Através de Algoritmo Genético e Memético com Construção de Vocábulos para o Problema de Atribuição de Localidades a Anéis SONET

Natal, RN

2008

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

ANA CRISTINA GIRÃO E SILVA

Busca Heurística Através de Algoritmo Genético e Memético com Construção de Vocábulos para o Problema de Atribuição de Localidades a Anéis SONET

Dissertação de Mestrado apresentada à Universidade Federal do Rio Grande do Norte como parte dos requisitos para obtenção do título de Mestre em Engenharia de Produção.

Orientador:
Prof. Dr. Dario José Aloise

Natal, RN
2008

Busca Heurística Através de Algoritmo Genético e Memético com Construção de Vocábulos para o Problema de Atribuição de Localidades a Anéis SONET

Ana Cristina Girão e Silva

Dissertação de Mestrado apresentada à Universidade Federal do Rio Grande do Norte como parte dos requisitos para obtenção do título de Mestre em Engenharia de Produção.

Aprovada por:

Prof. Dr. Dario José Aloise - UFRN (Presidente)

Prof. Dr. José Alfredo Ferreira Costa - UFRN

Prof. PhD. Christophe Didier Duhamel -
Université Blaise Pascal

Natal, 23 de Dezembro de 2008

Agradecimentos

A Deus, por seu amor incondicional.

À minha família, em especial minha mãe e irmãs, pelo amor, incentivo, apoio emocional e tanta pasciência durante esse período.

Ao Professor Dario, por me receber como sua orientanda e ter dedicado, na medida do possível, parte do seu precioso tempo me ajudando com idéias e sugestões.

Aos amigos: Allison Guedes, pelas idéias e sugestões que contribuiram para este trabalho; Éberton Marinho, pela imprescindível ajuda na liguagem de programação C^{++} e dicas de LaTeX; e aos colegas do Laboratório de Programação Metaheurística - PROMETH.

À todos os demais amigos, em especial, Ana Katarina, Carlos Barboza e Emerson Faião pela amizade sincera nos momentos difíceis e inúmeros favores realizados.

Aos meus colegas de trabalho na INFRAERO, em especial: André Sena e Adailton Gomes (os chefes), Andrea e Eduardo. Esse quarteto consegue criar um ambiente de trabalho maravilhoso todos os dias, com muita responsabilidade e paixão pelo que fazem; Margarethe, Elvis e aos que frequentam a academia comigo depois do expediente, por me ouvirem e darem a maior força quando me viram estressada durante a fase de conclusão deste trabalho.

À CAPES, pelo apoio financeiro.

Resumo

As telecomunicações desempenham um papel fundamental na sociedade contemporânea. Mas à medida que novas tecnologias são introduzidas ao mercado, cresce também a demanda por novos produtos e serviços que dependem da infra-estrutura oferecida, tornando os problemas de planejamento de redes de telecomunicações, apesar da evolução tecnológica, cada vez maiores e complexos. No entanto, 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. Neste trabalho, foram desenvolvidas duas implementações metaheurísticas puras – Algoritmo Genético (AG) e Algoritmo Memético (AM) – além de uma terceira implementação híbrida – Algoritmo Memético com *Vocabulary Building* (AM+VB) – para um problema de telecomunicações que é conhecido na literatura por Problema de Atribuição de Localidades a Anéis SONET ou SRAP (do inglês, *SONET Ring Assignment Problem*). O SRAP surge durante a etapa do planejamento físico da rede e consiste na determinação das conexões entre um conjunto de localidades (clientes), de modo a satisfazer uma série de restrições ao menor custo possível. Esse problema é NP-difícil e portanto algoritmos exatos eficientes (de complexidade polinomial) não são conhecidos, podendo, inclusive, nem existir.

Palavras-chave: Problema de Atribuição de Localidades a Anéis SONET; Algoritmo Genético, Algoritmo Memético e *Vocabulary Building*.

Abstract

Telecommunications play a key role in contemporary society. However, as new technologies are put into the market, it also grows the demanding for new products and services that depend on the offered infrastructure, making the problems of planning telecommunications networks, despite the advances in technology, increasingly larger and complex. However, many of these problems can be formulated as models of combinatorial optimization, and the use of heuristic algorithms can help solving these issues in the planning phase. In this project it was developed two pure metaheuristic implementations – Genetic algorithm (GA) and Memetic Algorithm (MA) – plus a third hybrid implementation – Memetic Algorithm with Vocabulary Building (MA+VB) – for a problem in telecommunications that is known in the literature as Problem SONET Ring Assignment Problem or SRAP. The SRAP arises during the planning stage of the physical network and it consists in the selection of connections between a number of locations (customers) in order to meet a series of restrictions on the lowest possible cost. This problem is NP-hard, so efficient exact algorithms (in polynomial complexity) are not known and may, indeed, even exist.

Keywords: SONET Ring Assignment Problem, Genetic Algorithm, Memetic Algorithm, Vocabulary Building.

Sumário

Glossário

ADM	:	Add-Drop Multiplexer;
AG	:	Algoritmo Genético;
AM	:	Algoritmo Memético;
DCS	:	Digital Cross-Connect System;
SDH	:	Synchronous Digital Hierarchy;
VB	:	Vocabulary Building;
SONET	:	Synchronous Optical NETwork;
SRAP	:	SONET Ring Assignment Problem.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

O serviço de telecomunicações sempre teve um importante papel para a sociedade, mas com o advento da globalização ele passou a ser imprescindível. Acompanhamos a evolução das redes: à medida que novas tecnologias são introduzidas ao mercado, cresce a oferta de novos produtos e serviços (por exemplo, tranferências de arquivos de dados, voz e imagens, transações bancárias, reservas de passagens aéreas), provocando o aumento da demanda de usuários atraídos pela inovação e praticidade dos serviços oferecidos. Por outro lado, a infra-estrutura da rede vai ficando sem capacidade para atender os clientes com a mesma qualidade, ocasionando a insatisfação destes. Assim, a tecnogia evolui, a rede se expande e este ciclo se repete. A consequência disto, é que os problemas de planejamento de rede de telecomunicações, apesar da alta qualidade dos equipamentos desenvolvidos, têm se tornado cada vez maiores e complexos (OMIDYAR; ALDRIDGE, 1993) e (WANSEM; WU; CARDWELL, 1994).

É nesse contexto que empresas, indústrias e organizações como um todo, a fim de obterem vantagens competitivas diante da concorrência, investem cada vez mais em recursos tecnológicos que garantam a qualidade de seus produtos e serviços. Os problemas de planejamento de redes também têm despertado bastante interesse entre pesquisadores de diversas áreas, dentre as quais, destacacamos a Pesquisa Operacional pois muitas dessas questões da fase de planejamento podem ser formuladas como problemas de otimização combinatória. E, embora a aplicação de métodos exatos para resolução de problemas de otimização exiga, na maioria dos casos, um tempo computacional considerado inviável,

grande parte das aplicações reais necessitam apenas de um bom resultado aproximado em vez de resultados ótimos, favorecendo o uso de métodos heurísticos, cada vez mais frequentes na resolução de problemas dessa natureza.

O Problema de Atribuição de Localidades a Anéis SONET ou SRAP (do inglês, *SONET Ring Assignment Problem*), abordado neste trabalho, faz parte da etapa do planejamento físico de um da rede de telecomunicações. Este é um problema de otimização combinatória, e consiste na determinação das conexões entre um conjunto de localidades, de modo a satisfazer um conjunto de restrições ao menor custo possível. Dentre essas restrições, a capacidade de sobrevivência da rede, que é a capacidade de permanecer ativa em caso de falha numa localidade ou numa ligação, é fundamental. A topologia em anel em conjunto com a tecnologia SONET (*Synchronous Optical NETwork*) são bastante eficazes para atender esse requisito. Esse problema pertence à classe NP-Difícil (GOLDSCHMIDT; LAUGIER; OLINICK, 2003), ou seja, algoritmos exatos eficientes (de complexidade polinomial) não são conhecidos, podendo inclusive, nem existir.

As principais motivações para elaboração deste trabalho foram: primeiro, propor novos algoritmos a partir de métodos evolutivos que solucionassem de forma satisfatória o SRAP; e segundo, testar a técnica *Vocabulary Building* até então pouco explorada em problemas de telecomunicações. Assim, foram implementadas duas metaheurísticas puras – Algoritmo Genético (AG) e Algoritmo Memético (AM) – e outra híbrida – Algoritmo Memético com *Vocabulary Building* (AM+VB). Cada fase dos algoritmos propostos será estudada detalhadamente ao longo desse texto.

O restante desse trabalho está organizado da seguinte forma: no Capítulo 2 apresentaremos de forma detalhada o Problema de Atribuição de Localidades a Anéis SONET. No Capítulo 3, introduziremos as meta-heurística Algoritmos Genéticos, Algoritmos Meméticos e a técnica *Vocabulary Building*. Em seguida, no Capítulo 4 apresentaremos os algoritmos propostos. Os experimentos e resultados computacionais serão descritos e analisados no Capítulo 5. Por fim, a conclusão é a apresentada no Capítulo 6.

Capítulo 2

Apresentação do Problema

Diversos problemas reais do cotidiano podem ser modelados segundo problemas de otimização combinatória (PAPADIMITRIOU; STEIGLITZ, 1982), (COOK et al., 1998) e (LAWLER, 1976). Infelizmente, para a mairoria deles não existem métodos de resolução exata capazes de obter soluções ótimas em tempo computacional viável. Resultados teóricos publicados na década de 70 e inúmeras evidências estatísticas obtidas até os dias atuais, apontam para inexistência de procedimentos exatos capazes de resolver eficientemente uma determinada classe de problemas combinatórios, denominada NP-difícil. No entanto, algoritmos que consigam obter respostas para problemas dessa natureza tornam-se extremamente necessários, logo, a solução encontrada foi a de se tentar chegar a respostas que, embora não-ótimas, satisfaçam as necessidades daqueles que precisam delas. Dentro dessa classe de problemas de difícil resolução encontram-se alguns problemas de planejamento de redes de telecomunicações.

O planejamento de uma rede SONET com topologia em anel é uma tarefa bastante complexa (OMIDYAR; ALDRIDGE, 1993) e (WANSEM; WU; CARDWELL, 1994), podendo ser dividida em projeto físico, que é a determinação dos subconjuntos de localidades que darão origem aos anéis; e projeto lógico, estabelecimento das conexões entre as localidades. Neste capítulo, falaremos sobre a questão que envolve a determinação dos subconjuntos de localidades que comporão os anéis de uma rede SONET, conhecida como Problema de Atribuição de Localidades a Anéis SONET (SONET Ring Assignment Problem) (SRAP). Veremos que o SRAP pode ser descrito formalmente como um problema

de particionamento em grafos e tratado com as técnicas de otimização combinatória.

Este capítulo está organizado da seguinte maneira: na Seção 2.1 consta uma breve explicação sobre o planejamento de uma rede de telecomunicações. Em seguida, na Seção 2.2 apresentamos o Problema de Atribuição de Localidades a Anéis SONET, bem como alguns aspectos importantes sobre o padrão SONET. A descrição formal do problema é discutida na Seção 2.2.2 e as formulações matemáticas utilizadas são descritas na Seção 2.2.3. Por fim, na Seção 2.3 consta uma revisão dos trabalhos sobre o SRAP encontrados na literatura.

2.1 Planejamento de uma rede de telecomunicações

Uma rede de telecomunicações é usualmente composta de dois níveis: rede *backbone* e rede de acesso local (SORIANO et al., 1999). A primeira, facilita o tráfego de informações entre os usuários. A segunda, é encarregada de concentrar essas informações e disponibilizá-las para as localidades da rede *backbone*. O planejamento de uma rede *backbone* consiste em determinar e estabelecer uma conexão entre as localidades da rede, a um custo mínimo, respeitando um conjunto de restrições. A Figura 2.2 exemplifica o estado inicial de uma rede, ou seja, as localidades e as demandas existentes entre elas.

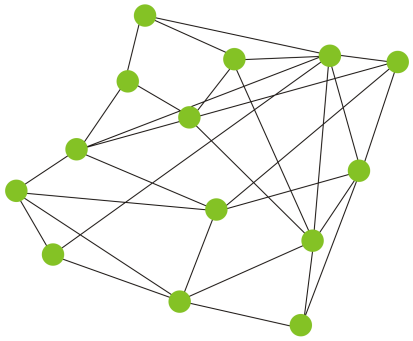

Figura 2.1: Estado inicial de uma rede backbone.

Uma das restrições impostas, por exemplo, é a capacidade de sobrevivência da rede, isto é, a capacidade de continuar ativa em caso alguma falha em uma localidade ou uma

ligação. Esta característica de sobrevivência é tratada nas redes *backbones* adotando-se a topologia em anel. A definição de uma topologia em anel apresenta várias versões. O problema que estudaremos neste trabalho apresenta uma versão que envolve uma estrutura hierárquica em que múltiplos anéis disjuntos são conectados através de um anel especial denominado Anel Federal (GOLDSCHMIDT; LAUGIER; OLINICK, 2003).

O planejamento de uma rede *backbone* seguindo essa topologia é uma tarefa complexa e pode ser dividida em duas etapas, projeto físico e lógico. O projeto físico consiste na determinação dos subconjuntos de localidades que darão origem aos anéis da rede. O projeto lógico estabelece uma conexão entre as localidades de cada subconjunto. A Figura 2.2 mostra o particionamento das localidades da rede acima, em subconjuntos. Um exemplo de conexão para estas localidades é apresentado na Figura 2.3.

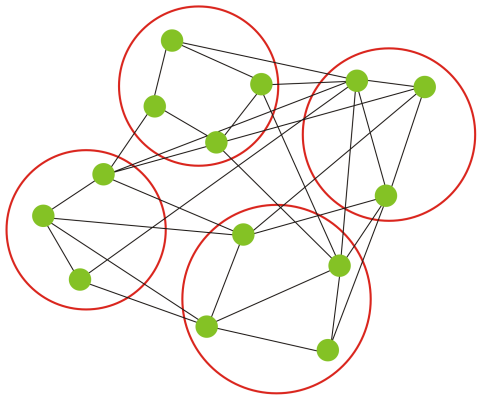

Figura 2.2: Projeto fisico de uma rede *backbone*.

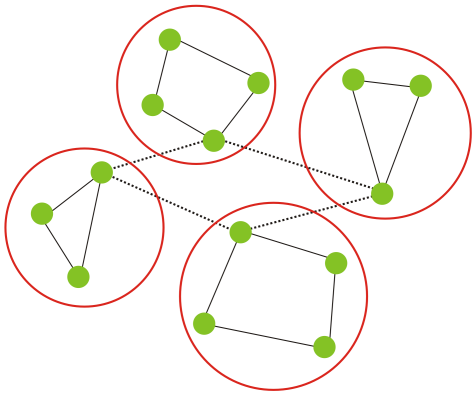

Figura 2.3: Projeto lógico de uma rede *backbone*.

Por se tratar da determinação de agrupamentos, o projeto físico pode ser visto

como um problema de particionamento dos vértices de um grafo (MEHOTRA; TRICK, 1998). A fase projeto lógico, construção dos anéis, corresponde a resolver uma instância do Problema do Caixeiro Viajante Simétrico (JüNGER; REINELT; RINALDI, 1995), em cada agrupamento e, em seguida, devido ao anel federal, resolver uma instância Caixeiro Viajante Simétrico generalizado (FISCHETTI; SALAZAR; TOTH, 1995) e (LAPORTE; NOBERT, 1983).

2.2 Problema de Atribuição de Localidades a Anéis SONET

O problema de Atribuição de Localidades a Anéis SONET surge na etapa do planejamento físico de uma rede *backbone*. Neste problema, cada localidade cliente deve ser atribuída a exatamente um anel SONET e estes são conectados através de um anel especial, chamado de Anel Federal. O objetivo é encontrar uma atribuição de localidades clientes que minimize o número total de anéis utilizados, sendo imposta uma restrição de capacidade de demanda sobre cada anel.

O SRAP apresenta a versão de topologia na qual um conjunto de n localidades são conectadas através de um ou mais anéis locais (ou simplismente anéis), os quais se comunicam entre si através de um anel especial, denominado de Anel Federal (AF). As localidades são conectadas aos anéis locais através de um dispositivo chamado *Add-Drop Multiplexer* (ADM). E um equipamento chamado *Digital Cross Connect System* (DCS)é responsável pela pela comunicação entre os vários anéis da rede conectando-os ao anel federal. O custo do DCS em relação aos demais equipamentos da rede corresponde ao mais caro, por esta razão, uma solucão ótima para o SRAP consiste em encontrar uma atribuição para as localidades que minimize o número de anéis locais e este, por sua vez, corresponde a quantidade de DCSs a serem instalados. Semelhantemente aos trabalhos encontrados na literatura, consideraremos as seguintes restrições o problema: 1) cada localidade deverá ser atribuída a um único anel; e 2) a capacidade máxima dos anéis locais e do anel federal será limitada por um valor comum B. A Figura 2.4 mostra uma rede SONET segundo a versão apresentada.

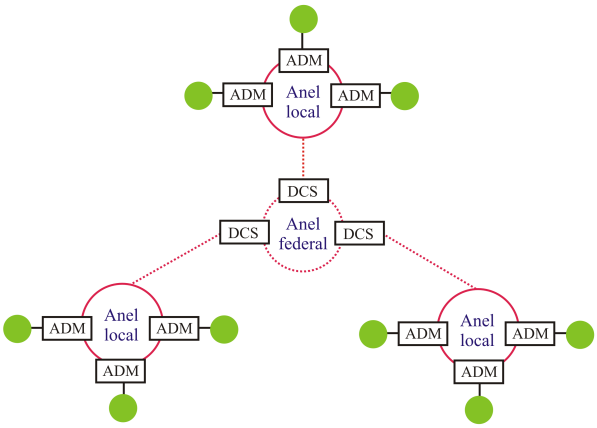

Figura 2.4: Topologia em anel de uma rede SONET.

2.2.1 Padrão SONET

Até meados da década de 1980, as empresas de telecomunicações utilizavam a hierarquia de multiplexação PDH (do inglês, *Plesyochronous Digital Hierarchy*). O crescimento dos serviços de comunicação, o surgimento da fibra ótica, a globalização e os problemas encontrados no padrão PDH (multiplexão dispendiosa, pouca padronização e baixas taxas de transmissão) cuminaram o desenvolvimento de uma nova hierarquia de transmissão.

O *Synchronous Optical Network (SONET)* é um padrão de transporte ótico, desenvolvido pela *Exchange Carriers Standard Association* (ECSA), para ser utilizado nos Estados Unidos. Posterior ao seu desenvolvimento, uma colaboração entre a *American National Standards Institute* (ANSI) e a *Comité Consultatif International Téléphonique et Télégraphique* (CCITT) produziram um padrão denominado *Synchronous Digital Hierarchy* (SDH), empregado na Europa e Japão. O padrão SDH é responsável pela intercomunicação entre a hierarquia não síncrona e o padrão SONET. Estima-se que o padrão SONET proporcione infra-estrutura de transporte para telecomunicações até o ano de 2020, aproximadamente.

O sinal SONET base é um sinal de transporte síncrono nível 1 ou, simplesmente, STS-1 (do inglês, *Synchronous Transport Signal level 1*) que opera a uma taxa de 51,84 Mb/s. Os sinais de níveis mais altos são múltiplos inteiros do STS-1. O conjunto de todos esses sinais forma a família STS-N. A Tabela 5.1 mostra maiores detalhes sobre os sinais e taxas de transmissão do padrão SONET e seus correspondentes no padrão SDH.

Sinal SONET	Taxa de bits (Mb/s)	Equivalente SDH
STS-1, OC-1	51,84	STM-0
STS-3, OC-3	155,52	STM-1
STS-12, OC-12	622,08	STM-4
STS-48, OC-48	2488,32	STM-16
STS-192, OC-192	9953,28	STM-64
STS-768, OC-768	39813,12	STM-256

Tabela 2.1: Hierarquia SONET/SDH.

O custo de uma rede SONET com a topologia em anel pode sair bastante caro, pois quanto mais anéis locais tiver a rede, maior será o gasto com DCSs. Em compensação, essa topologia possui uma grande vantagem que é a de assegurar a sobrevivência da rede em caso de falha ou rompimento de algum cabo de fibra ótica. Se isso acontecer, os multiplexadores enviarão automaticamente os serviços afetados, por um caminho alternativo. A Figura 2.5 simula a ocorrência de falha num anel SONET onde, inicialmente, o fluxo de dados no anel se dava no sentido horário mas, ocorrendo uma falha no cabo de fibra ótica, o fluxo nos dois sentidos foi ativado. Não havendo por tanto, interrupção dos serviços de comunicação da rede.

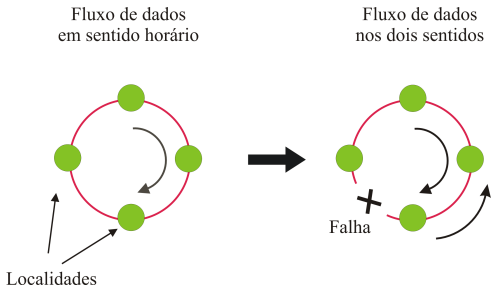

Figura 2.5: Sobrevivência de uma rede SONET com topologia em anel.

2.2.2 Descrição Formal do SRAP

Conforme foi visto na Seção 2.1, o SRAP pode ser descrito como um problema de particionamento de grafos. Neste caso, os seus vértices representam os clientes da rede (localidades), e os pesos associados às arestas, as demandas de tráfego entre eles. Assim, o problema pode ser formulado da seguinte maneira:

Seja $G = (V, E)$ um grafo completo não direcionado e um inteiro positivo B. Associado a cada aresta $(u, v) \in E$ está um inteiro não negativo d_{uv}. Uma solução viável do problema, corresponde a uma partição do conjunto V de vértices em k subconjuntos disjuntos V_1,V_2,...,V_k tal que:

$$\sum_{u,v \in V_i, u<v} d_{uv} + \sum_{u \in V_i, v \notin V_i} d_{uv} \leq B \quad \text{onde} \quad i = 1, 2, \ldots, k \tag{2.1}$$

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{u \in V_i} \sum_{v \in V_j} d_{uv} \leq B \tag{2.2}$$

A restrição definida pela equação 2.1 estabelece o limite de capacidade B aos anéis locais da rede. Por sua vez, a equação 2.2 impõe o limite B, de tráfego máximo, ao anel federal. O inteiro d_{uv} corresponde à demanda de tráfego entre as localidades u e v em ambos os sentidos. Se não houver demanda entre u e v então o valor de d_{uv} será zero.

2.2.3 Formulações Matemáticas

Seja S uma solução do problema com k anéis, os valores associados aos tráfegos de um anel de índice a em S são:

$$demI_a = \sum_{u,v \in a, u<v} d_{uv} \tag{2.3}$$

$$demE_a = \sum_{u \in a, v \notin a} d_{uv} \tag{2.4}$$

$$demT_a = demI_a + demE_a \tag{2.5}$$

Em que as equações 2.3, 2.4 e 2.5 representam, respectivamente, as demandas interna, externa e total no anel *a*. O anel *a* é viável quando $demT_a$ for menor ou igual a *B*. A demanda total no anel federal da solução *S* é dada por

$$demAF = \frac{\sum_{i=1}^{k} demE_i}{2} \tag{2.6}$$

O anel federal será viável caso *demAF* seja menor ou igual a *B*. Se todos os anéis (inclusive o anel federal) forem viáveis então *S* é uma solução viável.

2.3 Revisão de Literatura

Em (GOLDSCHMIDT; LAUGIER; OLINICK, 2003) os autores destacaram a relevância do SRAP no projeto de redes de telecomunicações que empregam a tecnologia SONET, especialmente, as que adotam a topologia em anel. Propuseram técnicas de programação inteira para o SRAP, demonstram que o problema é NP-Difícil, e ainda desenvolveram três heurísticas gulosas: *edge-based*, *cut-based* e *node-based*. Estas heurísticas foram testadas sobre um conjunto de 160 instâncias cujo número de localidades varivavam entre 15 e 50. Parte dessas instâncias foram geradas aleatoriamente, enquanto que outras são instâncias reais obtidas de uma empresa de telecomunicações. Evidências empíricas mostraram que os algoritmos heurísticos apresentam um bom desempenho. Cerca de $97,46\%$ das instâncias investigadas foram solucionadas de forma ótima.

Em (ARINGHIERI; DELL'AMICO, 2001) os autores introduziram um algoritmo busca tabu para resolução do SRAP denominado BTS (*Basic Tabu Search*). Os principais elementos do BTS são: quatro funções objetivo (z_1, z_2, z_3 e z_4), uma vizinhança chamada *node improvement neighborhood* (N_1); e duas listas tabu (*node-list* e *node-from-list*. Também foi proposto, um algoritmo busca tabu com oscilação estratégica, o TSSO

(*Tabu Search with Stategic Oscillation*). O qual, adiciona uma outra vizinhança *empty minimum cardinality ring neighborhood* (N_2), ao algoritmo anterior. Neste trabalho, também foram implementadas as seguintes técnicas de intensificação e diversificação: *Path Relinking*, *eXploring Tabu Search* e *Scatter Search*. Os autores realizaram testes sobre o mesmo cojunto de intâncias introduzidas em (GOLDSCHMIDT; LAUGIER; OLINICK, 2003), comparam o desempenho de seus algoritmos com o das heurísticas construtivas propostas por (GOLDSCHMIDT; LAUGIER; OLINICK, 2003), e verificaram que os algoritmos propostos apresentaram resultados melhores.

Posteriormente, em 2005, Aringhieri e Dell'Amico publicaram um novo trabalho sobre o SRAP (ARINGHIERI; DELL'AMICO, 2005). No qual, propõem uma extensão das vizinhanças aplicadas em (ARINGHIERI; DELL'AMICO, 2001), que só permitia troca ou permutação de localidades, caso os anéis resultantes fossem viavéis. Neste novo artigo é permitido às vizinhanças a obtenção de anéis inviáveis. O algoritmo aqui proposto é fundamentado em vizinhanças múltiplas e denominado DMN (*Diversification by Multiple Neighborhhods*). Este procedimento, na maior parte do tempo, trabalha com a vizinhaça de busca N_2, bastante eficiente, mas que em certo momento assume outra vizinhança, N_3, cujo objetivo é produzir soluções que sejam diferentes, mas não necessariamente boas ou viáveis. Os resultados mostraram que o procedimento DMN encontrou a solução ótima em $100,0\%$ das instâncias produzidas por (GOLDSCHMIDT; LAUGIER; OLINICK, 2003).

Em (MACAMBIRA, 2003) e (MACAMBIRA; MACULAN; SOUZA, 2005) são apresentadas e discutidas várias formulações de programação linear inteira para o SRAP. É feita uma investigação sobre a estrutura facial do poliedro associado ao problema e novas famílias de facetas são introduzidas. Também é apresentado um estudo sobre o problema de simetria que ocorre nas soluções. Além disso, foi proposto um algoritmo exato *branch-and-price* para a resolução do SRAP, através do qual, foram obtidos os valores das soluções ótimas para duas classes de instâncias da literatura, denotadas por C1 e C2.

Em (MACAMBIRA; FILHO; SOUZA, 2003), os autores desenvolveram uma metaheurística GRASP para o SRAP denominda GRASP básico (GB). Em (BASTOS; OCHI; MACAMBIRA, 2005b), foi proposta uma nova versão do GB denominada Algoritmo Construtivo baseado em Vizinhança Relativa, ou GB+ACVR. Um novo algoritmo

GRASP foi desenvolvido e implementado em (BASTOS; OCHI; MACAMBIRA, 2005a) uma outra versão que adicionava Path Relinking. Este novo algoritmo foi denominado GB+ACVR+PR e tinha como propósito aumentar a robustez do GB+ACVR, a fim de melhorar a qualidade das soluções obtidas. O Artigo (BASTOS; OCHI, 2008) publicado em junho deste ano, apresentou um Algoritmo Genético (AG) e um AG com *Evolucionary Path-Relinking* (AG+EvPR). O AG+EvPR obteve, respectivamente, 100% e 98, 3% dos valores ótimos para as instâncias das classes C1 e C2.

Em (SOARES, 2008) foi desenvolvido um algoritmo Busca Tabu em conjunto com a *Vocabulary Building*, com vocábulos 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. A BT apresentou um bom desempenho, atingindo os ótimos da grande maioria das instâncias C1 e C2. Foi observado a partir dos resultados, que os vocábulos 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 completo demonstrou aumento da robustez. Além de aumentar a quantidade de ótimos, a distância em relação ao ótimo (*gap*) diminuiu.

Capítulo 3

Métodos Computacionais

Neste capítulo faremos uma apresentação das meta-heurísticas Algoritmos Genéticos e Algoritmos Meméticos, bem como, da técnica *Vocabulary Build*, utilizados neste trabalho para resolver o Problema de Atribuição de Localidades a Anéis SONET. Por se tratar de um problema NP-Difícil (GOLDSCHMIDT; LAUGIER; OLINICK, 2003), algoritmos exatos com tempo de execução polinomial não são conhecidos, e o uso de heurísticas e meta-heurísticas acabam se tornando uma ferramenta fundamental para obtenção de soluções ótimas ou, pelo menos, boas soluções viáveis para os problemas dessa classe.

Nas Seções 3.1 e 3.2 são apresentadas, respctivamentes, as meta-heuristicas Algoritmos Genéticos e Algoritmos Meméticos. A técnica *Vocabulary Build* é descrita na Seção 3.3.

3.1 Algoritmos Genéticos

Os Algoritmos Genéticos, AGs, são métodos de otimização inspirados nos mecanismos de evolução de populações de seres vivos. Os AGs seguem o princípio da seleção natural e sobrevivência do mais apto, apresentado por Charles Darwin em seu livro **A Origem das Espécies**. A primeira idéia sobre os AGs, foi desenvolvida por Jonh Holland e colaboradores na universidade do Michigan durante as décadas de 1960 e 1970. Sua meta original era desenvolver maneiras pelas quais o fenômeno da adaptação natural pudesse ser importado para os sistemas computacionais.

O Algoritmo Genético proposto por Holland (HOLLAND, 1975) é um método em que uma "população" de "cromossomos" evolui através de várias "gerações" depois de passar por um tipo de "seleção natural" que utiliza operadores inspirados na genética, *crossover* (cruzamento) e mutação, para evoluir. Aqui, cromossomos são estruturas de dados, geralmente vetor ou cadeia de bits, que representam as soluções do espaço de busca do problema em questão. Na terminologia dos AGs esses são chamados de indivíduos. Um AG padrão pode ser descrito da seguinte forma: Uma população de indivíduos (populção inicial) é construída inicialmente, ou de forma aleatória ou por meio de alguma heurística construtiva elaborada para o problema. Em seguida, todos os seus idivíduos são avaliados através de uma fução denominada Função de Aptidão (*fitness*)que funciona como uma "nota" que cada idivíduo recebe para indicar a qualidade da solução a que corresponde. Dois cromossomos são selecionados através de um método de seleção (serão detalhados adiante) para sofrerem uma recombinação por meio dos mecanismos de *crossover* e mutação (GLOVER; KOCHENBERGER, 2003). A idéia presente no AG, é que indivíduos que possuem boas características, combinados com outros idíviduos de boa qualidade, gerem idivíduos ainda melhores para as próximas gerações. O Algoritmo 1 apresenta as etapas de um algoritmo genético padrão.

Os melhores indivíduos de uma população podem ser diretamente copiados de uma geração para outra. Esse procedimento é denominado de elitismo e tem como objetivo manter as características dos melhores cromossomos nas gerações futuras.

Algoritmo 1: Algoritmo Genético Padrão

```
1:  procedimento AlgoritmoGenetico
2:  P ← população Inicial;
3:  avaliar (P);
4:  enquanto condição de parada não satisfeita faça
5:  P' ← seleção (P);
6:  P ← cruzamentos (P');
7:  P ← mutações (P);
8:  avaliar (P);
9:  fim enquanto
10: solução ← melhor indivíduo (P);
11: fim AlgoritmoGenetico
```

3.1.1 Cromossomos

Existem vários tipos de representações possíveis para os cromossomos. A escolha do mais adequado vai depender do problema tratado, dos parâmetros que precisam ser representados e, também, da cardinalidade envolvida entre o espaço de busca do problema e o espaço dos cromossomos. Segundo Falknauer em (FALKNAUER, 1998), existem três situações possíveis representadas graficamente na Figura 3.1. Na primeira, um-pra-muitos, uma mesma solução poderá ser representada por vários cromossomos distintos. Neste caso, o código é redundante e isto causa um grande problema à eficiencia do AG. Na segunda, um-pra-um, cada solução é represenada por exatamente um cromossomo que decodificado retornará a mesma solução. Obviamente, este é o melhor caso pois o AG contará com o menor espaço possível, sem que nenhuma informação seja perdida. Na última situação, muitos-pra-um, várias soluções do problema podem ser representadas por um único cromossomo, consequentemente causará uma redução do espaço de busca do AG em relação ao espaço de busca do problema.

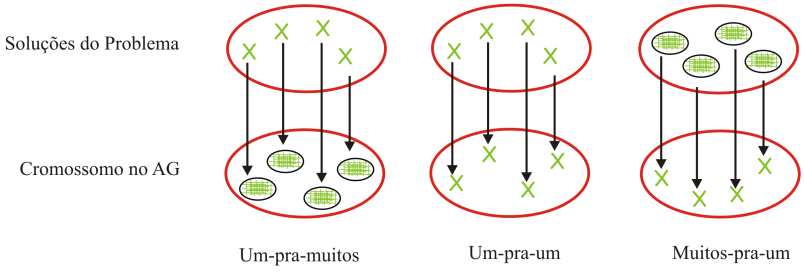

Figura 3.1: Cardinalidade entre o espaço de busca do problema e o espaço do AG.

Duas representações bastante conhecidas são a representação binária e a representação por permutação. A Figura 3.2 mostra um cromossomo que poderia representar uma solução para o Problema da Mochila, onde gene informaria se um determinado objeto vai entrar na mochila ou não. Já a Figura 3.4, poderia representar uma solução para o Problema do Caixeiro Viajante (*Traveling Salesperson Problem*), onde cada gene informaria a posição em que uma determinada cidade aparece numa rota.

Também é possível utilizarmos uma lista de vetores para compor a representação de um mesmo cromossomo. Esse tipo de representação pode ser aplicada a diversos problemas, dentre os quais, se encontram os de agrupamento, que é o caso do SRAP. A Figura 3.4

Figura 3.2: Representação binária de uma solução.

Figura 3.3: Representação de uma solução por permutação.

traz do lado esquerdo, um exemplo de agrupamento, e do direito, a representação do seu cromosso. Este é formado por três vetores nos quais, constam os elementos pertecentes ao seu grupo. Note que esses vetores não precisam possuir tamanho fixo. Utilizamos essa representação para as soluções do SRAP, maiores detalhes serão apresentados no Capítulo 4.

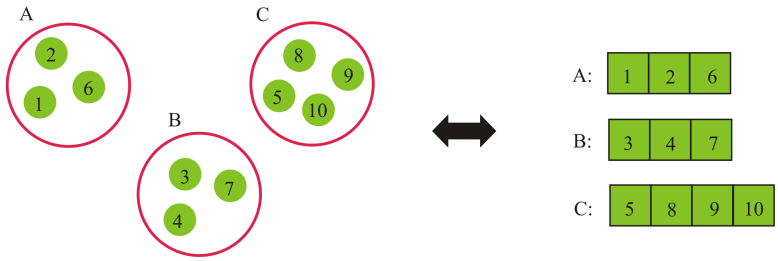

Figura 3.4: Exemplo de cromossomo formado por vários vetores.

3.1.2 Seleção

A seleção é a fase do AG na qual serão escolhidos os indivíduos mais aptos da população (cromossomos pais) que irão passar pelos operadores de *crossover* e mutação para se reproduzirem, gerando descendentes (cromossomos filhos) que são variantes dos pais. A seleção é baseada na aptidão dos indivíduos, pois quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes (DARWIN, 1859).

O Algoritmo Genético tradicional desenvolvido por Holland (HOLLAND, 1975), utilizava um método de seleção chamado *roulette-wheel method* (método da roleta), no qual indivíduos mais aptos possuem maior probabilidade de ser selecionados. Cada in-

divíduo é representado por uma fatia em uma roleta, proporcional à sua adaptação. A cada giro da roleta, um indivíduo é selecionado, tendo maior chance aqueles que possuem as maiores fatias (GLOVER; KOCHENBERGER, 2003). A Figura 3.5 exemplifica uma situação em que quatro cromossomos s_1, s_2, s_3 e s_4 ocupam a área do setor da roleta correspondente à sua aptidão que são 40, 20, 15 e 25 respectivamente. Depois, um número randômico é sorteado no intervalo [0, 1]. Se, por exemplo, o número sorteado foi 0.35, então o cromossomo a ser selecionado será s_1.

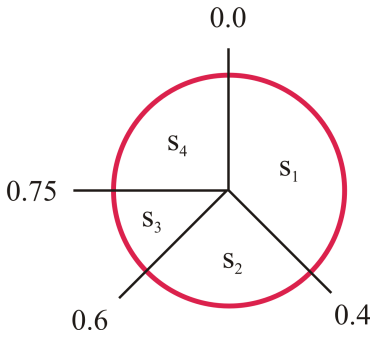

Figura 3.5: Ilustração do método da roleta.

Com a evolução dos AGs, novos métodos de seleção foram desenvolvidos como o *tournament selection* (método de seleção por torneio), e o *linear ranking selection* (método de seleção por nivelamento linear). O método da roleta foi o utilizado no presente trabalho.

3.1.3 Operadores Genéticos

Os operadores de cruzamento e mutação são os principais mecanismos de busca utilizadosdos pelos AGs para explorar regiões desconhecidas do espaço de busca.

1. Cruzamento

   O operador genético conhecido por cruzamento (*crossover*), é um mecanismo que cobina os indivíduos selecionados (pais), gerando os indivíduos (filhos) da próxima geração, que terão boas chances de possuir características melhores que as de seus antecedentes. Existem diversos procedimentos de *crossover* (GLOVER; KOCHENBERGER, 2003), o mais clássico é o *crossover* de um ponto. Neste, uma posição de corte é aleatoriamente escolhida entre os cromossomos pais e então a parte esquerda

de cada cromossomo é recombinada com a parte direita do outro (figura 3.6). Bem similar a este modelo, é o *crossover* de dois pontos, no qual um segundo ponto de corte é adicionado e a iformação dos pais é permutada entre eles (figura 3.7).

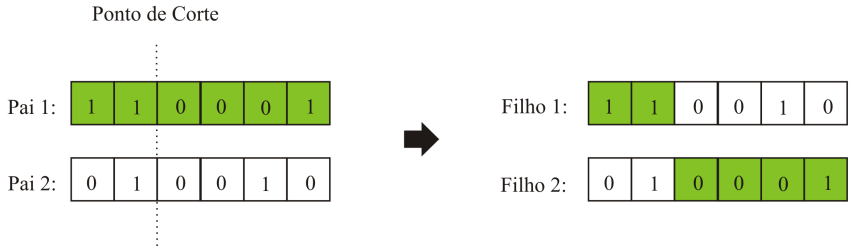

Figura 3.6: *Crossover* de um ponto.

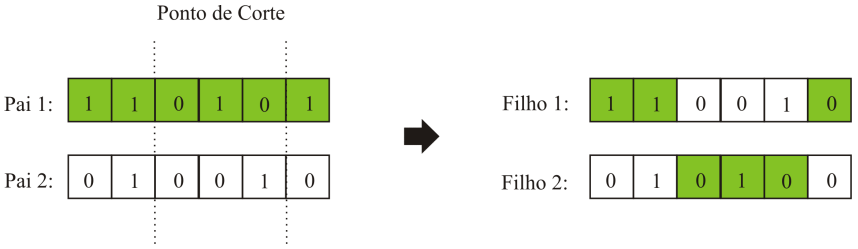

Figura 3.7: *Crossover* de dois pontos.

2. Mutação

O operador genético mutação é aplicado, com dada probabilidade, em cada bit dos cromossomos obtidos através do *crossover*. Ela inverte os valores de bits, por exemplo, muda o valor de um dado bit de 1 para 0 ou de 0 para 1. Este operador tem como função gerar perturbações aleatórias no processo de busca, e tem por objetivo diversificar populações homogêneas, evitando-se uma convergência prematura da população. A mutação melhora a diversidade dos cromossomos na população, mas por outro lado, destrói informação contida no cromossomo, portanto, deve ser utilizada uma taxa de mutação pequena (normalmente de $0,1\%$ a 5%), mas suficiente para assegurar a diversidade. A figura 3.8 ilustra um exemplo clássico de mutação.

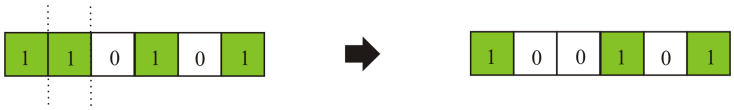

Figura 3.8: Operador clássico de mutação.

3.2 Algoritmos Meméticos

Os Algoritmos Meméticos, ou AMs, constituem uma variação dos Algoritmos Genéticos. Estes são acrescidos de uma busca local em seus operadores de forma a agilizar sua execução (NORONHA; ALOISE, 2001). O mecanismo de busca local e demais acréscimos ao formato básico do AG permitem que o algoritmo evolua além da "evolução genética", baseada praticamente só em cruzamento e mutação. Essas novas facetas do processo de busca evolucionário claramente não se encaixam na formulação tradicional de um algoritmo genético e um novo termo foi dado para denominar tais procedimentos (MOSCATO, 1989).

Enquanto o algoritmo genético tenta imitar o processo de evolução natural, o algoritmo memético é um modelo computacional que procura imitar o processo de evolução cultural, em que cada indivíduo pode vir a dar uma contribuição significativa para o processo de busca como um todo. No algoritmo genético, os genes são transmitidos através das gerações, enquanto que os memes (unidade replicante de informação cultural) podem ser transmitidos de um indivíduo para outro sem que uma geração transcorra.

Algoritmo 2: Algoritmo Memético Padrão

```
1:  procedimento AlgoritmoMemetico
2:  P ← população Inicial;
3:  busca local (P);
4:  avaliar (P);
5:  enquanto condição de parada não satisfeita faça
6:  P' ← seleção (P);
7:  P ← cruzamentos (P');
8:  P ← mutações (P);
9:  busca local (P);
10: avaliar (P);
11: fim enquanto
12: solução ← melhor indivíduo (P);
13: fim AlgoritmoMemetico
```

Este método tem aparecido na literatura como uma Metaheurística muito eficiente na resolução de problemas de Otimização Combinatória. Se comparado, por exemplo, ao

Algoritmo Genético, consegue realizar uma busca mais rápida e com melhores resultados.

Neste trabalho, utiliza-se o modelo de algoritmo memético acima para a inserção dos procedimentos de vocabulary building apresentados.

3.3 Vocabulary Building

A técnica de otimização conhecida por Construção de Vocábulo (*Vocabulary Building*) foi idealizada por Fred Glover e pode ser considerada como uma variação de Path Relinking (GLOVER, 1992), tendo em vista que também utiliza o conceito de "vizinhanças construtivas" para gerar novas soluções a partir de outras já existentes. A diferença, no entanto, está na possibilidade de se utilizar soluções parciais em conjunto com soluções completas. A técnica recebe esse nome por causa da analogia com o processo de formação das palavras, introduzidas na linguagem a fim de expressarem novas necessidades decorrentes do desenvolvimento da cultura humana.

A técnica aparece em diversos trabalhos: Em (GLOVER; LAGUNA, 1993), foi proposta como uma estratégia a ser implementada dentro da Busca Tabu. Em 1995, surgiram dois trabalhos cujos conteúdos possuem ligação com as idéias relativas ao Vocabulary Building (ROCHAT; TAILLARD, 1995) e (KELLY; XU, 1995) aplicadas a problemas de roteamento de veículos. Em 1997, a técnica é mais uma vez mencionada em um trabalho que trata de Scatter Search e Path Relinking (GLOVER, 1997) . A partir de 1997, surgiram outras publicações que abrangem o Vocabulary Building, como: (GLOVER; LAGUNA, 1997); (SCHOLL; KLEIN; DOMSCHKE, 1998); (GLOVER, 1999), (GLOVER; LAGUNA; MARTI, 2000), (GLOVER, 2003). Em 2006, a técnica foi aplicada em um Algoritmo Memético para o Problema Do Caixeiro Viajante Assimétrico em (GUEDES; ALOISE, 2006) e, recentemente, em um algoritmo Busca Tabu para o Problema de Atribuição de Localidades em Anéis SDH/SONET em (SOARES, 2008).

A idéia central da Construção de Vocábulos consiste, primeiramente, em identificar boas soluções parciais (trechos de soluções completas) prosseguindo, posteriormente, com a busca pelo ótimo global. Uma boa solução parcial pode ser, por exemplo, uma configuração que esteja presente em várias soluções-elite. O objetivo é tomar vantagem destes

contextos em que certas configurações parciais de soluções ocorrem com frequência como componentes de boas soluções completas. Esta estratégia de busca por "oas configurações parciais" pode ajudar a evitar a explosão combinatorial resultante da manipulação de apenas elementos primitivos.

Denominamos por "Vocábulo" um elemento identificado como sendo importante para a obtenção de soluções de boa qualidade (por exemplo, um subconjunto de localidades para o SRAP). A partir desses vocábulos, é possível chegar a combinações mais complexas e úteis. O método também requer uma estrutura para armazenar os vocábulos identificados como úteis. Esse conjunto de vocábulos recebe o nome de "*Pool* ou Coleção de vocábulos" que funciona com uma espécie de memória adaptativa, ou seja, sofre modificações no decorrer da busca (GUEDES; ALOISE, 2006). Pois a cada momento, soluções parciais distintas podem ser de diferente atratividade. Por esta razão, é interessante que o *Pool* mantenha somente as que forem mais promissoras dentro do processo de busca.

A idéia de *Vocabulary Building* é bem geral e pode ser aplicada em diversos problemas e de várias maneiras. Neste trabalho, idealizou-se uma implementação das técnicas de *Vocabulary Building* em um método que baseia-se na contração de vértices.

Capítulo 4

Algoritmos Propostos

Neste capítulo serão apresentados detalhes de implementação dos algoritmos aqui desenvolvidos para a resolução do Problema de Atribuição de Localidades a Anéis SONET. Foram propostos um Algoritmo Genético Puro, um Algoritmo Memético Puro e um Algoritmo Memético hibridizado com a técnica *Vocabulary Building*. O Algoritmo Genético proposto obtem sua popução inicial a partir das heurísticas *Ramdon Edge-Based* e *Ramdon Cut-Based* propostas por (BASTOS, 2005) que fizeram uma adaptação das heurísticas construtivas introduzidas por (GOLDSCHMIDT; LAUGIER; OLINICK, 2003), incluíndo sobre estas um fator de aleatoriedade. Foram aplicados os operadores genéticos de elitismo e o *Crossover BPX* para reprodução das novas gerações. O Algoritmo Memético proposto adiciona uma busca local ao AG que utiliza duas vizinhanças propostas por (Citação) denominadas N_1 e N_3. A técnica *Vocabulary Building* aplicada ao AM é introduzida na fase de busca local sobre as duas vizinhanças mencionadas.

A Seção 4.1 apresenta uma descrição detalhada do Algoritmo Genético Proposto. O Algoritmo Memético Puro é a presentado em detalhes na Seção 4.2 e a iplementação com *Vocabulary Building* na Seção 4.3.

4.1 Algoritmo Genético Puro Aplicado ao SRAP

Conforme mencionado no Capítulo 3, o AG é uma metaheurística que trabalha com uma "população" de "indivíduos" que evoluem através das "gerações" depois de passarem por um tipo de "seleção natural" que utiliza operadores inspirados na genética, *crosso-*

ver (cruzamento) e mutação (GLOVER; KOCHENBERGER, 2003). Nesta seção são discutidos os detalhes de implementação do AG proposto para resolução SRAP.

4.1.1 Representação do Cromossomo

Utilizamos as listas de vetores para representar os cromossomos dos indivíduos do AG implementado. Conforme visto na Seção 2.2.2, uma solução do SRAP é uma partição de um conjunto de n localidades em k subconjuntos disjuntos. Assim, cada vetor da lista que compõe o cromossomo, conterá as localidades que formam determinado anel da solução por ele representada. A figura 4.1 exemplifica uma solução S do propblema para um conjunto de 5 localidades distribuídas em dois anéis locais. Um deles com as localidades 1, 2 e 3; e o outro com 4 e 5. A parte (A) da figura indica como ficaria a configuração dessa rede. A parte (B) representa o agrupamento entre as localidades. E (C) ilustra o cromossomo com dois vetores, um para cada anel local. Note que o anel federal não precisa ser representado por um vetor, uma vez que não existem localidades a ele atribuídas.

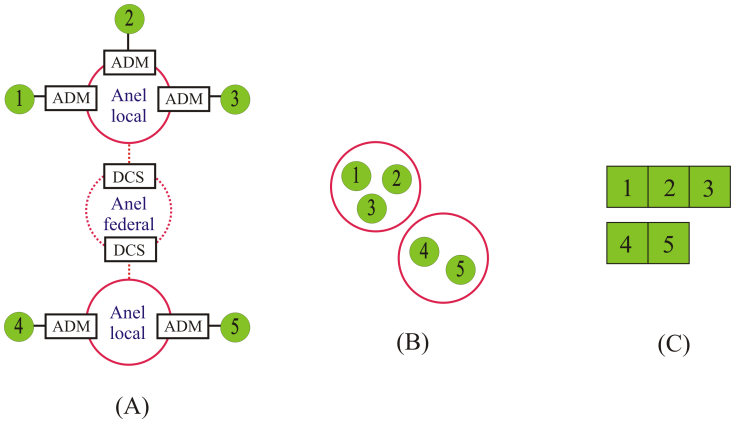

Figura 4.1: (A) Configuração de uma solução, (B) Agrupamento das localidades e (C) Cromossomo.

Esse tipo de representação possibilitou tratar a questão da redundâcia, visto na Seção 3.1.1 e bastante comum em problemas de agrupamento. A redundância ocorre, quando a representação escolhida permite a geração de diferentes cromossomos para indicar a mesma solução, no caso do SRAP, o mesmo agrupamento (Figura 4.2). Em nosso AG, este problema foi minimizado fazendo-se a ordenação dos vetores, isto possibilitou a

verificação da redundância e partir daí realizar um trabalho de tentar bloquear a geração de cromossomos que representassem soluções já existentes na população. Não conseguimos eliminar esse problema em 100% devido ao aumento do tempo computacional, então optamos, por limitar o bloqueio em um número de x tentativas. Vários testes foram realizados com e sem tratamento das soluções repitidas. Mas o procedimento mostrou-se bastante eficaz em seu objetivo, uma vez que as populações geradas sem tratamento apresentaram-se muito homogêneas, e que dependendo da instância, até convergiam para um ótimo local.

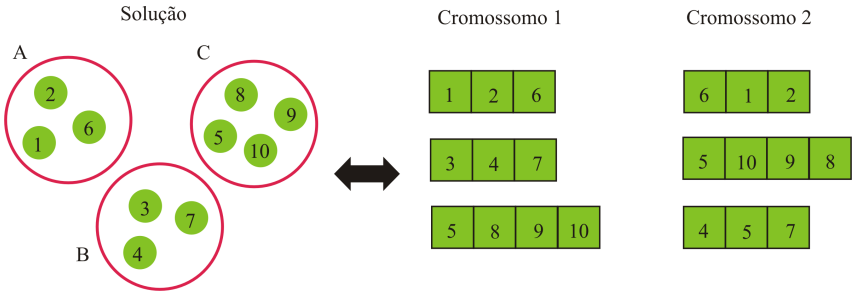

Figura 4.2: Exemplo de dois cromossomos distintos que representam a mesma solução

4.1.2 População Inicial

A população inicial do AG proposto é produzida através das heurística *random edge-based* e *cut edge-based* desenvolvidas por (BASTOS, 2005), e cada uma é resposável pela geração de metade dos índivíduos. Essas são uma extenção das heurísticas construtivas *edge-based* e *cut-based* introduzidas por(GOLDSCHMIDT; LAUGIER; OLINICK, 2003). Inicialmente, tanto *edge-based* quanto *cut-based*, atribuem cada um dos n nós a um anel diferente construindo, desta forma, uma solução com n anéis. Daqui em diante, nos referiremos a essa solução inicial por solução trivial. Note que ela só será inviável caso a demanda no anel federal seja maior do que B, portanto, se o problema for viável, todos os anéis da solução trivial também o serão. Em seguida, a cada iteração, dois anéis distintos são unidos, caso o anel resultante desta união seja viável, isto é, atenda as restrições impostas. A cada iteração da fase construtiva, a escolha do próximo elemento a compor a solução é determinada por alguma função gulosa. No caso de *edge-based*, as arestas são colocadas em ordem crescente de suas demandas, e a escolha dos anéis para a união é

feita, tomando-se a aresta de maior peso que ainda não foi selecionada. Para *cut-based*, a escolha dos anéis é realizada, tomando-se o tráfego máximo entre dois anéis quaisquer.

As versões das heurísticas *edge-based* e *cut-based* implementadas por (BASTOS, 2005) foram escolhidas para este trabalho, porque as de Goldberg, da forma como foram definidas no trabalho original, produzem sempre a mesma solução. As novas implementações incluem aleatoriedade em alguns passos para aumentar a diversificação, tornando os procedimentos mais adequados para a fase de construção da população inicial do AG. Na heurística *random edge-based* o componente probabilístico foi adicionado no momento da escolha da aresta. Quanto maior o seu peso, maior será a chance de ser escolhida. Seu pseudo código é apresentado no Algoritmo 3.

Algoritmo 3: Heurística *random edge-based*

```
1: procedimento randomEdgeBased()
2: iniciar();
3: S ← construirSolucaoTrivial();
4: LRC ← {(u, v) ∈ E | u < v};
5: ordenar(LRC);
6:    enquanto LRC ≠ ∅ faça
7:    (u, v) ← retirarAresta(LRC);
8:       se anel(u) ∪ anel(v) for viável então
9:          unirAneis(anel(u), anel(v));
10:      fim se;
11:   fim enquanto;
12: retornar(S);
13: fim randomEdgeBased;
```

No passo 3, a solução trivial é criada e atribuída a S. Em seguida, no passo 4, a LRC recebe todas as arestas do grafo inicial e é ordenada no passo 5. Cada aresta (u, v) ∈ LRC é obtida (passo 7) e analisada individualmente (passo 8). Neste ponto, um fator de aleatoriedade é introduzido na escolha das arestas, dando as primeiras maior probabilidade de serem escolhidas. Se a demanda resultante da união do anel anel(u) com o anel anel(v) for menor ou igual a B, os anéis são unidos (passo 9). Esse procedimento é repetido até que a lista de arestas candidatas LRC fique vazia.

Em *random cut-based* o fator aleatório é inserido no momento da seleção do tráfego, quanto maior o tráfego entre os anéis, maiores as chances de serem escolhidos. O pseudo código deste procedimento é descrito no Algoritmo 4.

Algoritmo 4: Heurística *random cut-based*

```
1: procedimento randomCutBased()
2: iniciar();
3: S ← construirSolucaoTrivial();
4: enquanto for possivel reunir aneis faça
5:    LRC ← {(i, j) ∈ E_virtuais | i < j};
6:    ordenar(LRC);
7:    enquanto LRC ≠ ∅ faça
8:       (i, j) ← retirarAresta(LRC);
9:       se i ∪ j for viável então;
10:          unirAneis(i, j);
11:      fim se;
12:   fim enquanto;
13: fim enquanto;
14: retornar(S);
15: fim randomEdgeBased;
```

No passo 3, S é iniciada com a construção da solução trivial. Nos passos 5 e 6, é criada a LRC com a inserção das arestas virtuais (i, j) dispostas em ordem decrescente do valor de d_{ij} . No passo 8, uma aresta é aleatoriamente retirada da LRC com probabilidade proporcional ao valor de d_{ij} . Caso a demanda total resultante da união do anel i com o anel j seja menor ou igual a B, os anéis são unidos no passo 10.

4.1.3 Função de Aptidão

Uma solução é viável se todos os anéis locais e o anel federal forem viáveis, isto é, se a demanda total de todos os anéis for menor ou igual a B. Uma boa função de aptidão precisa diferenciar soluções viáveis de soluções inviáveis, pois podemos ter uma que viável com o mesmo número de anéis de uma inviável. Como uma solução viável é sempre melhor que uma inviável acrescentamos à aptidão desta a demanda do anel federal, como um fator de penalidade. Desta forma, seja S uma solução qualquer para o SRAP, a função

de aptidão é dada pela equação 4.1.

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

4.1.4 Reprodução

Aplicamos um tipo de *crossover* BPX, ou *bin packing crossover* (FALKNAUER, 1998), exemplificado na figura 4.3 com soluções do SRAP. Sejam duas soluções Pai1 e Pai2 selecionadas através do método da roleta, um anel local de Pai1 é aletoriamente selecionado e inserido em Pai2. Um componente probabilístico foi acrescentado à etapa de seleção do anel de Pai1. Quanto mais localidades atribuídas um anel local tiver, maiores serão suas chances de ser selecionado. Uma vez que todos os anéis locais são viáveis (característica das heurísticas *Random Edge-based* e *Random Edge-based*), é razoável supor que anéis locais com muitas localidades são bons anéis para compor uma nova solução. Depois que o anel é inserido em pai2, as localidades repetidas são excluídas. Isso promove o aparecimento de localidades "perdidas", que ficam isoladas em um anel. Em seguida, realizamos uma busca local na tentativa de melhorar a solução gerada, realocando essas localidades para outros anéis, caso esses não sejam inviabilizados.

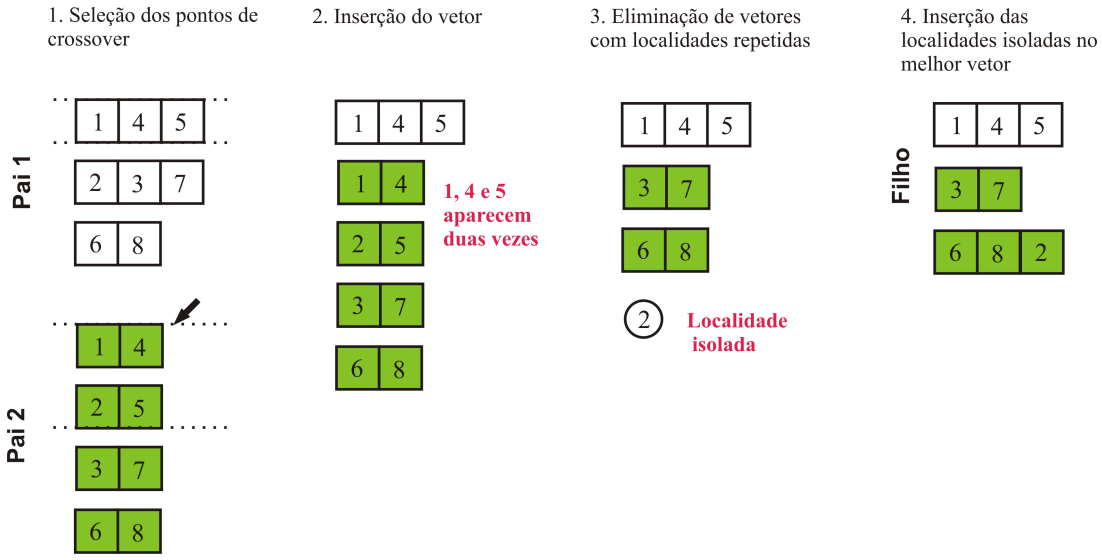

Figura 4.3: Passo a passo do *crossoverBPX* sobre uma solução do SRAP.

Não aplicamos mutação, pois durante os testes, verificamos que não surtiu muita diferença entres as soluções encontradas. O operador de elitismo foi utilizado pois apresentou uma melhora considerável sobre os resultados. Abaixo, segue o algoritmo genético que desenvolvemos para o SRAP.

Algoritmo 5: Algoritmo Genético Aplicado ao SRAP

```
1:  procedimento GeneticoSRAP
2:  pop_atual ← gerar_população_inicial();
3:  avaliar_populacao(pop_atual);
4:      enquanto (condições de paradas não satisfeitas) faça
5:          nova_pop ← ∅;
6:          elitismo(pop_atual, nova_pop);
7:              enquanto (nova_pop não estiver completa) faça
8:                  s1, s2 ← selecionar_pais(pop_atual);
9:                  s1', s2' ← crossover(s1, s2);
10:                 nova_pop ← nova_pop ∪ {s1'} ∪ {s2'};
11:             fim-enquanto;
12:         avaliar_populacao(nova_pop);
13:         pop_atual ← nova_pop;
14:     fim-enquanto;
15: retorna pop_atual.melhor;
16: fim-procedimento;
```

4.2 Algoritmo Memético Puro Aplicado ao SRAP

Esta implementação distingui-se da anterior apenas pelo acréscimo da busca local em seu processo de otimização. Seja S uma solução para o problema, uma vizinhança de busca N é um subconjunto de soluções N(S). Quando não houver nenhuma solução em N(S) melhor do que S então essa é dita um ótimo local. Como não há garantia de que as soluções da população inicial do AG sejam ótimos locais, então, será necessária a aplicação de um procedimento de busca local na tentativa de aperfeiçoar as soluções geradas. O sucesso da busca local está diretamente associado à escolha adequada das vizinhanças de busca, à eficiência das técnicas de busca associadas e às soluções iniciais.

Antes de detalharmos as vizinhaças aplicadas durante a fase de busca, será importante levantarmos uma característica da população gerada pelas heurísticas construtivas desenvolvidas em (GOLDSCHMIDT; LAUGIER; OLINICK, 2003). Essas heurísticas produzem uma população com dois tipos de soluções: viáveis (em que todos os anéis são viáveis, inclusive o anel federal) e inviáveis (com todos os anéis locais viáveis e apenas o anel federal inviável).

Foram testadas as quatro vizinhaças aplicadas por Bastos em (BASTOS, 2005), são elas: N_1, N_2, N_3 e N_4. As duas primeiras, são uma variante das vizinhanças $Pr1$ e $Pr2$ propostas por (MACAMBIRA, 2003). Os procedimentos N_1 e N_2 só são aplicados sobre soluções que possuem todos anéis locais viáveis e anel federal inviável. Além disso, uma troca só é efetuada se a demanda no anel federal for reduzida e os anéis resultantes permanecerem viáveis. Obviamente, esse movimento melhora a avaliação da solução pela função de aptidão. Os procedimentos $Pr1$ e $Pr2$, por sua vez, trabalham com anéis locais viáveis e inviáveis.

4.2.1 Vizinhança N_1

A vizinhança de busca N_1 procura um vértice u pertencente a um anel r e tenta realocá-lo para outro anel t, de modo que a demanda no anel federal seja reduzida sem que nenhum anel local seja inviabilizado. Em outras palavras, 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 $demAF$ seja reduzida. A Figura 4.4 ilustra o movimento e o procedimento associado à vizinhança N_1 encontra-se descrito no Algoritmo 6.

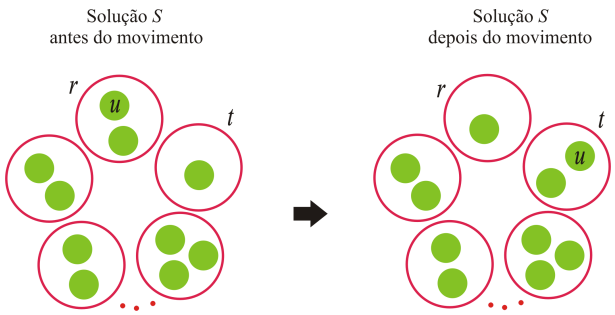

Figura 4.4: Ilustração do movimento de vizinhança N_1.

Algoritmo 6: Procedimento de busca local N_1

1: **procedimento** busca local_$N_1(S)$
2: $u_m \leftarrow$ nulo;
3: $r_m \leftarrow$ nulo;
4: $t_m \leftarrow$ nulo;
5: $dem_{menor} \leftarrow MAX\_INT$;
6: **para** cada anel $r \in S$ **faça**
7: **para** cada localidade $u \in r$ **faça**
8: **para** cada anel $t \in S$, tal que, $r \neq t$ **faça**
9: **se** $(demT_{t \in \{u\}} \leq B)$ e $(demAF_{AposTroca} < dem_{menor})$ **então**
10: $dem_{menor} \leftarrow demAF_{AposTroca}$;
11: $u_m \leftarrow u$;
12: $r_m \leftarrow r$;
13: $t_m \leftarrow t$;
14: **fim se** ;
15: **fim para**;
16: **fim para**;
17: **fim para**;
18: **se** $dem_{menor} < MAX\_INT$ **então** ;
19: $r_m \leftarrow r_m \setminus u_m$;
20: $t_m \leftarrow t_m \cup \{u_m\}$;
21: **fim se**;
22: retornar(S);
23: **fim-procedimento**;

4.2.2 Vizinhança N_2

A vizinhança N_2 procura duas localidades u e v pertencentes a anéis distintos r e t, para realizar um movimento de permutação entre elas. Assim, sejam $u \in r$ e $v \in t$, após a permutação, teremos $v \in r$ e $u \in t$. Novamente, o movimento só é permitido caso os anéis locais permaneçam viáveis e a demanda do anel federal seja reduzida (Figura 4.5). O Algoritmo 7 decscreve o procedimento associado à vizinhança N_2.

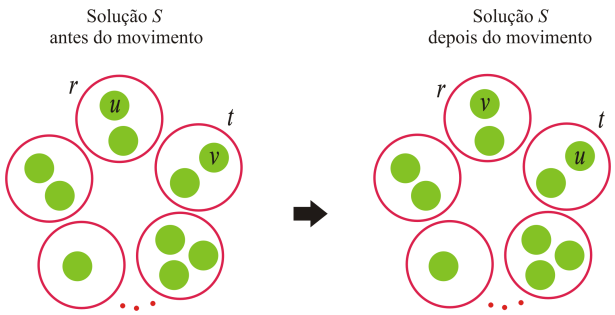

Figura 4.5: Ilustração do movimento de vizinhança N_2.

4.2.3 Vizinhança N_3

A vizinhança N_3 foi introduzida em (BASTOS; OCHI; MACAMBIRA,) e tem como objetivo esvaziar um anel, o de menor demanda total em S, no caso. O procedimento tenta redistribuir as localidades presentes neste anel, digamos r, entre os demais. Essa vizinhança é importante, pois muitas das soluções geradas na população inicial apresentam um ou mais anéis com pouco tráfego que podem ser esvaziados sem inviabilizar outros anéis. A vizinhança N_3 procura o anel t mais promissor a receber o vértice $u \in r$ de modo que t continue viável. O procedimento é repetido até que o anel r seja esvaziado ou até que nenhuma retirada de vértices de r seja possível. Este procedimento tende a reduzir o tráfego no anel federal e mantém ainda viáveis os anéis locais. Note que, o procedimento de busca associado à N_3 não inviabiliza soluções viáveis. No entanto, não há garantia que o anel de menor demanda poderá ser esvaziado através da busca local associada à N_3 (Figura 4.6). O procedimento associado à vizinhança N_3 encontra-se descrito no Algoritmo 8.

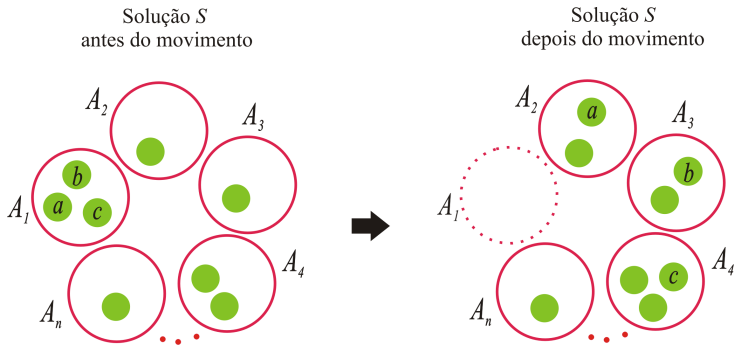

Figura 4.6: Ilustração do movimento de vizinhança N_3.

Algoritmo 7: Procedimento de busca local N_2

1: **procedimento** busca local_$N_2(S)$
2: $u_m \leftarrow$ nulo;
3: $N_m \leftarrow$ nulo;
4: $r_m \leftarrow$ nulo;
5: $t_m \leftarrow$ nulo;
6: $dem_{menor} \leftarrow MAX\_INT$;
7: **para** cada anel $r \in S$ **faça**
8: **para** cada localidade $u \in r$ **faça**
9: **para** cada localidade $v \in t$ **faça**
10: **para** cada anel $t \in S$, tal que, $r \neq t$ **faça**
11: **se** $(demT_{t\in\{u\}} \leq B)$ e $(demT_{r\in\{v\}} \leq B)$ e $(demAF_{AposTroca} < dem_{menor})$ **então**
12: $dem_{menor} \leftarrow demAF_{AposTroca}$;
13: $u_m \leftarrow u$;
14: $N_m \leftarrow u$;
15: $r_m \leftarrow r$;
16: $t_m \leftarrow t$;
17: **fim se** ;
18: **fim para**;
19: **fim para**;
20: **fim para**;
21: **fim para**;
22: **se** $dem_{menor} < MAX\_INT$ **então** ;
23: $r_m \leftarrow r_m \setminus u_m$;
24: $t_m \leftarrow t_m \cup \{u_m\}$;
25: $t_m \leftarrow t_m \cup \{v_m\}$;
26: $r_m \leftarrow r_m \cup \{v_m\}$;
27: **fim se**;
28: retornar(S);
29: **fim-procedimento**;

4.2.4 Vizinhança N_4

A quarta vizinhança, denominada N_4 tem por objetivo aumentar o número de anéis de uma solução, fazendo a redistribuição do tráfego dos anéis, de tal forma que o tráfego no anel federal seja reduzido e a solução seja viabilizada. Isso é necessário, pois para algumas instâncias, as heurísticas construtivas geram soluções inviáveis com menos anéis do que a

Algoritmo 8: Procedimento de busca local N_3

```
1:  procedimento busca local_N3(S)
2:  k ← lerNumeroAneis(S);
3:  a_menor ← selecionarMenorAnel(S);
4:  repita
5:     para cada v ∈ a_menor faça
6:        moveu ← moverParaAnelMaisPromissor(v, S);
7:     fim para;
8:  até (a_menor) = ∅ ou moveu = falso
9:  se lerNumeroAneis(S) < k então
10:   S ← buscaLocal_N3(S);
11:   fim se;
12:   retornar(S);
13:   fim-procedimento;
```

solução ótima. A vizinhança N_4 é fundamental nesses casos, para a descoberta de soluções viáveis, pois as demais vizinhanças de busca, discutidas anteriormente, não possibilitam o aumento do número de anéis numa solução. A aplicação do procedimento N_4 é feita se não existir nenhuma solução viável descoberta e se, após um tempo t de processamento, a freqüência de soluções geradas com um número k_{min} de anéis for elevada. Estas soluções com k_{min} anéis são submetidas à N_4. Uma ilustração do movimento é apresentada na Figura 4.7, e o procedimento, descrito no Algoritmo 9.

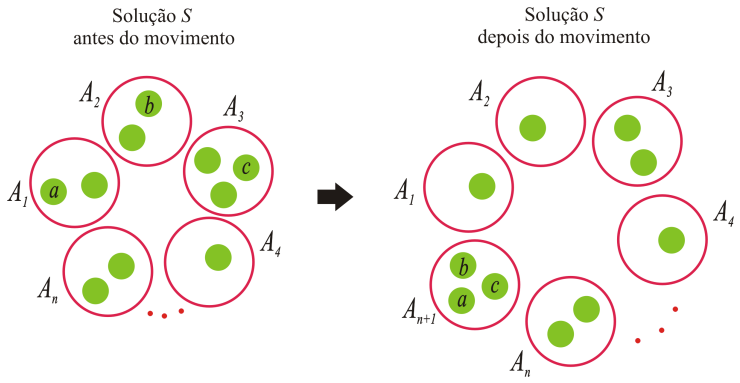

Figura 4.7: Ilustração do movimento de vizinhança N_4.

O Pseudo-código do Algoritmo Memético produzido para resolver o SRAP está descrito no algoritmo 10.

Algoritmo 9: Procedimento de busca local N_4

```
1:  procedimento busca local_N4(S)
2:  a ← criarAnel(S);
3:  S_nova ← S ∪ a;
4:  valorMin ← demAF;
5:  para i = 1, ..., n faça
6:    para j = 1, ..., n | j ∉ a faça
7:      a_j ← lerAnel(j);
8:      moverLocalidade(j, a);
9:      se calcularDemAF() < valorMin então
10:       valorMin ← calcularDemAF()
11:       l ← j;
12:     fim se;
13:     moverLocalidade(j, a_j);
14:   fim para;
15:   moverLocalidade(l, a);
16: fim para;
17: se S_nova for viável então
18:   retornar(S_nova);
19: senão
20:   retornar(S);
21: fim se;
22: fim-procedimento;
```

4.3 Algoritmo Memético Com Vocabulary Building Aplicado ao SRAP

A implementação da técnica *Vocabulary Building* proposta nesse trabalho utiliza a noção de contração de vértices (no nosso caso, localidades) introduzida por Guedes em (GUEDES; ALOISE, 2006), na qual o autor faz uso da idéia proposta por Glover em (GLOVER, 1963), de que a partir de um padrão identificado em diversas soluções é possível chegar a combinações mais complexas e úteis. O mecanismo de combinação dependerá de um conjunto de soluções para a construção dos vocábulos. Estes são trechos tidos como interessantes nas melhores soluções, os quais depois de identificados serão posteriormente posteriormente condensados. A partir desses novos nós, um grafo auxiliar

Algoritmo 10: Algoritmo Memético Aplicado ao SRAP

```
1:  procedimento GeneticoSRAP
2:  pop_atual ← gerar_população_inicial();
3:  otimizar_populacao(pop_atual); //Busca local
4:  avaliar_populacao(pop_atual);
5:      enquanto (condições de paradas não satisfeitas) faça
6:          nova_pop ← ∅;
7:          elitismo(pop_atual, nova_pop);
8:              enquanto (nova_pop não estiver completa) faça
9:                  s1, s2 ← selecionar_pais(pop_atual);
10:                 s1', s2' ← crossover(s1, s2);
11:                 nova_pop ← nova_pop ∪ {s1'} ∪ {s2'};
12:             fim-enquanto;
13:         otimizar_populacao(pop_atual);
14:         avaliar_populacao(nova_pop);
15:         pop_atual ← nova_pop;
16:     fim-enquanto;
17: retorna pop_atual.melhor;
18: fim-procedimento;
```

é criado para ser utlizado em algum momento do processo resolução do problema. Segue abaixo, a descrição do mecanismo proposto:

1. As x melhores soluções são separadas para formar uma população-elite.
2. Sobre essa população-elite, executa-se um procedimento de identificação das localidades que aparecem juntas (no mesmo anel) em y soluções-elite ou mais;
3. Os grupos de localidades identificados são condensados - cada grupo tem suas localidades unidas em um único nó. Esses nós contraídos são chamados de vocábulos;
4. Uma população-auxiliar é então formada, onde um nó pode representar um vocábulo (conjunto de clientes) ou uma localidade;
5. A população-auxiliar é inserida na população atual;
6. Executa-se a busca local nas soluções da população atual;

7. O processo de condensação das localidades é desfeito e a população é atualizada.

A idéia implícita neste procedimento baseia-se na hipótese que uma variável com um determinado valor em várias soluções de boa qualidade tende a possuir esse mesmo valor em uma solução ótima. Por esta razão, identifica-se arestas que estejam presentes em diversas soluções de alta qualidade e faz-se com que elas apareçam nas próximas soluções. A Figura 4.8 traz uma ilustração do processo de identificação e condensação das localidades num conjunto de soluções-elite para a formação de uma população auxiliar composta de soluções com vocábulos.

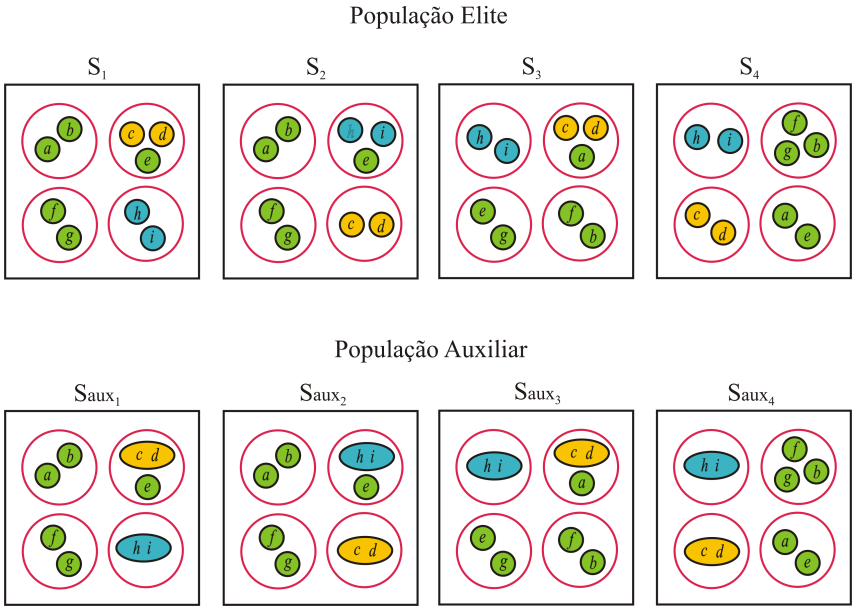

Figura 4.8: Identificação e formação dos Vocábulos.

O procedimento de busca local com aplicação das vizinhanças N_1, N_2, N_3 e N_4 sobre as soluções que contém os vocábulos é exemplificado na Figura 4.9.

O algoritmo 11 traz o procedimento do algoritmo memético hibridizado com *vocabulary building*.

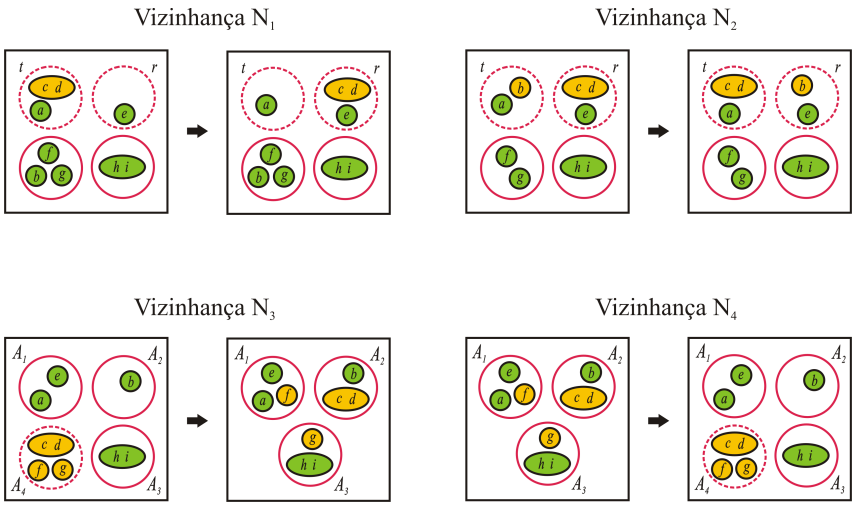

Figura 4.9: Vizinhanças sobre soluções com vocábulos.

Algoritmo 11: Algoritmo Memético Hibridizado Com VB Para o SRAP

```
1:  procedimento MemeticoVBSRAP
2:  pop_atual ← gerar_população_inicial();
3:  avaliar_populacao(pop_atual);
4:  pop_elite ← encontrar_população_elite(pop_atual);
5:  pop_aux ← contracao_de_vertices(pop_elite); //Formar vocábulos
6:  otimizar_populacao(pop_aux); //Busca local
7:  pop_aux ← descontracao_de_vertices(pop_aux); //Desfazer vocábulos
8:  pop_atual ← (pop_atual − pop_elite) ∪ pop_aux;
9:  otimizar_populacao(pop_atual); //Busca local
10:    enquanto (condições de paradas não satisfeitas) faça
11:      nova_pop ← ∅;
12:      elitismo(pop_atual, nova_pop);
13:        enquanto (nova_pop não estiver completa) faça
14:          s1, s2 ← selecionar_pais(pop_atual);
15:          s1', s2' ← crossover(s1, s2);
16:          nova_pop ← nova_pop ∪ {s1'} ∪ {s2'};
17:        fim-enquanto;
18:      avaliar_populacao(pop_atual);
19:      pop_elite ← encontrar_população_elite(pop_atual);
20:      pop_aux ← contracao_de_vertices(pop_elite); //Formar vocábulos
21:      otimizar_populacao(pop_aux); //Busca local
22:      pop_aux ← descontracao_de_vertices(pop_aux); //Desfazer vocábulos
23:      pop_atual ← (pop_atual − pop_elite) ∪ pop_aux;
24:      otimizar_populacao(pop_atual); //Busca local
25:    fim-enquanto;
26: retorna pop_atual.melhor;
27: fim-procedimento;
```

Capítulo 5

Resultados computacionais

Neste capítulo, apresentaremos os resultados computacionais obtidos durante os experimentos realizados neste trabalho. Os algoritmos descritos no Capítulo 4 foram implementados na linguagem C^{++}. Os testes foram executados sobre duas classes de instâncias - C1 e C2 - em computador do tipo PC com processador Core2Duo de 2,2 GHz, de 4MB de cache e 2 GB de RAM no Sistema Operacional Linux Ubuntu 8.04.

Na seção 5.1, serão apresentadas as propriedades das instâncias C1 e C2, bem como, os detalhes de sua geração. Os resultados computacionais estão descritos na seção 5.2.

5.1 Instâncias do Problema

Proposta em (GOLDSCHMIDT; LAUGIER; OLINICK, 2003), a classe C1 é composta de 118 instâncias viáveis. A classe C2 foi proposta por (ARINGHIERI; DELL'AMICO, 2001) e possui 230 instâncias viáveis. As classes C1 e C2 são compostas por dois grupos de instâncias: geométricas e aleatórias. As geométricas apresentam agrupamentos naturais, isto é, localidades mais próximas se comunicam com mais frequência do que com localidades mais distantes. Já as instânticias aleatórias foram geradas a partir de grafos completos com retirada de probabilidade $p \in (0, 1)$ de arestas. Cada grupo é composto por instâncias de capacidades diferentes para os anéis: $B = 155MB/s$ e $B = 622MB/s$. A nomenclatura dessas instâncias está relacionada à classe e à capacidade dos anéis. A tabela 5.1 descreve essa nomeclatura. As instâncias podem assumir os seguintes valores para número de vértices: 15, 25, 30 e 50. Por fim, em (MACAMBIRA, 2003) foi provoda

a otimalidade dessas instâncias e fornecidos seus valores.

Classe	Grupo	Capacidade (B)	Nomeclatura
C1	Geométricas	$155MB/s$	GS (*Geometric Short*)
	Geométricas	$622MB/s$	GL (*Geometric Large*)
	Aleatórias	$155MB/s$	RS (*Random Short*)
	Aleatórias	$622MB/s$	RL (*Random Large*)
C2	Geométricas	$155MB/s$	new.GL (*Geometric Low*)
	Geométricas	$622MB/s$	new.GH (*Geometric High*)
	Aleatórias	$155MB/s$	new.RL (*Random Low*)
	Aleatórias	$622MB/s$	new.RH (*Random High*)

Tabela 5.1: Nomeclatura das intâncias.

5.2 Resultados Computacionais

Na Seção 5.2.1 consta a descrição dos experimentos realizados. Nas Seções 5.2.2 e 5.2.3 serão apresentados os resultados detalhados para cada instância. Serão fornecidos dados para comparação do desempenho entre os algoritmos propostos: Algoritmo Genético, Memético e o Memético hibridizado com a técnica *Vocaburaly Building*.

5.2.1 Experimentos

Uma série de testes foram desenvolvidos para investigar o desempenho dos algoritmos propostos sobre as instâncias das classe C1 e C2. Cada algoritmo foi execultado dez vezes sobre cada instância do problema, e os parâmetros utilizamos foram os mesmos para os três algoritmos. 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. A Tabela 5.2 contém o percentual de soluções ótimas (*) encontradas nos três algoritmos propostos e a respectiva média do tempo computacional.

Os resultados detalhados desses experimentos, são apresentados nas Tabelas 5.3 a 5.13 em onze colunas. As duas primeiras trazem informações sobre a instância do problema que são: nome da instância e o número de anéis da solução ótima (*). As três colunas seguintes trazem respectivamente: o número de anéis da melhor solução viável

obtida pelo AG, o tempo computacional médio (em minutos:segundos:milissegundos) das dez execuções do algoritmo e, o número de execuções em que a solução ótima foi encontrada (#). As três colunas posteriores trazem essas mesmas informações para o AM, e as três últimas para o AM+VB. A última linha de cada tabela, exibe o percentual de ótimos encontrados, a média global de tempo e o número médio de execuções que retornaram valor ótimo para as instâncias da tabela.

Tabela 5.2: Quadro geral

Instância		AG		AM		AM+VB	
Classe	#Inst.	*(%)	Mtempo	*(%)	Mtempo	*(%)	Mtempo
C1	118	98,30	0:04:794	99,15	0:14:317	100	0:15:995
C2	230	88,26	0:11:720	94,34	0:37:567	99,56	0:34:064

5.2.2 Resultados da Classe C1

A Tabela 5.3 exibe os resultados computacionais para as instâncias geométricas da classe C1 com B = 155MB/s. Pode-se observar que os três algoritmos testados foram capazes de encontrar as soluções ótimas para todas as instâncias consideradas. O Algoritmo Memético foi o que apresentou a melhor taxa de convergência, seguido pelo Algoritmo Memético com Vocabulary Building e pelo Algoritmo Genético. Embora esse último tenha sido o procedimento menos eficaz, pode-se observar que ele foi o mais rápido, executando em um tempo médio 4 vezes menor do que o dos demais. O AM+VB exigiu um pouco mais de tempo que o AM, e obteve a menor média de execuções com valor ótimo.

A Tabela 5.4 exibe os resultados computacionais para as instâncias geométricas da classe C1 com B = 622MB/s. Para essas instâncias, têm-se resultados similares aos do subconjunto anterior. Os três algoritmos foram capazes de encontrar as soluções ótimas para todas as instâncias em questão. O Algoritmo Memético foi o que apresentou a melhor taxa de obtenção de ótimos por execução, seguido pelo AM+VB e pelo Algoritmo Genético. Além disso, os dados indicam uma maior eficiência do AG, que executou em tempo bem inferior aos dos demais algoritmos. É interessante notar que essas instâncias foram resolvidas em um tempo computacional médio inferior ao das instâncias da tabela anterior (Tabela 5.3).

Tabela 5.3: Resultados computacionais para as instâncias geométricas da classe C1 com $B = 155MB/s$.

Instância		AG			AM			AM+VB		
Nome	*	Melhor	Tempo	#	Melhor	Tempo	#	Melhor	Tempo	#
GS_15_1	3	3	0:00:503	10	3	0:00:370	10	3	0:01:059	7
GS_15_4	3	3	0:00:479	10	3	0:00:489	10	3	0:00:374	10
GS_15_7	3	3	0:00:492	10	3	0:00:634	10	3	0:00:567	10
GS_15_9	3	3	0:00:482	10	3	0:00:416	10	3	0:00:498	10
GS_25_1	4	4	0:00:588	10	4	0:02:312	10	4	0:02:841	10
GS_25_3	3	3	0:01:539	8	3	0:01:964	10	3	0:02:150	10
GS_25_4	4	4	0:00:239	10	4	0:02:252	10	4	0:03:033	9
GS_25_7	3	3	0:00:365	10	3	0:01:681	10	3	0:01:903	10
GS_25_8	4	4	0:02:203	8	4	0:02:614	10	4	0:02:812	10
GS_30_1	4	4	0:01:284	10	4	0:04:614	10	4	0:08:964	7
GS_30_2	4	4	0:01:970	10	4	0:05:009	10	4	0:05:518	10
GS_30_3	4	4	0:01:715	10	4	0:04:755	10	4	0:10:569	6
GS_30_4	4	4	0:01:768	10	4	0:04:948	10	4	0:11:378	6
GS_30_5	3	3	0:06:091	1	3	0:16:324	5	3	0:04:787	10
GS_30_6	4	4	0:01:620	10	4	0:04:422	10	4	0:10:086	7
GS_30_7	4	4	0:01:690	10	4	0:05:268	10	4	0:06:686	10
GS_30_8	4	4	0:01:522	10	4	0:04:784	10	4	0:05:252	10
GS_30_9	4	4	0:01:685	10	4	0:04:203	10	4	0:09:447	7
GS_50_1	5	5	0:14:605	8	5	0:43:722	10	5	0:47:978	10
GS_50_4	6	6	0:17:668	7	6	0:46:068	10	6	1:07:907	8
GS_50_7	5	5	0:16:536	9	5	0:45:732	10	5	0:49:245	10
GS_50_8	5	5	0:21:855	2	5	2:01:201	5	5	1:26:363	10
GS_50_9	4	4	0:17:224	1	4	0:54:416	6	4	1:02:168	10
Resumo		*(%)	MTempo	M#	*(%)	MTempo	M#	*(%)	MTempo	M#
		100	0:04:961	8,43	100	0:16:443	9,39	100	0:17:460	9,00

Tabela 5.4: Resultados computacionais para as instâncias geométricas da classe C1 com $B = 622MB/s$.

Instância		AG			AM			AM+VB		
Nome	*	Melhor	Tempo	#	Melhor	Tempo	#	Melhor	Tempo	#
GL_15_10	2	2	0:00:478	10	2	0:00:506	10	2	0:00:462	10
GL_15_1	3	3	0:00:488	10	3	0:00:554	10	3	0:00:526	10
GL_15_2	3	3	0:00:536	10	3	0:02:716	9	3	0:00:488	10
GL_15_3	2	2	0:00:431	10	2	0:00:515	10	2	0:00:512	10
GL_15_6	2	2	0:00:442	10	2	0:00:413	10	2	0:00:454	10
GL_15_7	2	2	0:00:490	10	2	0:00:503	10	2	0:00:590	10
GL_15_8	3	3	0:00:443	10	3	0:00:506	10	3	0:00:479	10
GL_15_9	3	3	0:00:531	10	3	0:00:428	10	3	0:00:452	10
GL_25_1	3	3	0:00:312	10	3	0:01:535	10	3	0:03:174	8
GL_25_2	2	2	0:00:142	10	2	0:01:312	10	2	0:01:594	10
GL_25_3	2	2	0:00:614	10	2	0:01:451	10	2	0:01:818	10
GL_25_4	3	3	0:00:701	10	3	0:01:798	10	3	0:02:145	10
GL_25_7	4	4	0:00:737	10	4	0:02:171	10	4	0:03:251	10
GL_25_8	3	3	0:00:549	10	3	0:01:810	10	3	0:02:088	10
GL_25_9	3	3	0:00:127	10	3	0:01:773	10	3	0:02:134	10
GL_30_10	3	3	0:01:373	10	3	0:03:095	10	3	0:03:569	10
GL_30_1	3	3	0:01:344	10	3	0:03:606	10	3	0:04:090	10
GL_30_2	4	4	0:01:681	10	4	0:05:075	10	4	0:07:186	9
GL_30_3	4	4	0:01:697	10	4	0:05:240	10	4	0:06:451	10
GL_30_4	3	3	0:01:790	10	3	0:04:292	10	3	0:07:947	6
GL_30_5	4	4	0:01:653	10	4	0:04:729	10	4	0:07:412	9
GL_30_9	4	4	0:01:299	10	4	0:04:379	10	4	0:08:726	7
GL_50_1	5	5	0:13:473	10	5	0:44:929	10	5	0:49:274	10
GL_50_4	4	4	0:18:512	1	4	1:47:100	2	4	0:52:056	9
GL_50_5	5	5	0:11:890	10	5	0:42:500	10	5	1:04:953	8
GL_50_6	5	5	0:16:794	6	5	0:48:860	10	5	0:49:484	10
GL_50_7	5	5	0:12:440	10	5	0:44:173	10	5	0:48:905	10
Resumo		*(%)	MTempo	M#	*(%)	MTempo	M#	*(%)	MTempo	M#
		100	0:03:369	9,51	100	0:12:443	9,66	100	0:12:230	9,48

A Tabela 5.5 mostra os resultados computacionais para as instâncias aleatórias da classe C1 com $B = 155MB/s$. Assim como nos dois subconjuntos anteriores, os algoritmos testados foram capazes de encontrar as soluções ótimas para todas as instâncias consideradas. O Algoritmo Memético se apresentou mais uma vez como a melhor opção. Embora o AM+VB tenha apresentado a mesma média de obtenção do ótimo por execução que o AM, a sua execução demandou um tempo computacional superior aos dos outros algoritmos. Para essas instâncias, o AG conseguiu uma média não muito inferior às dos demais e continuou apresentando um tempo de processamento bem inferior.

Finalmente, a Tabela 5.6 exibe os resultados computacionais para as instâncias aleatórias da classe C1 com $B = 622MB/s$. Diferentemente dos conjuntos de instâncias anteriores, houve casos em que as soluções ótimas não foram encontradas em nenhuma das 10 execuções. No caso do AG, não foi possível obter as configurações ótimas para as instâncias RL_25_5 e RL_50_3, mesmo após um alto tempo de processamento. Na coluna do Algoritmo Memético, vê-se a instância RL_50_3 não teve seu ótimo global encontrado. Nessa ocasião, verifica-se que foi consumido um tempo computacional bem acima da média. O Algoritmo Memético com *Vocabulary Building* foi capaz de achar os ótimos globais de todas as instâncias consideradas, tendo exibido também a maior média de obtenção de ótimos por execução. Mais uma vez, o AM+VB apresentou o pior tempo médio de execução, embora esse não tenha sido muito maior do que o do Algoritmo Memético.

5.2.3 Resultados da Classe C2

Os resultados da classe C2 são apresentados nas Tabelas 5.7 a 5.13. Foram encontrados 100% dos ótimos. O AM obteve valores ótimos nas 10 excuções de todas as instâncias.

As Tabelas 5.7 e 5.8 trazem os resultados dos testes para as instâncias geométricas da classe C2 com $B = 155MB/s$. A partir dos resultados, observa-se que o Algoritmo Genético não funcionou muito bem para esse subconjunto de instâncias. Esse procedimento continuou sendo o mais rápido dos três, mas não conseguiu encontrar as soluções ótimas para várias das instâncias testadas - todas elas contendo 50 localidades. Em alguns casos, o AG não conseguiu nem mesmo encontrar soluções viáveis (new.GL_50_6.1, new.GL_50_6.7 e new.GL_50_10.6). Já para as instâncias new.GL_50_6.5,

Tabela 5.5: Resultados computacionais para as instâncias aleatórias da classe C1 com $B = 155MB/s$.

Instância		AG			AM			AM+VB		
Nome	*	Melhor	Tempo	#	Melhor	Tempo	#	Melhor	Tempo	#
RS_15_10	3	3	0:00:603	10	3	0:00:583	10	3	0:00:633	10
RS_15_1	3	3	0:00:478	10	3	0:00:432	10	3	0:00:463	10
RS_15_3	2	2	0:00:499	10	2	0:00:563	10	2	0:00:473	10
RS_15_4	3	3	0:00:541	10	3	0:00:497	10	3	0:00:452	10
RS_15_6	3	3	0:00:496	10	3	0:00:474	10	3	0:00:420	10
RS_15_8	3	3	0:00:444	10	3	0:00:508	10	3	0:00:600	10
RS_15_9	3	3	0:00:481	10	3	0:00:444	10	3	0:00:426	10
RS_25_10	4	4	0:00:603	10	4	0:01:914	10	4	0:02:550	10
RS_25_1	4	4	0:05:638	7	4	0:02:216	10	4	0:03:699	10
RS_25_2	3	3	0:00:747	10	3	0:01:784	10	3	0:03:078	10
RS_25_4	4	4	0:01:070	10	4	0:01:903	10	4	0:02:479	10
RS_25_5	4	4	0:00:619	10	4	0:01:843	10	4	0:02:023	10
RS_25_6	4	4	0:00:679	10	4	0:02:083	10	4	0:02:788	10
RS_25_7	4	4	0:00:661	10	4	0:02:024	10	4	0:03:144	10
RS_25_8	3	3	0:00:893	10	3	0:01:503	10	3	0:01:851	10
RS_25_9	4	4	0:00:750	10	4	0:02:007	10	4	0:02:398	10
RS_30_10	4	4	0:01:741	10	4	0:04:677	10	4	0:05:664	10
RS_30_1	3	3	0:01:789	10	3	0:03:386	10	3	0:05:243	10
RS_30_3	4	4	0:01:899	10	4	0:04:906	10	4	0:05:804	10
RS_30_4	4	4	0:02:161	10	4	0:05:485	10	4	0:05:933	10
RS_30_5	4	4	0:01:442	10	4	0:03:873	10	4	0:04:303	10
RS_30_6	4	4	0:07:596	7	4	0:21:047	7	4	0:15:943	6
RS_30_7	3	3	0:01:634	10	3	0:03:965	10	3	0:04:590	10
RS_30_8	4	4	0:01:391	10	4	0:04:028	10	4	0:04:795	10
RS_50_10	4	4	0:15:147	9	4	0:41:212	10	4	1:01:048	10
RS_50_1	5	5	0:14:392	10	5	0:37:218	10	5	1:05:423	10
RS_50_3	4	4	0:15:167	8	4	0:59:053	8	4	1:13:524	9
RS_50_4	4	4	0:13:953	10	4	0:52:126	9	4	1:17:807	9
RS_50_5	4	4	0:14:879	8	4	0:31:460	10	4	0:36:320	10
RS_50_6	4	4	0:19:715	5	4	0:31:411	10	4	0:47:425	10
RS_50_7	5	5	0:15:046	9	5	0:38:285	10	5	0:49:593	10
Resumo		*(%)	MTempo	M#	*(%)	MTempo	M#	*(%)	MTempo	M#
		100	0:04:617	9,45	100	0:11:706	9,80	100	0:15:835	9,80

Tabela 5.6: Resultados computacionais para as instâncias aleatórias da classe C1 com $B = 622MB/s$.

Instância		AG			AM			AM+VB		
Nome	*	Melhor	Tempo	#	Melhor	Tempo	#	Melhor	Tempo	#
RL_15_1	3	3	0:00:516	10	3	0:00:477	10	3	0:00:526	10
RL_15_3	3	3	0:00:473	10	3	0:00:538	10	3	0:00:532	10
RL_15_4	2	2	0:00:573	10	2	0:00:454	10	2	0:00:434	10
RL_15_5	3	3	0:00:493	10	3	0:00:537	10	3	0:00:498	10
RL_15_6	3	3	0:00:440	10	3	0:02:600	9	3	0:00:947	8
RL_15_7	2	2	0:00:546	10	2	0:00:477	10	2	0:00:460	10
RL_15_8	2	2	0:00:540	10	2	0:00:562	10	2	0:00:447	10
RL_15_9	3	3	0:00:558	10	3	0:00:553	10	3	0:00:552	10
RL_25_10	3	3	0:00:799	10	3	0:06:730	8	3	0:01:851	10
RL_25_1	3	3	0:00:830	10	3	0:02:064	10	3	0:03:452	10
RL_25_2	3	3	0:00:810	10	3	0:01:412	10	3	0:01:886	10
RL_25_3	3	3	0:00:517	10	3	0:01:329	10	3	0:01:951	10
RL_25_4	3	3	0:01:952	8	3	0:02:461	10	3	0:02:542	10
RL_25_5	4	-	0:22:859	0	4	0:27:195	4	4	0:07:852	6
RL_25_6	4	4	0:00:339	10	4	0:19:301	10	4	0:12:947	2
RL_25_7	3	3	0:00:579	10	3	0:01:294	10	3	0:01:786	10
RL_25_8	4	4	0:02:063	9	4	0:01:855	10	4	0:02:225	10
RL_25_9	2	2	0:00:328	10	2	0:01:103	10	2	0:01:737	10
RL_30_10	3	3	0:04:267	4	3	0:04:866	10	3	0:06:847	10
RL_30_1	3	3	0:01:275	10	3	0:02:953	10	3	0:03:851	10
RL_30_2	4	4	0:01:324	10	4	0:04:184	10	4	0:05:128	10
RL_30_3	3	3	0:01:276	10	3	0:03:117	10	3	0:04:175	10
RL_30_4	3	3	0:01:517	10	3	0:03:743	10	3	0:04:879	10
RL_30_6	4	4	0:04:869	9	4	0:04:251	10	4	0:05:819	10
RL_30_7	4	4	0:01:596	10	4	0:04:093	10	4	0:05:203	10
RL_30_8	3	3	0:01:286	10	3	0:03:694	10	3	0:04:696	10
RL_30_9	3	3	0:01:448	10	3	0:03:794	10	3	0:04:615	10
RL_50_10	4	4	0:14:072	10	4	0:52:398	9	4	1:03:480	10
RL_50_1	4	4	0:12:513	10	4	0:35:578	10	4	0:57:798	10
RL_50_2	3	3	0:15:950	5	3	0:25:603	10	3	0:37:859	10
RL_50_3	4	5	0:23:317	0	5	2:31:040	0	4	1:30:529	9
RL_50_4	4	4	0:22:144	3	4	1:24:163	8	4	1:26:298	9
RL_50_5	3	3	0:13:941	9	3	0:26:294	10	3	0:45:453	10
RL_50_6	3	3	0:14:340	9	3	0:25:281	10	3	0:37:047	10
RL_50_7	5	5	0:22:600	3	5	0:46:109	10	5	1:08:607	10
RL_50_8	4	4	0:12:349	10	4	0:31:750	10	4	0:45:123	10
RL_50_9	4	4	0:12:232	10	4	0:28:572	10	4	0:44:722	10
Resumo		*(%)	MTempo	M#	*(%)	MTempo	M#	*(%)	MTempo	M#
		94,59	0:05:879	8,62	97,29	0:16:552	9,40	100	0:17:966	9,56

new.GL_50_10.1, new.GL_50_10.2, new.GL_50_10.4, new.GL_50_10.5 e new.GL_50_10.8, o AG encontrou soluções viáveis mas não foi capaz de obter as soluções ótimas. Para o Algoritmo Memético, verifica-se que não foram obtidas as soluções ótimas para as instâncias new.GL_50_6.1, new.GL_50_6.7, new.GL_50_10.2, new.GL_50_10.4, new.GL_50_10.6 e, new.GL_50_10.8. Esse procedimento sempre foi capaz de achar soluções viáveis para as instâncias em questão, e os tempos de execução reportados foram os mais altos dos três algoritmos. O AM+VB não apresentou a maior média de obtenção do ótimo por execução, mas foi capaz de encontrar as soluções ótimas para todas as instâncias em apreço, em um tempo de processamento médio menor que o Algoritmo Memético puro.

As Tabelas 5.9 e 5.10 trazem os resultados dos testes para as instâncias geométricas da classe C2 com $B = 622MB/s$. Da mesma forma que nos casos anteriores, o Algoritmo Genético apresentou o menor tempo de processamento dos três procedimentos. Contudo, o AG não foi capaz de encontrar soluções viáveis para 5 instâncias e também não achou soluções ótimas para 10 outras. Além disso, obteve uma taxa de convergência bastante baixa. O Algoritmo Memético encontrou soluções viáveis para todas as instâncias, mas não conseguiu achar as soluções ótimas para 4 delas. Também é possível ver que o seu tempo médio de execução foi bastante alto. O Algoritmo Memético com *Vocabulary Building* obteve uma alta taxa de obtenção de ótimos por execução e conseguiu encontrar as soluções ótimas para todas as instâncias - ao custo de um maior tempo de processamento.

As Tabelas 5.11 e 5.12 trazem os resultados dos testes para as instâncias aleatórias da classe C2 com $B = 155MB/s$. Novamente, o Algoritmo Genético apresentou o melhor tempo de processamento, ao custo de uma baixa taxa de convergência. O AG não foi capaz de encontrar soluções viáveis para as instâncias new.RL_50_8.7 e new.RL_50_8.9, e não conseguiu encontrar a solução ótima para new.RL_50_2.3. O Algoritmo Memético apresentou o pior dos três tempos médios de execução, que foi bastante alto. O AM deixou de obter soluções viáveis para new.RL_50_2.4 e a solução ótima para new.RL_50_2.3. No caso do Algoritmo Memético com *Vocabulary Building*, pode-se perceber que apenas a instância new.RL_50_2.3 não teve sua solução ótima encontrada. O tempo computacional desse procedimento também foi alto, mas se mostrou inferior ao do AM. Também é interessante perceber que o AM+VB obteve a maior taxa de obtenção de ótimos.

Tabela 5.7: Resultados computacionais para as instâncias geométricas da classe C2 com $B = 155MB/s$.

Instância		AG			AM			AM+VB		
Nome	*	Melhor	Tempo	#	Melhor	Tempo	#	Melhor	Tempo	#
new.GL_15_3.10	4	4	0:00:516	10	4	0:0:508	10	4	0:02:709	4
new.GL_15_3.1	4	4	0:00:508	10	4	0:0:557	10	4	0:02:158	5
new.GL_15_3.2	4	4	0:00:451	10	4	0:0:481	10	4	0:04:048	9
new.GL_15_3.3	4	4	0:00:526	10	4	0:0:484	10	4	0:01:311	8
new.GL_15_3.4	4	4	0:00:538	10	4	0:0:548	10	4	0:04:177	9
new.GL_15_3.5	4	4	0:00:502	10	4	0:0:430	10	4	0:03:374	2
new.GL_15_3.6	4	4	0:00:462	10	4	0:0:480	10	4	0:02:588	4
new.GL_15_3.7	4	4	0:00:417	10	4	0:0:407	10	4	0:03:315	2
new.GL_15_3.8	4	4	0:00:538	10	4	0:0:492	10	4	0:01:839	6
new.GL_15_3.9	4	4	0:00:452	10	4	0:0:469	10	4	0:03:256	2
new.GL_15_6.10	3	3	0:00:500	10	3	0:0:487	10	3	0:00:470	10
new.GL_15_6.1	3	3	0:00:505	10	3	0:0:453	10	3	0:00:420	10
new.GL_15_6.2	3	3	0:00:449	10	3	0:0:523	10	3	0:00:387	10
new.GL_15_6.3	3	3	0:00:506	10	3	0:0:506	10	3	0:00:427	10
new.GL_15_6.4	3	3	0:00:541	10	3	0:0:443	10	3	0:00:377	10
new.GL_15_6.5	3	3	0:00:553	10	3	0:0:533	10	3	0:00:539	10
new.GL_15_6.6	3	3	0:00:516	10	3	0:0:456	10	3	0:00:567	10
new.GL_15_6.7	3	3	0:00:479	10	3	0:0:459	10	3	0:00:482	10
new.GL_15_6.8	3	3	0:03:946	9	3	0:0:577	10	3	0:00:523	10
new.GL_15_6.9	3	3	0:00:552	10	3	0:0:450	10	3	0:00:622	10
new.GL_25_9.10	4	4	0:00:975	10	4	0:2:352	10	4	0:04:030	10
new.GL_25_9.1	4	4	0:03:729	7	4	0:2:166	10	4	0:03:856	10
new.GL_25_9.2	4	4	0:01:464	9	4	0:2:380	10	4	0:03:715	10
new.GL_25_9.3	4	4	0:02:346	9	4	0:2:163	10	4	0:03:524	10
new.GL_25_9.4	4	4	0:03:736	9	4	0:2:233	10	4	0:04:004	10
new.GL_25_9.5	4	4	0:01:299	10	4	0:2:271	10	4	0:03:742	10
new.GL_25_9.6	4	4	0:01:110	10	4	0:2:397	10	4	0:03:824	10
new.GL_25_9.7	4	4	0:01:193	10	4	0:2:250	10	4	0:03:793	10
new.GL_25_9.8	4	4	0:01:258	10	4	0:2:130	10	4	0:03:875	10
new.GL_25_9.9	4	4	0:01:010	10	4	0:2:159	10	4	0:03:699	10
new.GL_25_10.10	5	5	0:00:384	10	5	0:2:632	10	5	0:12:376	3
new.GL_25_10.1	5	5	0:00:481	10	5	0:2:648	10	5	0:09:162	4
new.GL_25_10.2	5	5	0:00:790	10	5	0:2:557	10	5	0:10:680	3
new.GL_25_10.3	5	5	0:00:393	10	5	0:2:535	10	5	0:11:207	3
new.GL_25_10.4	5	5	0:00:759	10	5	0:2:725	10	5	0:11:993	2
new.GL_25_10.5	5	5	0:00:652	10	5	0:2:489	10	5	0:13:487	1
new.GL_25_10.6	5	5	0:00:804	10	5	0:2:569	10	5	0:14:180	9
new.GL_25_10.7	5	5	0:00:642	10	5	0:2:615	10	5	0:07:847	6
new.GL_25_10.8	5	5	0:00:333	10	5	0:2:507	10	5	0:09:711	3
new.GL_25_10.9	5	5	0:00:462	10	5	0:2:712	10	5	0:11:721	2

Tabela 5.8: Resultados computacionais para as instâncias geométricas da classe C2 com $B = 155MB/s$ (Continuação).

Instância		AG			AM			AM+VB		
Nome	*	Melhor	Tempo	#	Melhor	Tempo	#	Melhor	Tempo	#
new.GL_30_10.10	5	5	0:01:808	10	5	0:05:893	10	5	0:08:324	9
new.GL_30_10.1	5	5	0:01:847	10	5	0:05:894	10	5	0:06:761	10
new.GL_30_10.2	5	5	0:01:146	10	5	0:05:811	10	5	0:06:426	10
new.GL_30_10.3	5	5	0:02:137	10	5	0:05:914	10	5	0:06:547	10
new.GL_30_10.4	5	5	0:01:913	10	5	0:06:189	10	5	0:06:896	10
new.GL_30_10.5	5	5	0:04:045	9	5	0:05:867	10	5	0:08:137	9
new.GL_30_10.6	5	5	0:01:616	10	5	0:06:220	10	5	0:08:857	9
new.GL_30_10.7	5	5	0:01:742	10	5	0:06:000	10	5	0:08:454	9
new.GL_30_10.8	5	5	0:01:306	10	5	0:05:787	10	5	0:08:152	9
new.GL_30_10.9	5	5	0:01:730	10	5	0:05:880	10	5	0:06:465	10
new.GL_50_6.10	6	6	0:21:885	6	6	1:05:470	9	5	0:58:479	10
new.GL_50_6.1	6	-	0:29:663	0	7	2:49:366	0	6	1:01:735	10
new.GL_50_6.2	6	6	0:36:703	1	6	1:42:120	8	6	1:16:400	9
new.GL_50_6.3	6	6	0:32:725	3	6	1:35:080	7	6	1:08:812	9
new.GL_50_6.4	6	6	0:34:869	1	6	1:27:645	8	6	1:03:490	10
new.GL_50_6.5	6	7	0:39:413	0	6	2:50:705	1	6	1:31:210	8
new.GL_50_6.6	6	6	0:22:526	6	6	1:17:991	9	6	1:10:449	9
new.GL_50_6.7	6	-	0:41:328	0	7	2:48:851	0	6	1:39:507	7
new.GL_50_6.8	6	6	0:25:829	4	6	1:01:985	9	6	0:57:491	10
new.GL_50_6.9	6	6	0:21:933	5	6	0:52:795	10	6	0:59:118	10
new.GL_50_10.10	6	6	0:20:604	4	6	0:48:183	10	6	1:42:636	9
new.GL_50_10.1	5	6	0:24:317	0	5	2:04:776	2	5	1:32:500	8
new.GL_50_10.2	5	6	0:25:298	0	6	2:22:524	0	5	1:15:787	10
new.GL_50_10.3	6	6	0:25:906	1	6	0:52:383	10	6	1:23:624	9
new.GL_50_10.4	5	6	0:23:528	0	6	2:17:206	0	5	1:29:138	8
new.GL_50_10.5	5	6	0:23:261	0	5	1:18:740	9	5	1:31:531	8
new.GL_50_10.6	5	-	0:25:492	0	6	2:35:533	0	5	1:26:924	9
new.GL_50_10.7	6	6	0:27:722	1	6	0:44:235	10	6	1:10:002	10
new.GL_50_10.8	5	6	0:23:970	0	6	2:26:582	0	5	1:18:438	10
new.GL_50_10.9	6	6	0:22:228	4	6	0:44:783	10	6	1:08:164	10
Resumo		*(%)	MTempo	M#	*(%)	MTempo	M#	*(%)	MTempo	M#
		87,14	0:08:653	7,54	91,42	0:30:637	8,74	100	0:25:778	8,07

Tabela 5.9: Resultados computacionais para as instâncias geométricas da classe C2 com $B = 622MB/s$.

Instância		AG			AM			AM+VB		
Nome	*	Melhor	Tempo	#	Melhor	Tempo	#	Melhor	Tempo	#
new.GH_25_5.10	4	4	0:06:318	8	4	0:01:962	10	4	0:02:786	10
new.GH_25_5.1	4	4	0:00:694	10	4	0:02:255	10	4	0:03:112	10
new.GH_25_5.2	4	4	0:00:595	10	4	0:02:155	10	4	0:02:610	10
new.GH_25_5.3	4	4	0:01:114	10	4	0:02:526	10	4	0:02:570	10
new.GH_25_5.4	4	4	0:00:656	10	4	0:02:096	10	4	0:02:451	10
new.GH_25_5.5	4	4	0:00:849	10	4	0:02:272	10	4	0:02:391	10
new.GH_25_5.6	4	4	0:00:769	10	4	0:02:156	10	4	0:02:871	10
new.GH_25_5.7	4	4	0:00:956	10	4	0:01:971	10	4	0:02:333	10
new.GH_25_5.8	4	4	0:00:840	10	4	0:02:122	10	4	0:02:755	10
new.GH_25_5.9	4	4	0:13:813	6	4	0:02:086	10	4	0:02:721	10
new.GH_30_8.10	4	4	0:01:812	10	4	0:04:479	10	4	0:06:869	10
new.GH_30_8.1	4	4	0:06:636	7	4	0:04:533	10	4	0:04:984	10
new.GH_30_8.2	4	4	0:05:431	7	4	0:05:140	10	4	0:08:781	10
new.GH_30_8.3	4	-	0:24:410	0	5	0:24:025	6	4	0:13:907	7
new.GH_30_8.4	4	4	0:01:989	10	4	0:04:484	10	4	0:05:217	10
new.GH_30_8.5	4	4	0:01:884	10	4	0:04:613	10	4	0:04:949	10
new.GH_30_8.6	4	4	0:01:906	10	4	0:04:745	10	4	0:07:857	10
new.GH_30_8.7	4	4	0:02:974	9	4	0:05:010	10	4	0:08:255	10
new.GH_30_8.8	4	4	0:04:846	9	4	0:04:709	10	4	0:07:401	10
new.GH_30_8.9	4	4	0:01:810	10	4	0:04:331	10	4	0:06:599	10
new.GH_50_2.10	6	6	0:17:989	9	6	0:52:551	10	6	0:56:395	10
new.GH_50_2.1	6	6	0:16:947	10	6	0:52:596	10	6	0:54:879	10
new.GH_50_2.2	6	6	0:21:255	4	6	0:52:885	10	6	0:58:313	10
new.GH_50_2.3	6	6	0:15:244	10	6	0:53:257	10	6	0:55:698	10
new.GH_50_2.4	6	6	0:18:866	9	6	0:51:901	10	6	0:55:821	10
new.GH_50_2.5	6	6	0:14:962	10	6	0:53:110	10	6	0:56:020	10
new.GH_50_2.6	6	6	0:18:194	9	6	0:51:620	10	6	1:07:098	9
new.GH_50_2.7	6	6	0:18:507	8	6	0:53:372	10	6	0:55:202	10
new.GH_50_2.8	6	6	0:15:864	10	6	0:53:140	10	6	0:56:922	10
new.GH_50_2.9	6	6	0:14:372	10	6	0:53:838	10	6	0:57:391	10
new.GH_50_3.10	6	6	0:18:797	5	6	0:47:372	10	6	1:26:068	10
new.GH_50_3.1	5	6	0:24:560	0	6	2:14:229	0	5	3:03:330	1
new.GH_50_3.2	5	5	0:21:397	3	5	2:01:418	1	5	1:32:996	10
new.GH_50_3.3	5	5	0:16:116	9	5	0:55:753	9	5	1:18:058	10
new.GH_50_3.4	5	6	0:22:707	0	6	2:08:231	0	5	2:37:035	3
new.GH_50_3.5	5	6	0:22:331	0	5	1:48:478	4	5	2:43:597	3
new.GH_50_3.6	5	5	0:20:149	5	5	1:08:065	8	5	1:26:645	10
new.GH_50_3.7	5	5	0:15:673	8	5	0:46:640	10	5	1:17:951	10
new.GH_50_3.8	5	5	0:14:246	9	5	0:53:353	10	5	1:17:614	10
new.GH_50_3.9	5	5	0:18:342	3	5	1:04:869	9	5	1:18:184	10

Tabela 5.10: Resultados computacionais para as instâncias geométricas da classe C2 com $B = 622MB/s$ (Continação).

Instância		AG			AM			AM+VB		
Nome	*	Melhor	Tempo	#	Melhor	Tempo	#	Melhor	Tempo	#
new.GH_50_8.10	6	6	0:52:623	3	6	0:48:227	10	6	1:03:089	9
new.GH_50_8.1	6	6	0:39:745	5	6	0:49:422	10	6	0:52:612	10
new.GH_50_8.2	6	6	0:14:001	10	6	0:47:734	10	6	0:51:749	10
new.GH_50_8.3	6	6	0:13:779	10	6	0:48:897	10	6	0:52:434	10
new.GH_50_8.4	6	6	0:14:462	10	6	0:48:313	10	6	0:51:827	10
new.GH_50_8.5	6	6	0:40:836	4	6	0:53:204	10	6	1:01:897	9
new.GH_50_8.6	6	6	0:18:753	9	6	0:48:748	10	6	0:52:661	10
new.GH_50_8.7	6	-	0:34:982	0	6	0:54:612	10	6	0:54:838	10
new.GH_50_8.8	6	6	0:32:021	5	6	0:48:011	10	6	0:52:124	10
new.GH_50_8.9	6	6	0:16:573	9	6	0:51:467	10	6	0:55:840	10
new.GH_50_9.10	5	5	0:18:097	9	5	0:46:311	10	5	1:12:042	10
new.GH_50_9.1	5	5	0:24:647	1	5	1:28:195	7	5	1:06:372	10
new.GH_50_9.2	5	5	0:16:416	9	5	0:45:022	10	5	1:19:964	10
new.GH_50_9.3	5	6	0:25:313	0	5	2:09:501	2	5	1:36:068	8
new.GH_50_9.4	5	-	0:26:310	0	6	2:19:224	0	5	1:59:517	6
new.GH_50_9.5	5	5	0:19:108	5	5	0:48:062	10	5	0:56:274	10
new.GH_50_9.6	5	6	0:26:254	0	5	1:21:469	8	5	1:35:047	9
new.GH_50_9.7	5	-	0:27:662	0	5	2:21:243	1	5	2:48:819	2
new.GH_50_9.8	5	5	0:24:230	4	5	1:29:899	6	5	1:39:203	9
new.GH_50_9.9	5	5	0:19:251	6	5	0:52:256	10	5	0:56:504	10
new.GH_50_10.10	5	5	0:31:505	3	5	0:45:948	10	5	1:05:136	10
new.GH_50_10.1	5	5	0:31:277	1	5	1:08:635	8	5	1:01:315	10
new.GH_50_10.2	5	6	0:38:669	0	5	1:32:732	7	5	0:50:679	10
new.GH_50_10.3	5	6	0:35:909	0	5	1:05:734	9	5	0:52:563	10
new.GH_50_10.4	5	5	0:28:431	2	5	0:52:473	10	5	0:58:740	10
new.GH_50_10.5	5	6	0:25:682	0	5	2:30:319	2	5	1:10:633	10
new.GH_50_10.6	5	-	0:25:130	0	5	0:58:783	9	5	0:50:484	10
new.GH_50_10.7	5	7	0:35:259	0	6	3:09:634	0	5	1:02:043	9
new.GH_50_10.8	5	5	0:29:279	1	5	0:59:347	10	5	0:57:072	9
new.GH_50_10.9	5	6	0:27:912	0	5	1:07:310	8	5	0:58:723	10
Resumo		*(%)	MTempo	M#	*(%)	MTempo	M#	*(%)	MTempo	M#
		78,57	0:18:013	5,90	94,28	0:52:215	8,48	100	0:54:327	9,32

Tabela 5.11: Resultados computacionais para as instâncias aleatórias da classe C2 com $B = 155MB/s$.

Instância		AG			AM			AM+VB		
Nome	*	Melhor	Tempo	#	Melhor	Tempo	#	Melhor	Tempo	#
new.RL_15_2.10	3	3	0:00:737	10	3	0:00:429	10	3	0:00:507	10
new.RL_15_2.1	3	3	0:00:375	10	3	0:00:552	10	3	0:00:479	10
new.RL_15_2.2	4	4	0:00:489	10	4	0:00:479	10	4	0:00:515	10
new.RL_15_2.3	3	3	0:00:545	10	3	0:00:436	10	3	0:00:587	10
new.RL_15_2.4	3	3	0:00:443	10	3	0:00:590	10	3	0:00:722	10
new.RL_15_2.5	3	3	0:00:608	10	3	0:00:433	10	3	0:00:610	10
new.RL_15_2.6	3	3	0:00:449	10	3	0:03:730	9	3	0:00:409	10
new.RL_15_2.7	4	4	0:00:390	10	4	0:00:537	10	4	0:00:491	10
new.RL_15_2.8	3	3	0:00:478	10	3	0:00:551	10	3	0:00:422	10
new.RL_15_2.9	4	4	0:00:521	10	4	0:00:469	10	4	0:01:027	9
new.RL_15_5.10	3	3	0:00:556	10	3	0:00:461	10	3	0:00:662	10
new.RL_15_5.1	3	3	0:00:390	10	3	0:00:556	10	3	0:00:529	10
new.RL_15_5.2	3	3	0:00:484	10	3	0:00:446	10	3	0:00:397	10
new.RL_15_5.3	3	3	0:00:548	10	3	0:00:482	10	3	0:00:299	10
new.RL_15_5.4	3	3	0:00:592	10	3	0:00:511	10	3	0:00:419	10
new.RL_15_5.5	3	3	0:00:566	10	3	0:00:605	10	3	0:00:624	10
new.RL_15_5.6	3	3	0:00:443	10	3	0:00:487	10	3	0:00:393	10
new.RL_15_5.7	3	3	0:00:459	10	3	0:00:518	10	3	0:00:388	10
new.RL_15_5.8	3	3	0:00:561	10	3	0:00:501	10	3	0:00:438	10
new.RL_15_5.9	3	3	0:00:453	10	3	0:00:509	10	3	0:00:699	10
new.RL_25_3.10	4	4	0:00:772	10	4	0:01:970	10	4	0:02:408	10
new.RL_25_3.1	4	4	0:00:695	10	4	0:01:925	10	4	0:02:253	10
new.RL_25_3.2	4	4	0:00:899	10	4	0:02:222	10	4	0:02:410	10
new.RL_25_3.3	4	4	0:06:240	7	4	0:02:415	10	4	0:02:355	10
new.RL_25_3.4	4	4	0:00:872	10	4	0:02:195	10	4	0:02:289	10
new.RL_25_3.5	4	4	0:00:735	10	4	0:02:024	10	4	0:02:421	10
new.RL_25_3.6	4	4	0:01:415	10	4	0:02:211	10	4	0:02:672	10
new.RL_25_3.7	4	4	0:00:734	10	4	0:01:927	10	4	0:02:491	10
new.RL_25_3.8	4	4	0:00:769	10	4	0:02:103	10	4	0:02:456	10
new.RL_25_3.9	4	4	0:00:969	10	4	0:01:962	10	4	0:02:662	10
new.RL_30_2.10	4	4	0:02:226	10	4	0:05:877	10	4	0:09:011	9
new.RL_30_2.1	4	4	0:02:129	10	4	0:05:369	10	4	0:06:484	10
new.RL_30_2.2	4	4	0:07:905	7	4	0:36:847	5	4	0:12:245	8
new.RL_30_2.3	4	4	0:15:121	2	4	0:25:127	5	4	0:13:432	7
new.RL_30_2.4	4	4	0:05:827	8	4	0:05:941	10	4	0:09:962	9
new.RL_30_2.5	4	4	0:02:068	10	4	0:06:154	10	4	0:08:671	9
new.RL_30_2.6	4	4	0:01:850	10	4	0:05:728	10	4	0:06:407	10
new.RL_30_2.7	4	4	0:06:099	8	4	0:11:490	9	4	0:07:780	10
new.RL_30_2.8	4	4	0:10:740	7	4	0:06:826	10	4	0:07:727	10
new.RL_30_2.9	4	4	0:04:360	9	4	0:05:954	10	4	0:07:577	10

Tabela 5.12: Resultados computacionais para as instâncias aleatórias da classe C2 com $B = 155MB/s$ (Continuação).

Instância		AG			AM			AM+VB		
Nome	*	Melhor	Tempo	#	Melhor	Tempo	#	Melhor	Tempo	#
new.RL_30_9.10	4	4	0:20:103	1	4	0:19:836	6	4	0:16:058	6
new.RL_30_9.1	4	4	0:02:160	10	4	0:09:593	9	4	0:06:390	10
new.RL_30_9.2	4	4	0:01:955	10	4	0:05:331	10	4	0:06:327	10
new.RL_30_9.3	4	4	0:11:773	6	4	0:09:819	9	4	0:06:105	10
new.RL_30_9.4	4	4	0:15:145	3	4	0:29:645	5	4	0:14:773	6
new.RL_30_9.5	4	4	0:04:661	9	4	0:05:084	10	4	0:05:983	10
new.RL_30_9.6	4	4	0:20:253	2	4	0:06:134	10	4	0:13:285	7
new.RL_30_9.7	4	4	0:11:394	7	4	0:06:014	10	4	0:07:082	10
new.RL_30_9.8	4	4	0:15:058	5	4	0:05:496	10	4	0:09:200	9
new.RL_30_9.9	4	4	0:11:479	5	4	0:05:442	10	4	0:07:138	10
new.RL_50_2.10	5	5	0:25:002	5	5	1:17:729	9	5	1:27:196	9
new.RL_50_2.1	5	5	0:26:072	4	5	1:12:141	9	5	1:15:275	9
new.RL_50_2.2	5	5	0:19:150	6	5	1:34:496	7	5	1:29:055	8
new.RL_50_2.3	5	6	0:29:160	0	6	2:52:135	0	6	2:44:573	0
new.RL_50_2.4	5	5	0:29:977	1	-	2:36:574	0	5	2:15:174	4
new.RL_50_2.5	5	5	0:22:421	4	5	1:50:194	5	5	2:00:268	5
new.RL_50_2.6	5	5	0:16:315	9	5	1:17:218	9	5	1:13:851	9
new.RL_50_2.7	5	5	0:32:298	1	5	2:27:564	2	5	1:51:699	7
new.RL_50_2.8	5	5	0:23:364	4	5	2:25:740	1	5	1:33:843	10
new.RL_50_2.9	5	5	0:27:938	5	5	2:02:101	5	5	1:32:092	9
new.RL_50_8.10	5	5	0:20:057	6	5	1:54:781	6	5	1:21:681	10
new.RL_50_8.1	5	5	0:26:412	4	5	1:25:407	8	5	1:28:174	9
new.RL_50_8.2	5	5	0:22:289	1	6	2:28:757	0	5	1:08:562	10
new.RL_50_8.3	5	5	0:24:170	2	5	2:25:881	3	5	2:04:577	6
new.RL_50_8.4	5	5	0:24:340	3	5	1:49:145	6	5	2:02:677	4
new.RL_50_8.5	6	6	0:20:870	6	6	1:06:967	9	6	1:07:089	9
new.RL_50_8.6	5	5	0:23:384	5	5	2:04:853	5	5	1:10:259	10
new.RL_50_8.7	5	-	0:31:970	0	5	2:44:891	1	5	1:28:653	9
new.RL_50_8.8	5	5	0:21:148	4	5	2:36:485	1	5	1:28:297	9
new.RL_50_8.9	5	-	0:39:459	0	5	2:58:716	1	5	1:24:218	10
Resumo		*(%)	MTempo	M#	*(%)	MTempo	M#	*(%)	MTempo	M#
		95,71	0:10:032	7,22	95,71	0:38:910	8,05	98,57	0:30:626	9,07

Tabela 5.13: Resultados computacionais para as instâncias aleatórias da classe C2 com $B = 622MB/s$.

Instância		AG			AM			AM+VB		
Nome	*	Melhor	Tempo	#	Melhor	Tempo	#	Melhor	Tempo	#
new.RH_15_10.10	3	3	0:00:469	10	3	0:00:512	10	3	0:00:620	10
new.RH_15_10.1	3	3	0:00:557	10	3	0:00:452	10	3	0:00:373	10
new.RH_15_10.2	3	3	0:00:499	10	3	0:00:424	10	3	0:00:544	10
new.RH_15_10.3	3	3	0:02:725	9	3	0:00:610	10	3	0:00:424	10
new.RH_15_10.4	3	3	0:02:638	9	3	0:00:467	10	3	0:01:639	8
new.RH_15_10.5	3	3	0:00:465	10	3	0:00:495	10	3	0:00:438	10
new.RH_15_10.6	3	3	0:00:358	10	3	0:00:480	10	3	0:00:527	10
new.RH_15_10.7	3	3	0:00:515	10	3	0:00:426	10	3	0:00:518	10
new.RH_15_10.8	3	3	0:05:037	8	3	0:00:554	10	3	0:01:513	8
new.RH_15_10.9	3	3	0:11:957	5	3	0:00:480	10	3	0:01:014	9
new.RH_30_5.10	4	4	0:18:225	2	4	0:31:802	5	4	0:10:468	8
new.RH_30_5.1	4	4	0:04:583	9	4	0:04:219	10	4	0:05:407	10
new.RH_30_5.2	4	4	0:04:530	9	4	0:04:078	10	4	0:05:334	10
new.RH_30_5.3	4	4	0:01:923	10	4	0:04:204	10	4	0:05:259	10
new.RH_30_5.4	4	4	0:04:108	9	4	0:04:354	10	4	0:05:408	10
new.RH_30_5.5	4	4	0:21:374	2	4	0:17:921	8	4	0:15:823	6
new.RH_30_5.6	4	4	0:12:005	5	4	0:04:951	10	4	0:05:768	10
new.RH_30_5.7	4	4	0:06:953	8	4	0:05:079	10	4	0:06:379	10
new.RH_30_5.8	4	4	0:12:688	5	4	0:05:001	10	4	0:07:071	10
new.RH_30_5.9	4	4	0:15:062	1	4	0:30:514	4	4	0:09:014	9
Resumo		*(%)	MTempo	M#	*(%)	MTempo	M#	*(%)	MTempo	M#
		100	0:06:333	7,55	100	0:05:851	9,35	100	0:04:177	9,40

Finalmente, a Tabela 5.13 mostra os resultados dos testes para as instâncias aleatórias da classe C2 com $B = 622MB/s$. Nesse subconjunto de instâncias, ocorreu um fato singular: o tempo de processamento mais alto foi apresentado pelo AG, enquanto o AM+VB se mostrou o algoritmo mais rápido. Os dados mostram que o AG obteve uma taxa de obtenção de ótimos não muito alta. Também é possível perceber que os algoritmos meméticos com e sem o VB se comportaram de maneira bastante semelhante, obtendo taxas de convergência quase idênticas e executando em tempos de processamento não muito diferentes.

Capítulo 6

Conclusões e Trabalhos Futuros

Neste trabalho, apresentamos duas implementações metaheurísticas puras – Algoritmo Genético (AG) e Algoritmo Memético (AM) – além de uma terceira implementação híbrida – Algoritmo Memético com *Vocabulary Building* (AM+VB) – para o Problema de Atribuição de Localidades a Anéis SONET. O AG utilizou duas heurísticas gulosas randomizadas (*Random Edge-based* e *Random Cut-based*) para compor a população inicial, o Método da Roleta para fazer a seleção, e Elitismo e BPX*crossover* para reprodução dos indivíduos. O AM foi composto através de uma nova versão do AG que empregava busca local sobre duas vizinhanças: N_1 e N_3. A técnica Construção de Vocábulos foi aplicada na terceira implementação, AM+VB, em um modelo de contração de vértices o qual identificava um padrão em várias soluções de boa qualidade e a partir destes formava os vocábulos. A população composta de soluções mistas, isto é, localidades e vocábulos, foi otimizada com as buscas locais através das vizinhanças N_1 e N_3 devidamente adaptadas a essas novas configurações.

Extensivos experimentos computacionais foram realizados sobre as classes de instâncias C1 e C2. Os três algoritmos foram executados 10 vezes sobre cada uma dessas instâncias utilizando os mesmos parâmetros. Os resultados computacionais foram coletados e analisados observando-se a qualidade das soluções, a média de tempo e quantidade de ótimos encontrados durante as 10 execuções de cada umas das 348 instâncias.

Verificamos que o algoritmo AM+VB foi o que apresentou maior número de instâncias resolvidas de forma ótima para as duas classes testadas, seguido pelo AM e AG, respectivamente. Em compensação, o AG mostrou-se bem mais rápido que os de-

mais algoritmos nas duas classes. O AM foi mais rápido que o AM+VB na classe C1, equanto que para as instâncias de C2 o AM+VB apresentou menor tempo que o AM.

Como sugestões para trabalhos futuros podemos citar as seguintes possibilidades:

1. A exploração de outras vizinhanças dentro do Algoritmo Memético que permitam, por exemplo, sair de soluções viáveis para inviáveis.

2. O desenvolvimento de outras abordagens da Construção de Vocábulos dentro do AM proposto, ou em outras meta-heurísticas.

3. A utilização das Mateheurísticas, aplicando-se os métodos exatos nas vizinhanças utilizadas.

Referências Bibliográficas

ARINGHIERI, R.; DELL'AMICO, M. Solution of the sonet ring assignment problem with capacity constraints. *Technical Report 12*, DISMI - University of Modena and Reggio Emilia, 2001.

ARINGHIERI, R.; DELL'AMICO, M. Comparing metaheuristic algorithms for sonet network design problems. *Journal of Heuristics*, v. 11, p. 35–57, 2005.

BASTOS, L. de O. *Soluções Heurísticas para o Problema de Atribuição de Localidades a Anéis em Redes SONET*. [S.l.]: Dissertação (Mestrado em Computação) – Programa de Pós-Graduação em Computação, UFF, 2005.

BASTOS, L. de O.; OCHI, L. S. A genetic algorithm with evolutionary path-relinking for the sonet ring assignment problem. *EngOpt 2008 - International Conference on Engineering Optimization*, 2008.

BASTOS, L. de O.; OCHI, L. S.; MACAMBIRA, E. M. Grasp with path-relinking for the sonet ring assignment problem. *Hybrid Intelligent Systems*.

BASTOS, L. de O.; OCHI, L. S.; MACAMBIRA, E. M. A grasp with path-relinking for the sonet ring assignment problem. *Proceendings of the 5th International Conference on Hybrid Intelligent Systems*, 2005.

BASTOS, L. de O.; OCHI, L. S.; MACAMBIRA, E. M. A relative neighborhood grasp for the sonet ring assignment problem. *Proceendings of the International Network Optimization*, p. 833–838, 2005.

COOK, W. et al. *Combinatorial Optimization*. New York: John Wiley & Sons, 1998.

DARWIN, C. *On The Origin of Species*. 1. ed. [S.l.]: Harward University Press, 1859.

FALKNAUER, E. *Genetic Algorithms and Grouping Problems*. [S.l.]: WILEY, 1998.

FISCHETTI, M.; SALAZAR, J. J.; TOTH, P. The symmetric generalized traveling salesman problem. *Networks*, v. 26, p. 113–123, 1995.

GLOVER, F. Parametric combinations of local job shop rules. *Research Memorandum*, GSIA, Carnegie Melon University, Pittsburg, v. 117, 1963.

GLOVER, F. New ejection chain and alternating path methods for traveling salesman problems. *Computer Science and Operations Research*, Cambridge, p. 449–509, 1992.

GLOVER, F. A template for scatter search and path relinking. in: Hao, j.k. (ed.); lutton, e. (ed.); ronald, e. (ed.); shoenauer, m. (ed.); snyers, d.(ed.). *Lecture Notes in Computer Science*, p. 13–54, 1997.

GLOVER, F. Scatter search and path relinking. *New Ideas in Optimization*, Maidenhead: McGraw Hill, p. 297–319, 1999.

GLOVER, F. Tutorial on surrogate constraint approaches for optimization in graphs. *Journal of Heuristics*, v. 9, p. 175–228, 2003.

GLOVER, F.; KOCHENBERGER, G. A. *Handbook of Metaheuristics*. 1. ed. [S.l.]: Kluwer Academic Publishers, 2003.

GLOVER, F.; LAGUNA, M. Tabu search. *Modern Heuristic Techniques for Combinatorial Problems*, Oxford: Blackwell Scientific Publishing, p. 71–140, 1993.

GLOVER, F.; LAGUNA, M. *Tabu search*. [S.l.]: Kluwer Academic Publishers, 1997.

GLOVER, F.; LAGUNA, M.; MARTI, R. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, v. 29, p. 653–684, 2000.

GOLDSCHMIDT, O.; LAUGIER, A.; OLINICK, E. V. Sonet/sdh ring assignment with capacity constraints. *Discrete Applied Mathematics*, v. 129, p. 99–128, 2003.

GUEDES, A. B. da C.; ALOISE, D. J. Um algoritmo memético para o problema do caixeiro viajante assimétrico: Uma abordagem baseada em vocabulary building. *SBPO*, 2006.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. [S.l.]: Ann Arbor: University of Michigan Press, 1975.

JüNGER, M.; REINELT, G.; RINALDI, G. *The Traveling Salesman Problem*. Elsevier, North Holland: Handbook on Operations Research and Management Science, 1995.

KELLY, J. P.; XU, J. Tabu search and vocabulary building for routing problems. *Technical Rreport*, 1995.

LAPORTE, G.; NOBERT, Y. Generalized traveling salesman problem through n sets of nodes: an interger programming. *Information Systems and Operational Research*, v. 21, p. 61–75, 1983.

LAWLER, E. L. *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rinehart and Winston, 1976.

MACAMBIRA, E. M. *Modelos e Algoritmos de Programação Inteira no Projeto de Redes de Telecomunicações*. [S.l.]: Tese (Doutorado) – COPPE, Programa de Engenharia de Sistemas e Computação, UFRJ, 2003.

MACAMBIRA, E. M.; FILHO, N. M.; SOUZA, C. C. de. Projeto de uma rede de telecomunicações usando metaheurística. *Anais do XXXV Simpósio Brasileiro de Pesquisa Operacional*, 2003.

MACAMBIRA, E. M.; MACULAN, N.; SOUZA, C. C. de. A column generation approach for sonet ring assignment. *Networks*, 2005.

MEHOTRA, A.; TRICK, M. A. Cliques and clustering: a combinatorial aproach. *Operations Research Letters*, 1998.

MOSCATO, P. On evolution, search, optimization, genetic algorithms, and martial arts: Towards memetic algorithms. *Technical Report*, Caltech Concurrent Computation Program, 1989.

NORONHA, T. F.; ALOISE, D. J. Uma abordagem sobre estratégias metaheurísticas. *REIC. Revista eletrônica de iniciação científica*, v. 1, 2001.

OMIDYAR, C. G.; ALDRIDGE, A. Introduction to sdh/sonet. *Communications Magazine*, p. 30–33, 1993.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial Optimization: Algorithms and Complexity*. 1. ed. New York: Prentice Hall, 1982.

ROCHAT, Y.; TAILLARD, E. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, p. 147–167, 1995.

SCHOLL, A.; KLEIN, R.; DOMSCHKE, W. Pattern based vocabulary building for effectively sequencing mixed-model assembly lines. *Journal of Heuristics*, v. 4, p. 359–381, 1998.

SOARES, W. K. da S. *Heurísticas Usando Construção de Vocabulário Aplicadas ao Problema da Atribuição de Localidades a Anéis em Redes SONET/SDH*. [S.l.]: Dissertação (Mestrado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, UFRN, 2008.

SORIANO, P. et al. *Design and Dimensioning of survivable SDH/SONET Networks*. [S.l.]: In e P. Soriano (eds), B. Sansó (editor), Telecomunications Network Planning. Kluwer Academics Publishers, 1999.

WANSEM, O. J.; WU, T. H.; CARDWELL, R. H. Survivable sonet networks - design methodologies. *IEEE Journal on Selected Areas in Communications*, v. 12, p. 205–212, 1994.