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

CHARLES MILLER DE GÓIS OLIVEIRA

MODELO E MÉTODO DE RESOLUÇÃO PARA REDE DE ALTA VELOCIDADE, UMA ABORDAGEM DO PROBLEMA DO CAIXEIRO VIAJANTE COM COLETA DE PRÊMIOS

MOSSORÓ - RN
2018

CHARLES MILLER DE GÓIS OLIVEIRA

MODELO E MÉTODO DE RESOLUÇÃO PARA REDE DE ALTA VELOCIDADE, UMA ABORDAGEM DO PROBLEMA DO CAIXEIRO VIAJANTE COM COLETA DE PRÊMIOS

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

Orientador: Prof. Dr. Francisco Chagas de Lima Júnior
Co-orientador: Prof. Dr. Carlos Heitor Pereira Liberalino

MOSSORÓ - RN
2018

Catalogação da Publicação na Fonte.

Universidade do Estado do Rio Grande do Norte.

O48m Oliveira, Charles Miller de Góis
MODELO E MÉTODO DE RESOLUÇÃO PARA REDE DE AL-TA VELOCIDADE, UMA ABORDAGEM DO PROBLEMA DO CAIXEIRO VIAJANTE COM COLETA DE PRÊMIOS. / Charles Miller de Góis Oliveira. - Mossoró, 2018.
92p.

Orientador(a): Prof. Dr. Francisco Chagas de Lima Júnior.
Coorientador(a): Prof. Dr. Carlos Heitor Pereira Liberalino.
Dissertação (Mestrado em Programa de Pós-Graduação em Ciência da Computação). Universidade do Estado do Rio Grande do Norte.

1. Programa de Pós-Graduação em Ciência da Computação. 2. Redes Giga. 3. GigaMossoró. 4. Algoritmo Genético. I. Lima Júnior, Francisco Chagas de. II. Universidade do Estado do Rio Grande do Norte. III. Título.

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

CHARLES MILLER DE GÓIS OLIVEIRA

MODELO E MÉTODO DE RESOLUÇÃO PARA REDE DE ALTA VELOCIDADE, UMA ABORDAGEM DO PROBLEMA DO CAIXEIRO VIAJANTE COM COLETA DE PRÊMIOS

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

APROVADA EM: 05 / 10 /2018.

BANCA EXAMINADORA

Dr. Francisco Chagas de Lima Júnior - UERN
Orientador

Dr. Carlos Heitor Pereira Liberalino - UERN
Co-orientador

Dr. Dario José Aloise - UERN
Membro Interno

Dr. Gustavo Augusto Lima de Campos - UECE
Membro Externo

Dedico está este trabalho primeiramente a DEUS, por ser essecial em minha vida e sua infinita força e grandeza que me guia, a minha esposa Maria Laudinete pelo apoio incondicional e constante incentivo, ela também é responsável pela conclusão deste e meus pais Valdeci e Luzia pelo amor e incentivo.

AGRADECIMENTOS

Quero primeiramente agradecer a DEUS pela oportunidade única que me foi dada “a VIDA”.

Agradeço a minha minha esposa, pelo amor, incentivo, força, que sempre me apoiou nos momentos difíceis e de incerteza, muito comum para quem tentar trilhar novos caminhos. Obrigado por sempre estar me encorajando e me entendendo nas horas mais difíceis, agradeço por todo o seu carinho e amor, você tem sido fonte inesgotável de força e coragem nessa caminhada.

Agradeço a minha família, em especial, a meus pais pela imensa manifestação de amor, carinho e respeito que recebo deles, sendo a fonte propulsora que me alimenta todos os dias. A meu irmão, minhas irmãs e sobrinhos, pela energia e sintonia que sempre nos rodeou, e, sobretudo a todo o carinho e respeito que temos um pelo outro, me mostrando que o sorriso e a alegria de estarmos juntos é sempre maior que qualquer outro prêmio.

Agradeço ao professor Lima Jr, por seu emprenho, compreensão, conversa sempre amigável e por não ter desistido e acreditado no meu esforço. Além de um excelente professor e orientados é um excelente ser humano, sempre demonstrandos ética, paciência e bondade em seus atos.

Ao meu co-orientador Heitor, pela paciência e dedicação para estar sempre lendo e realizando correções no trabalho, com sua alegria e sempre bem receptivo a ideias, que isso seja sempre uma fonte inesgotável.

Agradeço a UERN pela oportunidade aqui dada e a banca por aceitar o convite e contribuir de forma significativa para a melhorai deste trabalho.

“Tenho a impressão de ter sido uma criança brincando à beira-mar, divertindo-me em descobrir uma pedrinha mais lisa ou uma concha mais bonita que as outras, enquanto o imenso oceano da verdade continua misterioso diante de meus olhos” - Isaac Newton.

RESUMO

As Redes Comunitárias de Educação e Pesquisa (REDECOMEP) tem como base o desenvolvimento científico através da conexão de instituições de ensino superior, um exemplo disso é a rede GigaMossoró. Este trabalho discute aspectos e desafios de como encontrar a melhor rota, propondo um método de resolução que defina o perímetro da rede GigaMossoró, de modo a obter um caminho que contemple todos os pontos prioritários e consiga alcançar o máximo de possíveis clientes dentro do mesmo. Como parâmetro será utilizado o Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP), O PCVCP foi desenvolvido inicialmente por Balas em 1989 para a programação de fabricas que produzia lâminas de aço. Dado que o problema proposto é reconhecidamente um problema de alta complexidade computacional, ou seja, comprovadamente NP-difícil, para resolução do problema foi proposta um modelo matemático comando. Definido o modelo foi necessário validar o mode, para isso utilizou as instâncias propostas no trabalho do Chaves (2003) e as instâncias elaboradas na pesquisa para o problema estudado. Diante da validação através de um modelo exato, percebeu-se a necessidade da utilização de uma abordagem via Metaheurística com base em um algoritmo genético, esses têm as caraceristicas de utilizar chaves aleatórias e cruzamento genético como parâmetros de seleção da melhor solução, para isso foi desenvolvido um algoritmo genético para solução do problema e obtidos resultados satisfatórios para o mesmo. Diante dessa abordagem a pesquisa apresentou contribuições quanto a definição de conceitos da rede GigaMossoró de modo a conhecer seus futuros clientes, e definir sua estrutura e restrições, além do mais foi possível entender abordagem de solução do problema través de Metaheurística proposta.

Palavras-chaves: Redes Giga; GigaMossoró; Algoritmo Genético.

ABSTRACT

The Community Education and Research Networks (REDECOMEP) is based on scientific development through the connection of higher education institutions, an example of which is the GigaMossoró network. This paper discusses aspects and challenges of how to find the best route, proposing a method of resolution that defines the perimeter of the GigaMossoró network, in order to obtain a path that contemplates all the priority points and is able to reach the maximum of possible clients within it. As a parameter will be used the Traveling Collector's Problem with Collecting Prizes (PCVCP), PCVCP was initially developed by Balas in 1989 for the programming of factories that produced steel blades. Given that the proposed problem is admittedly a problem of high computational complexity, ie, con-proved NP-difficult, to solve the problem was proposed a mathematical model command. Defined the model was necessary to validate the mode, for this he used the proposed instances in the work of Chaves (2003) and the instances elaborated in the research for the studied problem. Given the validation through an exact model, it was realized the need to use a Metaheuristic approach based on a genetic algorithm, these have the characteristics of using random keys and genetic mutation as parameters to select the best solution, to this was developed a genetic algorithm to solve the problem and obtained satisfactory results for the same. Given this approach, the research presented contributions regarding the definition of concepts of the GigaMossoró network in order to know its future clients, and to define its structure and restrictions, in addition, it was possible to understand a solution approach of the problem through proposed Metaheuristics.

Key-words: Giga Networks; GigaMossoró; Genetic Algorithm.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

REDECOMEP	Redes Comunitárias de Educação e Pesquisa
MCTI	Ministério da Ciência, Tecnologia e Inovação
TLBO	*Teaching Learning Based Optimization*
UERN	Universidade do Estado do Rio Grande do Norte
UFERSA	Universidade Federal Rural do Semiárido
IFRN-Mossoró	Instituto Federal do Rio Grande do Norte campus Mossoró
ARPANET	*Advanced Research Projects Agency Network*
RNP	Rede Nacional de Pesquisa
MCTI	Ministérios da Ciência e Tecnologia e Inovação
RMAV	Redes Metropolitanas de Alta Velocidade
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico-
PoPs	Pontos de Presença
FO	Função Objetivo
ARPANET	*Advanced Research Projects Agency Network*
PCV	Problema do Caixeiro Viajante
PCVCP	Problema do Caixeiro Viajante com Coleta de Prêmios
PLF	Problema de Localização de Facilidades
PMP	p-Median Problem
PLMC	Problemas de Localização de Máxima Cobertura
PGA	Problema Geral de Atribuição
PLA	Problema Linear de Atribuição
PQA	Problema Quadrático de Atribuição
QAP	*Quadratic Assignment Problem*
PAM	Problema de Atribuição Multidimensional
PAD	Problema de Atribuição Dinâmica
TDMA	*The Time Slot Assignment Problem*
PKS	Problema dos K-Servos
PCM	Problema do Caminho Mínimo
PFM	Problema do Fluxo Máximo
PFCM	Problema de Fluxo com Custo Mínimo
PCTSP	Prize Collecting Traveling Salesman Problem
PCVCP	Problema do Caixeiro Viajante com Coleta de Prêmios

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

1.1. Contextualização

A evolução das redes de alta velocidade é uma realidade constante nos centros urbanos, e têm se tornado cada vez mais presente no cotidiano de diversos segmentos como educação, indústria, comércio, entretenimento, etc. Como exemplo destas "infovias" podemos citar, as Redes Comunitárias de Educação e Pesquisa (REDECOMEP), uma iniciativa do Ministério da Ciência, Tecnologia e Inovação (MCTI) que é coordenada pela Rede Nacional de Ensino e Pesquisa (RNP).

Os modelos de implantação da REDECOMEP têm como base central a instalação de uma infraestrutura de fibra óptica própria, com o intuito de conectar centros de pesquisa e de educação superior, sejam públicos ou privados. O objetivo é dar uma melhor infraestrutura de rede e atender com aplicações e suporte à serviços inovadores, oferecendo apoio ao desenvolvimento da ciência e tecnologia. O modelo de negócio para sustentação da rede é a formação de um consórcio entre os centros participantes (clientes), de modo que esses financiarão a manutenção e a sustentação da mesma. Outros parceiros também poderão contribuir de forma significativa para manutenção da rede como escolas, hospitais, instituições públicas federais, estaduais, municipais e privadas.

Para alcançar esse cenário um conjunto de ações são levadas em consideração. Uma destas é a de que a rede deve obrigatoriamente conectar centros de pesquisa e ensino públicos e privados dentro do perímetro determinado e para garantir a auto sustentação, deve atingir o maior número de possíveis clientes em seu trajeto, uma vez que estes irão promover um retorno financeiro, ao utilizarem os serviços de conexão da rede.

Diante desse contexto, está a rede metropolitana de conexão de alta velocidade da cidade de Mossoró, denominada GigaMossoró, com uma infraestrutura própria. A ideia da rede é criar uma conexão de alta velocidade entre as principais instituições públicas de ensino superior da cidade, que são: a Universidade do Estado do Rio Grande do Norte (UERN), a Universidade Federal Rural do Semiárido (UFERSA) e o Instituto Federal do Rio Grande do Norte campus Mossoró (IFRN-Mossoró). Outros parceiros também poderão participar da rede, como: entidades públicas, associações, fundações, sociedades civis de fins não lucrativos e pessoas jurídicas

de qualquer natureza, desde que tenham relevância para a oferta e demanda de tecnologia para se conectarem à rede dentro da região metropolitana da cidade de Mossoró. A adesão desses participantes irá ajudar na manutenção da rede.

A rede GigaMossoró tem como proposta inicial sua construção em formato de anel, com fibras que farão a conexão em ambos os sentidos, garantindo assim sua redundância. Esse anel integrará toda a região central e as instituições que inicialmente farão parte do projeto e que são as prioritárias dentro das condições exigidas pela RNP. A mesma terá o tamanho máximo de 24 km, estimados com base nos recursos disponíveis do projeto.

Diante desse cenário existe a necessidade de se traçar uma rota visando garantir a interconexão em formato de anel das instituições parceiras, de modo que a fibra atinja o maior número de possíveis clientes que contribuam na manutenção da rede, além de garantir a conexão das instituições formadoras do consórcio e que esta não ultrapasse o perímetro de 24km.

Analise do tema e de suas características técnicas e de relevância para a sociedade, bem como a sua proposição dentro do ambiente no qual está inserido. Pode-se destacar que o problema se alinha como o problema do caixeiro viajante, uma vez que a rede, por ter formato de anel, possui ponto de partida e chegada coincidentes, além de diversos nós intermediários. Entretanto, o problema aqui abordado apresenta uma variação, pois durante o percurso algumas anuências devem ser consideradas como: os clientes prioritários devem obrigatoriamente fazer parte da rede, a mesma não poderá ultrapassar um circuito de 24km e durante esse percurso deve tentar abranger um número elevado de possíveis clientes.

Assim, o objetivo da pesquisa consiste em propor um modelo matemático que defina o perímetro da rede GigaMossoró, de modo a obter um caminho que contemple todos os pontos prioritários e ao mesmo tempo o maior número de possíveis clientes dentro do mesmo. De forma a tratar este problema, foi utilizado como base o modelo matemático do caixeiro viajante com coleta de prêmios. Para a sua resolução, será utilizada a metaheurística com algoritmo genético.

1.2. Motivação

O tema se caracteriza como um problema prático e de grande relevância para sociedade, uma vez que o problema se caracteriza dentro do contexto da rede Giga que será implementada na cidade de Mossoró. Além disso a definição de um caminho crítico da rede GigaMossoró, irá garantir a sobrevivência da mesma e trará desenvolvimento científico e tecnológico para a região, uma vez que seu pleno funcionamento ajudará no desenvolvimento de tecnologias e ciência, com o compartilhamento de informações e conhecimento entre as instituições parceiras.

Além disso existe a contribuição quanto a abordagem de resolução do problema, pois haverá o emprego do conhecimento aqui desenvolvido e a implementação de uma metaheurística adaptada para este tipo de problema.

1.3. Objetivo geral

Propor um método de resolução que defina o perímetro da rede GigaMossoró de modo a obter o caminho que contemple todos os pontos prioritários e consiga alcançar o máximo de possíveis clientes, dentro do percurso determinado, utilizando uma abordagem com base em otimização combinatória.

1.4. Objetivos específicos

Como objetivos específicos deste trabalho podemos destacar:

- Propor uma modelagem matemática para o problema abordado;
- Propor um método de resolução com base em otimização combinatória para uma rede de alta velocidade;
- Entender o método de modelagem baseado no problema do caixeiro viajante com coleta de prêmio (PCVCP);
- Propor e implementar uma metaheurística com algoritmo genético;
- Propor solução para o problema de modo a definir a rota de menor caminho que consiga conectar os pontos prioritários da rede GigaMossoró e seus futuros clientes;
- Apresentar rotas alternativas para rede GigaMossoró de modo que consiga agregar o número máximo de possíveis clientes;

- Realizar experimentos computacionais;
- Testar o desempenho do método proposto.

1.5. Organização da dissertação

A presente pesquisa encontra-se dividida em sete capítulos. O primeiro capítulo apresenta a contextualização do problema, mostrando suas definições básicas, como também a motivação da pesquisa, os objetivos gerais e específicos, além da organização da dissertação.

No capítulo 2 será trabalhada a base conceitual da pesquisa, a fundamentação teórica do estudo, abordando e apresentando os conceitos de modelos matemáticos e realizando uma revisão de literatura acerca do estudo.

O capítulo 3 trás o contexto e abordagem proposta do problema, identificando sua definição e abordagem proposta para os modelos e métodos de resolução, além de fazer abordagem da proposta de metaheurística a ser aplicada. No capítulo 4 será abordada a contribuição cientifica da proposta desenvolvida no estudo.

O levantamento e detalhamento da proposta é apresentado no capítulo 5, com também as instâncias utilizadas e o processo de parametrização das mesmas. Esse detalhamento da pesquisa é complementado no capítulo 6, com validação do modeo e os testes computacionais e apresentação dos resultados sobtidos.

Por fim no capítulo 7 serão abordadas as considerações finais do trabalho bem como recomendações para trabalhos futuros.

CAPÍTULO 2

REFERENCIAL TEÓRICO

Neste capítulo são apresentados os fundamentos acerca dos assuntos abordados no presente trabalho, com definições que irá permitir o leitor no melhor entendimento do tema.

2.1. Redes de alta velocidade

O desenvolvimento das tecnologias de redes de computadores vem crescendo em ritmos acelerados nas últimas décadas. Essa evolução, que iniciou com um experimento acadêmico, o *Advanced Research Projects Agency Network (*ARPANET) em 1969, o que possibilitou a criação de possibilitou a criação várias topologias e tecnologias que deram origem às redes de diferentes portes (LANs, MANs e WANs) e diferentes arquiteturas.

Contribuições mais recentes em relação ao meio físico para a transmissão digital e equipamentos de conexão de rede computadorizados criaram subsídios para novas técnicas de transmissão que apresentam um desempenho muito melhor, maximizando a velocidade e minimizando as perdas de dados durante a transmissão. Além disto, proporcionou a diminuição dos custos associados aos equipamentos. Essas novas tecnologias deram origem as redes de alta velocidade ou, como algumas vezes são chamadas “redes Gigabits” ou “infovias”.

As redes de alta velocidade surgiram e foram impulsionadas principalmente pela influência das redes públicas de telefonia e de telecomunicação. Tais redes sofreram um processo de migração da tecnologia analógica para digital devido a necessidade de fornecer serviço de transmissão de voz a custos menores (SILVEIRA, 2006).

A ideia subjacente dos estudos das redes é de que elas promovem um ambiente favorável ao compartilhamento de informações, de conhecimentos, de habilidades e de recursos essenciais para os processos de inovação. A configuração em rede consiste, então, em uma forma eficaz das organizações alcançarem competitividade nos mercados, isto por meio de um complexo ordenamento de relacionamentos, em que as organizações estabelecem inter-relações (GARCIA et al, 2010).

Neste sentido, entende-se “por rede metropolitana de alta velocidade” uma estrutura de rede de computadores que, tipicamente, conecta instituições que se encontram distribuídas na área metropolitana de uma cidade através de uma tecnologia que permite a transferência de informação a altas velocidades (MEIRA Jr., 1999).

As redes de alta velocidade apresentam uma arquitetura padronizada, baseada em tecnologias e sistemas computacionais distribuídos, permitindo a implantação rápida e flexível de novos serviços, garantido a qualidade sobre serviços e a confiabilidade sobre os mesmos. Além disso, asseguram um bom desempenho da rede (MEIRA Jr., 1999).

O Projeto de Redes Metropolitanas de Alta Velocidade (RMAV) iniciou-se em meados de 1997 através da RNP com apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), que decidiram fomentar iniciativas em direção à Internet 2.

O termo internet 2.0 está associado a aplicações web onde o objetivo principal é facilitar o desenvolvimento e evolução do compartilhamento de informações de maneira interativa, interoperabilidade, desenvolvimento com foco no usuário e colaboração no *world wide web,* é a mudança para uma internet como plataforma, e um entendimento das regras para obter sucesso nesta nova plataforma (O'Reilly, 2003).

No Brasil essas redes receberam o nome de Redes Comunitárias de Educação e Pesquisa (REDECOMEP), e são responsáveis por implantar redes de alta velocidade nas regiões metropolitanas do país servidas por Pontos de Presença (PoPs), o PoP é onde se encontram os equipamentos de acesso ao usuário e IP da rede que se interliga com a internet, e em cidades do interior com duas ou mais instituições federais de ensino e pesquisa.

Esse modelo adotado prevê a formação de consórcios entre as organizações participantes de forma a assegurar a auto sustentação da rede, desenvolvendo uma infraestrutura de fibra óptica própria para a pesquisa e a educação superior.

Nesse contexto está a proposta da rede metropolitana de conexão de alta velocidade da cidade de Mossoró-RN, denominada GigaMossoró. A ideia desta rede é de criar uma conexão de alta velocidade entre as três principais instituições públicas de ensino superior da cidade, que são: a Universidade do Estado do Rio Grande do Norte (UERN), a Universidade Federal Rural do Semiárido (UFERSA) e o Instituto Federal do Rio Grande do Norte - Campus Mossoró (IFRN-Mossoró). Sendo toda esta operação possível através de uma infraestrutura própria.

A auto sustentação da rede é condicionada a entrada de parceiros, como entidades públicas, associações, fundações, sociedades civis de fins não lucrativos e pessoas jurídicas de qualquer natureza, mas que tenham relevância para a oferta e demanda de tecnologia na região

metropolitana da cidade de Mossoró. A adesão desses participantes contribuirá na manutenção da rede.

Na Europa e nos Estados Unidos, existem consórcios entre universidades e empresas que se preocupam em desenvolver as ferramentas e aplicações que utilizam as tecnologias desta rede. Nestes países estas iniciativas estão bem consolidadas (MEIRA Jr., 1999).

2.2. Otimização combinatória

A otimização combinatória se utiliza de algoritmos para resolução de problemas complexos e importantes para a computação. Entende-se por "otimizar" encontrar o valor ótimo para um problema, normalmente sob determinadas condições denominadas restrições dos problemas (PAPADIMITRIOU & STEIGLITZ, 1982).

Para Deb (2001) a otimização é o processo de encontrar e comparar soluções exequíveis até que nenhuma solução melhor possa ser encontrada, essas soluções são ditas boas ou ruins em termos de um objetivo.

A formulação matemática de um problema de otimização combinatória apresenta uma função e um conjunto de requisitos ou restrições, ambos relacionados às variáveis de decisão. Em problemas de otimização, a Função Objetivo (FO) deve ser de minimizar ou de maximizar (PAPADIMITRIOU & STEIGLITZ, 1982)

De acordo com Krasnogor (2002) os problemas de otimização combinatória são problemas para os quais o espaço de soluções possíveis (viáveis, candidatas ou factíveis) é finito (embora extremamente grande) e discreto.

Desse modo, os problemas de otimização permitem a análise e otimização de situações em um número elevado de aplicações do mundo real, como por exemplo redes de transporte logístico, redes sociais, rotas de transporte aéreo, mapeamento genético, sistemas de comunicação, sistemas elétricos, analise de bancos de dados, dentre outros. Dessa forma serão abordados conceitos desses problemas no próximo tópico.

2.3. Problemas clássicos de otimização combinatória

Para Miyazawa (2013) os problemas de otimização em seu modo geral têm como objetivo maximizar e minimizar funções definidas dentro de um domínio específico. Para os problemas de otimização combinatória é possível listar todos os elementos do problema e até mesmo testar se esse elemento faz parte do domínio.

Os problemas de análise combinatória são aplicados a diversos problemas práticos, como serão apresentados a seguir.

2.3.1. Problema do caixeiro viajante e derivações

Para Souza (2008) o Problema do Caixeiro Viajante (PCV) pode ser apresentado como um conjunto de *n* cidades/pontos, com distâncias conhecidas entre tais, onde um caixeiro viajante tem o seguinte objetivo: ele deve sair de uma cidade, chamada de cidade 0 (zero), deverá visitar cada uma das n-1 cidades e retornar para o ponto de partida, isto assumindo que ele deve visitar cada ponto uma única vez e de forma a reduzir os custos associado a viagem. Assim, o problema visa encontrar uma rota fechada mínima que passe uma única vez em cada cidade.

O PCV apresenta-se como um dos problemas clássicos de otimização combinatória, pois possui uma vasta aplicabilidade e ser um problema de fácil descrição e compreensão, é bastante utilizado no experimento de diversos métodos de otimização por ser, principalmente, um problema de fácil descrição e compreensão, entretanto, é um problema que possui grande complexidade na busca de uma solução ótima, ou seja, é classificado como NP-Árduo (KARP, 1975).

Para Laporte e Martello (1990), o PCV é amplamente difundido e estudado devido a características como: grande aplicabilidade prática, elevada dificuldade na obtenção de uma solução exata e uma significativa relação com outros modelos. Sua primeira abordagem formal é reportada a Hassler Whitney em 1934 em Princeton University (MUNHOZ et al, 2012).

De modo mais formal Dantizig et al (1954) definem o PCV como um grafo G = (V, A), de modo que o conjunto de vértices V representa as cidades a serem incluídas no circuito, e o conjunto de arestas A representa as possibilidades existentes de se viajar de uma cidade para outra.

De forma a garantir que só será possível uma viagem de ida e volta entre cada par de cidades descritas no roteiro, fazendo com que |A| = |V|(|V|-1), a solução consiste em determinar um circuito Hamiltoniano de menor custo para o grafo G.

Conhecido o número de cidades e os custos associados aos trajetos entre as mesmas, de modo que V = {1, ..., n} e, para cada par de vértices i, j ϵ V, existe um c_{ij}, ou seja, um custo associado ao arco (i, j) ϵ A. A variável de decisão do problema, $\boldsymbol{x_{ij}}$, indica se um arco de A pertence ou não à solução que está sendo produzida, assim:

$$x_{ij} = \begin{cases} 1, \text{se } j \text{ é visitado logo após } i \\ 0, \text{caso contrário} \end{cases}$$

Assim, a variável $\boldsymbol{x_{ij}}$ será binária, e só há variáveis para $\boldsymbol{i \neq j}$. O critério para a tomada de decisões é a obtenção do ciclo Hamiltoniano de menor custo. Assim a formulação matemática proposta pelos autores é a seguinte:

$$\textbf{min} \qquad \sum_{i \in V} \sum_{j \in V} \boldsymbol{c_{ij} x_{ij}} \tag{1}$$

s.a.

$$\sum_{j \in V} \boldsymbol{x_{ij} = 1} \qquad \boldsymbol{i = 1, \dots, n} \tag{2}$$

$$\sum_{i \in V} \boldsymbol{x_{ij} = 1} \qquad \boldsymbol{j = 1, \dots, n} \tag{3}$$

$$\textstyle\sum_{i \in S} \sum_{j \in S} \boldsymbol{x_{ij}} \leq \boldsymbol{|S| - 1}, \ \forall \boldsymbol{S \subset V, S \neq \emptyset} \tag{4}$$

$$\boldsymbol{x_{ij}} \in \{\boldsymbol{0, 1}\}, \ \boldsymbol{i, j = 1, \dots, n, i \neq j} \tag{5}$$

A equação (1) indica o objetivo do modelo para o PCV, que é delimitado pelo conjunto de restrições, como a (2) e (3) que asseguram respectivamente que em cada nó chega e de cada nó sai, apenas uma aresta. Isso garante que todo nó é visitado uma única vez.

A restrição (4) impede que sejam criadas sub rotas, garantindo que o modelo seja determinado como um circuito hamiltoniano. A restrição (5) indica o escopo de trabalho das equações, determinando que as variáveis sejam binárias, ou seja {0, 1}.

O PCV apresenta algumas generalizações como o Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP), definido por Balas na década de 80, que desenvolveu essa variação do PCV com base em estudos de programação de operações de uma fábrica de lâminas de aço.

O objetivo desta pesquisa era definir a sequência de ordem de processamento e sua metodologia consistia em escolher um número de lâminas, associadas às suas ordens de execução, que satisfizessem o limite inferior do peso, e que, ordenadas numa sequência apropriada, minimizasse a função de sequência. As tarefas de escolha das lâminas e das opções disponíveis para seu o sequenciamento, necessitavam ser resolvidas em conjunto (BALAS, 2001).

Para resolver esse problema Balas e Martins resolveram utilizar uma combinação de várias heurísticas para encontrar soluções próximas de um ótimo, de modo a organizar as programações diárias.

Assumindo um conjunto de vértices (cidades/pontos) e arestas entre estes, o PCVCP, pode ser associado a um caixeiro viajante que coleta um prêmio P_i, não negativo, em cada cidade *k* que ele visita e paga uma penalidade π_i para cada cidade que não visita, com um custo c_{ij} de deslocamento entre as cidades i e j. O problema encontra-se em minimizar o somatório dos custos da viagem e penalidades, enquanto inclui na sua rota um número suficiente de cidades que permitam coletar um prêmio mínimo, P_{min}, pré-estabelecido (CHAVES, 2003).

Para a definição matemática formal do problema, considere um grafo G = (V, A), como apresentado anteriormente, o qual fornece os seguintes dados:

- p_i = prêmio coletado ao visitar o cliente $i \in V$;
- π_i = penalidade pagas por não visitar o cliente $i \in V$;
- c_{ij} = custo associado ao arco $(i,\ j) \in A$;
- p_{min} = prêmio mínimo que deve ser coletado na rota percorrida.

Observa-se que a diferença dessa abordagem para a anterior é a de que nem todas as cidades necessariamente devem ser visitadas. Desse modo permanecem as variáveis x_{ij} desempenhando o mesmo papel, mas serão adicionadas as seguintes variáveis:

$$y_i = \begin{cases} 1, \text{Caso a cidade seja visitada,} \\ 0, \text{caso contrário.} \end{cases}$$

$$f_{ij}, \qquad para\ i = 1, \dots, n; j = 1, \dots, n.$$

As variáveis f_{ij} indicam a ordem em que o arco (i, j) é percorrido. Exemplo, $f_{ij} = 6$, caso (i, j) seja o sexto arco percorrido na solução. A variável f_{ij} também é conhecida como o fluxo do arco (i, j) (PEDRO, 2013).

A seguir, apresenta-se a formulação do PCVCP proposta por Balas, onde foram inseridas as restrições envolvendo as variáveis de fluxo (CHAVES et al 2003):

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} + \sum_{i \in V} \pi_i \cdot (1 - y_i) \tag{6}$$

s.a.

$$\sum_{j \in V} x_{ij} = y_i \qquad i = 1, \ldots, n \tag{7}$$

$$\sum_{i \in V} x_{ij} = y_i \qquad \text{j} = 1, \ldots, n \tag{8}$$

$$\sum_{i \in V} p_i y_i \geq p_{min} \tag{9}$$

$$\sum_{j \in V} f_{ij} - \sum_{j \in V} f_{ji} = y_i, \ \forall i \in V \tag{10}$$

$$f_{ij} \leq (n - 1) \cdot x_{ij,} \quad \forall (i, j) \in A \tag{11}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j = 1, \ldots, n, i \neq j \tag{12}$$

$$y_i \in \{0, 1\}, \qquad i = 1, \ldots, n \tag{13}$$

$$f_{ij} \geq 0, \qquad \forall (i, j) \in A \tag{14}$$

A função objetivo (6) do PCVCP é de minimizar, sendo composta basicamente por dois componentes: o custo total de deslocamento, representado pelo primeiro somatório, e o custo total das penalidades pagas por não ter visitado os determinados vértices, representado pelo segundo somatório.

As restrições (7) e (8) garantem que exatamente um arco chega e um arco sai de cada nó presente em uma rota de solução. A restrição (9) garante que o prêmio coletado na rota solução não seja inferior ao prêmio mínimo estabelecido no problema.

As restrições (11) e (14) asseguram que $f_{ij} = 0$, caso o (i, j) não esteja inserido no conjunto de solução. As restrições (10) e (11) são para eliminação de sub rotas, sendo que a restrição (10) garante que o fluxo que chega e que sai do vértice i seja igual a 1 caso o vértice seja visitado, e 0 (zero) caso contrário. A restrição (11) assegura que o fluxo máximo que pode

passar por uma aresta é o número de vértices menos 1, ou seja, (n-1), pois com n vértices o número máximo de arestas possíveis em uma solução é n - 1.

As restrições (12) e (13) asseguram a integralidade e não negatividade das variáveis x e y, respectivamente, e a restrição (14) assegura a continuidade do fluxo, isto é, eliminam as sub rotas.

2.3.2. Problemas de localização de facilidades

Outros tipos de problemas de otimização combinatória, são os Problemas de Localização de Facilidades (PLF). Os problemas de localização tratam de questões de como localizar um ou mais objetos, chamados de facilidades, onde os mesmos devem interagir com objetos que terão localização fixa. O objetivo central desse tipo de problema está em definir de forma ótima, onde determinado objeto deve ser instalado para que consiga atender o máximo de requisitos possíveis.

De acordo com Daskin (1995) os PLC tratam de decisões sobre onde devem ser localizadas facilidades, considerando os possíveis clientes que podem ser providos de modo a otimizar um determinado critério.

Pizzolato (2000) define o PLF como a escolha de uma posição geográfica para sua operação de forma que seja maximizada ou minimize uma medida de utilidade, satisfazendo diversas restrições, em particular, restrições de demanda por serviços ou produtos.

A localização de instalações de facilidades é quase sempre determinada por um fator fundamental: o fator econômico, que direciona onde deve ser instalado uma fábrica ou armazém. No varejo, a localização é direcionada pelo potencial de receitas que a região pode gerar, ao invés de custos como acontece no caso de fábricas e armazéns (BALLOU, 2006).

De modo geral, os problemas de localização de facilidade são direcionados a atender da melhor forma possível o posicionamento dos recursos de uma determinada organização, de modo a satisfazer os critérios estabelecidos pela organização.

Para Klose e Drexl (2005) os modelos são classificados de diversas formas, sendo possível a classificação por organização da região geográfica, objetivos, capacidade da facilidade, quantidade de estágios, quantidade de produtos e/ou serviços, influência da demanda e o dinamismo. Ainda do ponto de vista da classificação para Farahani, SteadieSeifi e Asgari (2010), os modelos ainda podem ser destacados quanto a quantidade de critérios a serem otimizados.

O PLF iniciou-se com Alfred Weber (WEBER, 1909) e possui outras aplicações além das relatadas aqui, como a localização de centros de comunicação em redes telefônicas, localização para instalação de subestações em redes de energia elétrica e de localização de correspondências (CHRISTOFIDES, 1975), essas por sua vez bastante desenvolvidas nos anos 70.

Os avanços com as pesquisas dos PLF vieram com Hakimi (1964), que apresentou as primeiras formulações matemáticas para o problema p-Medianas, cujo o objetivo central é localizar *p* facilidades que devem atender a *n* pontos, de forma que as distâncias entre cada ponto (facilidade) e a demanda, seja minimizada.

Esse tipo de problema é classificado como: p-Medianas não-capacitado e capacitado. O primeiro considera que cada ponto candidato a mediana pode atender a um número infinito de pontos de demanda, já no segundo considera que cada ponto tem a capacidade limitada não podendo atender mais pontos de demanda do que sua capacidade permite (TRAGANTALERNGSAK et al., 2000).

Para Araújo (2013) o objetivo do problema ou critério a ser otimizado vai depender do problema que está sendo modelado. Brandeau (1989) afirma que os problemas de localização estão classificados em seis grupos quanto a função objetivo: minimização do tempo médio de deslocamento custo médio ou maximização do lucro líquido; minimização do tempo médio de resposta; minimização do máximo tempo médio de deslocamento; minimização do tempo máximo de resposta; maximização do tempo/custo mínimo de deslocamento e problemas com outros objetivos não abordados pelos itens anteriores.

Além disso deve-se analisar os modelos de problemas utilizados Daskin (2008) define que os principais modelos de localização de facilidades estão listados em três categorias são os modelos baseados em medianas, os modelos baseados em coberturas e os que não estão classificados nessas em nenhuma das duas categorias.

Os modelos baseados em medianas são considerados pela literatura como um problema clássico de localização de facilidades e é um dos mais utilizados. Nesse tipo existe a necessidade de indicar uma distância de cobertura dentro da qual as demandas devem ser atendidas pelas facilidades a serem alocadas.

De acordo com Larson e Odoni (1981) o problema da localização de p-*medianas* objetiva localizar *p* facilidades de tal forma que a distância entre estas *p* instalações e os *n* locais de demanda seja minimizada. Observa-se que não se trata de um caso emergencial, mas sim de minimização das somas das distâncias percorridas.

Daskin (2008) corrobora afirmando que esses modelos buscam minimizar a distância ponderada média entre um nó demanda e a facilidade para o qual será alocada, normalmente utilizada no plano de distribuição de bens e/ou serviços para transporte urbano, onde deve-se minimizar o custo total de transporte.

Dessa forma o objetivo dos problemas p-Medianas (p-*Median Problem* – PMP) é de minimizar a distância média entre as demandas e a facilidade mais próxima, para a qual a demanda será alocada. No caso do PMP o custo é dado pela demanda vezes a distância de deslocamento até a facilidade que irá atender àquela demanda.

Ainda segundo Pereira (2005), os dados relevantes do problema das p-medianas são o número finito de pontos de demanda, o número finito de candidatos a instalação de facilidades, a distância entre os pontos de demanda e o número *p* de facilidades a serem instaladas.

Assim a PMP segundo Daskin (2008) apresenta a seguinte formulação matemática:

$$\boldsymbol{Min}: \sum_{\boldsymbol{i \epsilon I}} \sum_{\boldsymbol{j \epsilon J}} \boldsymbol{e_i d_{ij} y_{ij}} \tag{15}$$

s.a.

$$\sum_{\boldsymbol{j \epsilon J}} \boldsymbol{y_{ij}} = \mathbf{1} \quad \forall \boldsymbol{i} \in \boldsymbol{I} \tag{16}$$

$$\sum_{\boldsymbol{j \epsilon J}} \boldsymbol{x_j} = \boldsymbol{p}, \boldsymbol{y_{ij}} \leq \boldsymbol{x_j} \; \forall \; \boldsymbol{i} \in \boldsymbol{I}, \boldsymbol{j} \in \boldsymbol{J} \tag{17}$$

$$\boldsymbol{Vari\acute{a}veis\ de\ decis\tilde{a}o}$$

$$\boldsymbol{x_j} = \{\mathbf{0}, \mathbf{1}\} \quad \forall \, \boldsymbol{j} \in \boldsymbol{J} \tag{18}$$

$$\boldsymbol{y_{ij}} = \{\mathbf{0}, \mathbf{1}\} \quad \forall \, \boldsymbol{i} \in \boldsymbol{I} \tag{19}$$

Onde temos:

J = conjunto de possíveis facilidades (indexado por j)
I = conjunto de nós de demanda (indexado por i)
e_i = demanda no nó i
P = número de facilidades a serem instaladas
d_{ij} = menor distância entre o nó i e a facilidade j

$$x_j = \begin{cases} 1, \text{se a facilidade j for instalada} \\ 0, \text{caso contrário} \end{cases}$$

$$y_i = \begin{cases} 1, \text{se a facilidade i estiver coberta na solução} \\ 0, \text{caso contrário} \end{cases}$$

Para entender melhor o modelo, será descrito a finalidade de cada equação.

A função objetivo (15) visa minimizar a distância total percorrida pela a população (e_i) até a facilidade (j) que irá atendê-la. No conjunto de restrições, a restrição (16) impõe que a demanda de um setor (i) seja atendida somente por uma facilidade (j), já a restrição (17) determina que o número de facilidades instaladas (x_j) seja igual ao número de facilidades definidas no início a serem instaladas, essa mesma restrição impõe que y_{ij} seja maior ou igual ao somatório das facilidades eleitas que atendem à demanda i, de modo que não seja possível haver demanda atendida (y_{ij}) quando nenhum dos locais eleitos (x_j) para a solução atenda à demanda. As restrições (18) e (19) impõem os valores inteiros 0 ou 1 para as variáveis de decisão x_j e y_i.

Kariv e Hakimi (1979) definem que o problema das p-Medianas é classificado como NP-difícil*,* isto é, sendo improvável existir um algoritmo com um número de passos polinomial no tamanho da entrada para resolvê-lo, ou seja, encontrar uma solução ótima e provar sua otimalidade.

Ainda quanto à definição e formulação do problema das p-Medianas, Minieka (1977), classifica o problema em medianas e medianas absolutas. O problema das medianas absolutas, objetiva encontrar a melhor localização nos vértices e nos arcos, enquanto que o problema das p-Medianas procura a melhor solução somente nos nós do grafo.

Christofides (1975), ainda divide o problema em outros dois casos: p-Medianas puro e p-Medianas generalizado. O puro consiste em não considerar o custo de localização das facilidades, já o problema das p-Medianas generalizado introduz os custos de localização das instalações na função objetivo.

Outra classe de problemas p-Medianas que se destaca são os Problemas de Localização de Máxima Cobertura (PLMC), esse tipo de problema busca encontrar o menor número de facilidades, de forma a cobrir todos os pontos de demanda com um nível de serviço pré-determinado (facilidades).

Para Araújo (2013) o PLMC tem por objetivo localizar *p* facilidades de modo que o máximo de usuários sejam cobertos pelo perímetro de um serviço, dessa forma um usuário é considerado coberto quando está dentro do perímetro de uma facilidade.

O PLMC surgiu como alternativa para o problema de coberturas de conjuntos. Esse problema tem por objetivo localizar o menor número possível de facilidades de modo que cubra todos os usuários respeitando uma distância máxima permitida. Na prática, os recursos são limitados e não é possível obter todas as facilidades que seriam necessárias para atender todos os

usuários. No PLMC o quantitativo de facilidades é definido a priori, de acordo com os recursos disponíveis e o objetivo é cobrir o máximo de usuários dentro do perímetro de cobertura.

Para Arakaki (2002), o PLMC consiste em escolher locais para instalar facilidades de forma que o maior número de clientes (pontos de demanda) seja coberto, e determinar em qual facilidade cada cliente deverá ser atendido.

De modo geral o PLMC tem como objetivo maximizar a cobertura de uma determinada população em relação a um dado equipamento coletivo, desse modo o modelo tende a assegurar que o maior número de usuários seja atendido. Normalmente as variáveis de decisão são basicamente a distância ou o tempo de serviço, que são considerados críticos ao atendimento da demanda. Ou seja, o usuário é considerado coberto se estiver localizado dentro do raio de cobertura.

De um modo geral, o PLMC não faz restrições de capacidade e não exige que todas as áreas de demanda estejam cobertas; este problema tem como objetivo localizar p facilidades de modo que haja a máxima cobertura possível dentro da distância pré-definida.

Segue abaixo a formulação do problema descrita em (ARAKAK, 2002).

$$\boldsymbol{Max} \sum_{\boldsymbol{i} \in \boldsymbol{N}} \boldsymbol{D_i y_i} \tag{20}$$

s.a.

$$\sum_{\boldsymbol{j} \in \boldsymbol{N_i}} \boldsymbol{x_j} > \boldsymbol{y_i}, \forall\, \boldsymbol{i} \in \boldsymbol{N} \tag{21}$$

$$\sum_{\boldsymbol{j} \in \boldsymbol{M}} \boldsymbol{x_j} = \boldsymbol{p} \tag{22}$$

$$\boldsymbol{x_j} \in \{\mathbf{0}, \mathbf{1}\}, \forall\, \boldsymbol{j} \in \boldsymbol{M} \tag{23}$$

$$\boldsymbol{y_j} \in \{\mathbf{0}, \mathbf{1}\}, \forall\, \boldsymbol{i} \in \boldsymbol{N} \tag{24}$$

S: distância de serviço - a área de demanda é coberta se está dentro desta distância;

N = {1, ..., n}: conjunto de pontos de demanda;

M = {1, ..., n}: conjunto de possíveis facilidades;

Di: a demanda de população da área i;

p: número de facilidades a serem localizadas;

d_{ij} = a menor distância do nó i ao nó j;

$N_i = \{j \in J \mid d_{ij} \leq S\}$;

$$x_j = \begin{cases} 1, se\ a\ facilidade\ j\ for\ instalada \\ 0, caso\ contrário \end{cases}$$

$$y_i = \begin{cases} 1, se\ a\ facilidade\ i\ estiver\ coberta\ na\ solução \\ 0, caso\ contrário \end{cases}$$

De acordo com a definição acima, a função objetivo (20) procura maximizar a população coberta pelo problema. A restrição (21) diz que a área de demanda j ∈ J é coberto e se há pelo menos uma facilidade dentro da distância *S*. A restrição (22) limita o número de facilidades *p*, e as restrições (23) e (24) definem as variáveis de decisão a $\{0, 1\}$.

Os PLMC são aplicados nas mais diversas áreas possíveis, principalmente envolvendo a localização das facilidades dentro de áreas de emergências e na área da economia. São exemplos de aplicações do PLMC os problemas de planejamento de uma rede de ambulâncias para cobertura de uma cidade, planejamento de distribuição das viaturas de policias para cobrir ao máximo as áreas de possíveis eventos perturbadores da ordem social; também são aplicados na localização de sirenes de aviso, que orienta determinada população caso ocorra um sinistro; dentre outros (ARAKAKI, 2002).

2.3.3. Problemas de designação (A*ssignment Problem*)

O Problema de Designação/Atribuição é um caso específico de um Problema de Transporte, característico de Programação Combinatória. O Problema de Designação consiste em designar cada uma das origens a cada um dos destinos, de maneira ótima.

Esse tipo de problema também recebe a denominação de Problema Geral de Atribuição (PGA). De um modo geral, o problema pode ser entendido da seguinte forma: um conjunto de n tarefas deve ser atribuído a um conjunto de m agentes, sendo necessário que todas as tarefas sejam atribuídas e que cada uma delas seja endereçada a apenas a um agente, de tal forma que o custo total da atribuição seja minimizado (BALACHANDRAN, 1976).

O Problema Geral de Atribuição é classificado como um problema NP-Difícil, o que significa que ainda não existe algoritmos de complexidade polinomial para resolvê-lo de forma ótima (SAHNI & GONZALEZ, 1976 *apud* CHU & BEASLEY, 1997).

A relevância do PGA e as dificuldades relacionadas à busca de suas soluções têm justificado o emprego de vários métodos de otimização como heurísticas baseadas em teoria de con-

juntos segundo Cattrysse et al. (1994 *apud* Wilson, 2005), heurísticas aproximadas desenvolvidas com base na estrutura do problema (NAUSS, 2003), relaxação lagrangeana (NARCISO & LORENA, 1999), Busca Tabu (OSMAN, 1995 *apud* DIAZ & FERNANDEZ, 2001), e Algoritmos Genéticos (CHU & BEASLEY, 1997), dentre outros.

Pode-se entender que o PGA da seguinte forma, dado A o conjunto de agentes, T o conjunto de tarefas, c_{ij} é o custo de atribuir a tarefa $j \in T$ ao agente $i \in A$, a_{ij} são os recursos necessários ao agente $i \in A$ para desempenhar a tarefa $j \in T$, b_i é a capacidade do agente $i \in A$ e x_{ij} é a variável binária de decisão a qual assume o valor "1" se a tarefa $j \in T$ é atribuída ao agente $i \in A$ ou "0" em caso contrário. Desta maneira, o PGA pode ser apresentado como (Yagiura et al., 2006):

$$\boldsymbol{Min} \sum_{\boldsymbol{i \in A}} \sum_{\boldsymbol{j \in T}} \boldsymbol{a_{ij} x_{ij}} \tag{25}$$

s.a.

$$\sum_{\boldsymbol{j \in T}} \boldsymbol{a_{ij} x_{ij}} \leq \boldsymbol{b_i}, \quad \forall \, \boldsymbol{i} \in \boldsymbol{A} \tag{26}$$

$$\sum_{\boldsymbol{i \in A}} \boldsymbol{x_{ij}} = \boldsymbol{1}, \forall \, \boldsymbol{j} \in \boldsymbol{T} \tag{27}$$

$$\boldsymbol{x_{ij}} \in \{\boldsymbol{0}, \boldsymbol{1}\}, \quad \forall \, \boldsymbol{i} \in \boldsymbol{A}, \forall \, \boldsymbol{i} \in \boldsymbol{T} \tag{28}$$

Na modelagem do PGA, a equação (25) é a função de custo que deve ser minimizada. Nas equações (26) assegura que a capacidade dos agentes não é violada, a (27) restrição assegura que cada tarefa é atribuída a apenas um agente e a (28) restrição define que cada variável x_{ij} é binária.

Além dos PGA, existem diversos tipos de problemas de atribuição, na sua forma mais simples, como o problema que não envolve restrições de capacidade. Assim, apresenta-se quando m = n, ou seja, quando o número de agentes é igual ao número de tarefas. Dessa forma o problema é conhecido como Problema Linear de Atribuição (PLA) ou, em inglês *Linear Assignment Problem.*

O PLA é um caso especial de outro problema de otimização, conhecido como Problema de Transporte. Devido à sua estrutura especial, o PLA pode ser resolvido eficientemente em um tempo polinomial aceitável, por diversos métodos já conhecidos (KUHN, 1955; HUNG; ROM, 1980; WRIGHT, 1990).

Um outro tipo de problema de designação conhecido é o Problema Quadrático de Atribuição (PQA) ou *Quadratic Assignment Problem* (QAP) em inglês. O problema ocorre quando, para designar objetos a locais, deve-se levar em conta as distâncias entre os pares de locais e os fluxos de algum tipo de demanda entre os pares. O PQA consiste em encontrar uma alocação de custo mínimo, sendo os custos obtidos pela relação distância versus fluxo (SENNE, LORENA e SALOMÃO, 2007).

Proposto originalmente por Koopmans e Beckmann (1957) como um modelo matemático relacionado a atividades econômicas, o PQA são aplicados em inúmeros problemas práticos, Steinberg (1961) o utilizou para minimizar a quantidade de ligações entre componentes de placas de circuitos eletrônicos; Francis e White (1974), no desenvolvimento de um *framework* de decisões de alocação de uma nova facilidade (postos policiais, supermercados, escolas) que atenda a um dado conjunto de clientes; também são aplicados em problemas de escalonamento de horário, em análise estatística, em planejamentos de hospitais, dentre muitas outras como pode ser observado em Loiola et al. (2004).

Outra característica de problema de atribuição ocorre quando elementos de um número variável de conjuntos devem ser atribuídos mutuamente. Este problema é conhecido como Problema de Atribuição Multidimensional (PAM) ou *Multidimensional Assignment Problem* em inglês (PIERSKALLA, 1968; GRUNDEL et al., 2005).

Existem diversas situações relatadas na literatura que envolvem o PAM, as quais podem ser destacar a determinação de trajetórias de partículas elementares (PUSZTASZERI et al., 1996) e a determinação de células fibroblásticas a partir de sequências de imagens biomédicas (KIRUBARAJAN et al., 2001).

Outras variações conhecidas para o problema de atribuição são: o Problema de Atribuição Dinâmica (PAD) ou *The Time Slot Assignment Problem* - TDMA (Problema de Atribuição de Horário). Que corresponde a designação de agentes a tarefas, em que deve ser feita de forma dinâmica sobre períodos. O PAD, corresponde a encontrar, em uma matriz m × n de variáveis aleatórias, k elementos tais que sua soma seja mínima; e o Problema de Atribuição Axial, que consiste em encontrar um clique (subgrupo completo) de peso mínimo para um grafo tripartido completo. Uma revisão sobre os principais problemas de atribuição pode ser encontrada em Burkard (2002).

Os Problemas de atribuição são aplicados nas mais diversas áreas e tem grande importância prática, estando presentes desde o planejamento de tarefas diárias de uma pessoa a grandes organizações, de modo que estas utilizem melhor os seus recursos.

Suas aplicações práticas são as mais diversas possíveis como pode ser observado a seguir: atribuição de tarefas em redes de computadores (BALACHANDRAN, 1976) na distribuição de tarefas entre processadores de um sistema de computação paralela (BOKHARI, 1987); na localização de facilidades (ROOS e SOLAN, 1977); na distribuição de pacientes que necessitam de viagens médicas (RULAND, 1999); no carregamento de Contêineres (HUNG e FISK, 1979); no escalonamento de máquinas em paralelo (LENSTRA *et al*., 1990); na atribuição de frequências em redes de comunicação celular (FISCHETTI *et al*., 2000; WANG e GU, 2004), no roteamento de veículos (FISHER e JAIKUMAR, 2006); na comparação de estoque/documentos (DAWANDE e KALAGNAMAM, 1998); no escalonamento de recursos (MAZZOLA *et al*., 1989) e no escalonamento de multiprocessadores (TUREK *et al*., 1992). Além disto, existem muitos problemas de decisão que, não sendo diretamente um problema de atribuição, contêm um problema de atribuição como subproblema.

2.3.4. Problemas de k-servos

O problema dos K-Servos foi proposto inicialmente por Manasse, McGeoch e Sleator (1988), tal trabalho serviu de base para muitos temas. O modelo e o conceito dos K-Servos têm servido como um catalisador para o desenvolvimento da análise competitiva (BORODIN; EL-YANIV, 1998).

Para Bartal e Koutsoupias (2004) o problema dos k-servos é, provavelmente, o problema *on-line* mais influente. O Problema dos K-Servos (PKS) é de fácil descrição, onde denominam-se servos as unidades de atendimentos das demandas. A demanda é a indicação do local onde os atendimentos deverão ser realizados, ou seja, os servos devem atender às solicitações originadas nestes locais de demanda.

Para isso considera-se um espaço M habitado por k servos localizados em pontos de M não necessariamente distintos. Inicialmente, cada servo é posicionado em algum ponto de M. No decorrer do tempo, pedidos por serviço chegam de algum ponto do espaço métrico. A meta é servir a sequência de pedidos. Um pedido j é simplesmente um ponto do espaço métrico, e servir j implica em deslocar um servo ao ponto j. Associado ao deslocamento do servo localizado em i para um local j existe um custo de atendimento proporcional à distância d(i, j) do servo em i à demanda em j.

Dessa forma o objetivo é minimizar a distância total percorrida pelos servos no atendimento de todas as requisições (vetores de demanda). Portanto, um algoritmo que resolva o problema dos k-servos de maneira *on-line* deve decidir qual servo será deslocado para atender a cada uma das requisições, de modo que a distância percorrida seja a menor possível.

De acordo com Lima Jr (2002) deve-se observar se é de conhecimento antecipado a lista de todas as requisições, visto que caso isso aconteça o problema é dito como *off-line* e existe um algoritmo com estratégia ótima com tempo polinomial de ordem $O(n^2)$, para n igual a ordem de pedidos na lista (CHROBAK et al. 1991).

Ainda de acordo com Lima Jr (2002) caso não se conheça a lista prévia de todas as solicitações o problema é dito *on-line*, dessa forma os algoritmos irão recebendo os dados na medida que eles vão surgindo. Assim, a resposta desses algoritmos é feita da mesma forma, ou seja, após *t* unidades de entrada ele deve produzir a *t-ésima* unidade de saída. Um algoritmo online não produz conhecimento completo para a tomada de decisão, visto que são produzidos sobre uma parte do conhecimento e não sobre o todo, assim, produz uma solução aproximada do ótimo, o algoritmo *on-line* é tido competitivo se sua performance é próxima do algoritmo *off-line* em cada uma de suas entradas.

Alguns algoritmos são capazes de solucionar o problema dos K-Servos, dentre eles podemos destacar o algoritmo de Sleator e Tarjan (1988), o algoritmo se utiliza de uma estratégia gulosa que é, basicamente, escolher os movimentos que permitam que os diversos servos se desloquem de forma igualitária. O algoritmo mantém a distância total percorrida por cada servo k na variável D_k.

Se existe uma demanda σ_i a ser atendida, o servo k a ser escolhido para fazer o atendimento será aquele que minimize a expressão $D_k + d(j_k, \sigma_i)$, onde $d(j_k, \sigma_i)$ é a distância percorrida por um servo k localizado em um nó j até a requisição σ_i.

Borodin e El-Yaniv (1998) definem que soluções mais elaboradas cujas taxas de competitividade já foram estabelecidas, como os algoritmos *Harmonic e Work Function.* Bansal *et al.* (2011) utilizando um algoritmo polilogaritimico competitivo aleatório para o problema dos K-Servos em um espaço métrico finito arbitrário, obteve uma taxa de competitividade para o problema.

2.3.5. Problemas de fluxo de rede

Os fluxos em redes estão presentes no cotidiano das pessoas, podem ser encontradas em redes físicas, como telefônicas, das companhias de energias e telecomunicações, além de oleodutos, distribuição de água, transportes, dentre outros.

A teoria dos grafos teve início no século XVIII na Alemanha com Leonhard Euler que precisava resolver o seguinte problema: a cidade de Königsberg (atual Leningrado) era cortada pelo rio Pregel, que possuía duas ilhas (figura 1). Como era muito complicado fazer o transporte de cargas e pessoas através de barcos, algumas pontes foram construídas para auxiliar neste deslocamento entre as ilhas e as duas margens. Após algum tempo as pessoas começaram a se perguntar se era possível sair de sua casa, passar por cada ponte exatamente uma vez e voltar para a segurança de seu lar (BRADLEY e SANDIFER, 2007).

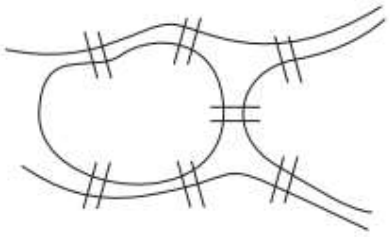

Figura 1 - Mapa de Königsberg
Fonte: Holanda (2011)

Para resolver o problema, Euler o modelou o problema real como uma rede de pontos ligados por linhas entre si, aos pontos que corresponderiam aos territórios ele atribuiu o nome de nós, e para as linhas que corresponderiam as pontes deu o nome de arcos. Assim, foi possível determinar uma abordagem padronizada para solucionar o problema somente utilizando um par ordenado formado por dois conjuntos, um de nós e outro de arcos, ao qual atribui-se o nome de grafo.

Um grafo é um par *(V, E),* onde *V* é um conjunto finito e *E* é um conjunto de pares ordenados de elementos de *V*. Os elementos de *V* são chamados nós, os elementos de *E* são chamados arcos. Para cada arco *(i; j)*, o nó *i* é a ponta negativa ou ponta inicial de *(i; j)* e *j* é a ponta positiva ou ponta final de *(i; j)*. As pontas inicial e final de cada arco são distintas; portanto, esse tipo de grafo não tem “laços”.

Os grafos formam redes de conexões, que apresentam determinados fluxos que se denomina de fluxos de rede, esses fluxos apresentam problemas de caminhos nos fluxos, para isso a literatura traz alguns problemas clássicos de fluxos como os apresentados a seguir.

O *Problema do Caminho Mínimo (PCM)* tenta responder o seguinte questionamento: qual é a melhor forma de percorrer uma rede indo de um dado ponto a outro, com o menor custo possível? Assim compreendem a determinação do caminho ou rota de menor tamanho (distância, tempo ou um custo qualquer) entre dois nós de uma rede.

Na teoria de grafos, o PCM irá consistir na minimização do custo de travessia de um grafo entre dois nós (vértices) ou mais, custo este dado pela soma dos pesos de cada aresta percorrida. Dessa forma os PCM's se utilizam de algoritmos para determinar a rota de menor tempo, distância ou custo entre um par ou vários pares de origem e de destino de uma rede.

O algoritmo para o PCM foi desenvolvido inicialmente por Dijkstra em 1959 e posteriormente Dantizg (1960) e Nicholson (1960) desenvolveram o algoritmo de "duas árvores" de Dijkstra (*Dijkstra two-tree algorithm*), cuja ideia é construir árvores de caminhos mínimos a partir de um nó origem *s* e de um nó de destino *t* simultaneamente. Ao chegar a um nó comum o caminho mínimo entre *s* e *t* foi encontrado. Esta ideia mostrou ser interessante do ponto de vista de tempo computacional, conforme foi constatado posteriormente por Helgason em 1988 (CAMPOS, 2001).

Em sua forma mais genérica o algoritmo determina o trajeto mais curto entre um dado nó considerado como origem e todos os outros, dessa forma o algoritmo de Dijkstra é o mais simples de cálculos de caminhos, apesar de, até aos dias de hoje, vários algoritmos terem sido desenvolvidos.

Esse algoritmo reduz o tempo de processamento e as capacidades necessárias em nível computacional para o cálculo do caminho ótimo, estabelecendo um equilíbrio através do cálculo do trajeto que é muito próximo do caminho ótimo, computacionalmente possível de ser calculado, partindo a rede em nós, sendo que os trajetos entre eles são representados por linhas. Adicionalmente, cada uma das linhas tem um custo associado, representado pelo seu comprimento até alcançar o nó seguinte (SILVA, 2009).

Assim, o grafo para o problema de Dijkstra é apresentado por um grafo valorado $G = \{\mathrm{N}, R\}$, com arcos de custos dados em uma matriz denominada de C=[c_{ij}]. O PCM, visa encontrar o caminho mínimo de um vértice inicial específico $s \in \mathrm{N}$ para um vértice final específico

$t \in N$, sendo que este caminho existe, isto é, contanto que $t \in R(s)$, onde $R(s)$é o conjunto de alcance do vértice *s*.

Para Steenbrick (1974 *apud* Janson 1987) existem diferentes formulação para o PCM como: de um nó para outro nó; de um nó para todos os outros nós da rede; entre todos os nós da rede, e k-caminhos mínimos entre dois nós.

Outro problema clássico de fluxo é o Problema de Fluxo Máximo (PFM), esse problema consiste em uma rede com arcos capacitados, que visa responder o seguinte questionamento: como e qual é o máximo de fluxo o que possível de se enviar de um dado ponto a outro da rede, respeitando a capacidade dos arcos?

Os PFM's se referem às situações em que se deseja avaliar a quantidade máxima de fluxo que pode ser enviada de um nó de origem a um nó de destino na rede. Existem diversas situações que são abordadas por esse problema como: maximizar o fluxo de uma rede de distribuição (cadeia de suprimentos) de um conjunto de fábricas (fornecedores) para seus clientes (lojas); maximizar o fluxo de óleo (ou água) através de um sistema de oleodutos (aquedutos); maximizar o fluxo de veículos através de uma rede de transportes, dentre outros.

Para solução desse tipo de problema se faz necessário a abordagem com o algoritmo de fluxo máximo. O primeiro algoritmo desta natureza foi desenvolvido por Ford e Fulkerson (1956). Mais tarde surgiram outros que visavam melhorar o desempenho computacional deste algoritmo. Algumas modificações foram também introduzidas na forma de trabalhar a rede, como foi o caso do algoritmo de Dinic (1970 *apud* Syslo 1983) que introduziu o conceito de redes em camadas e Mallhotra (1978 *apud* Syslo 1983) que introduziu o conceito de potencial de um nó.

A ideia central de um algoritmo de fluxo máximo é encontrar "caminhos de aumento de fluxo" de uma origem S para um nó de destino T e alocar nestes caminhos a maior quantidade de fluxo possível.

Para Sanches e Manic (2008) o problema de fluxo máximo consiste em, dada uma rede orientada, determinar o máximo de fluxo que pode ser enviado de um nó que caracteriza-se por fonte (denotado por *s*) até o destino (denotado por *t*), de forma que o fluxo x_{ij} entre cada par de nós (*i, j*) não exceda a capacidade u_{ij} e de forma que o balanço de fluxo em cada nó da rede com exceção de *s* e *t* seja mantido (ou seja, o fluxo que entra em cada nó, com exceção de s e t, é igual ao fluxo que sai deste nó). Mais formalmente, queremos maximizar *v*, sujeito as seguintes condições.

$$Max\ v \tag{27}$$

s.a.

$$\sum_{\{j:(i,j)\in E\}} x_{ij} - \sum_{\{j:(j,i)\in E\}} x_{ji} = v, \qquad i = s \tag{28}$$

$$\sum_{\{j:(i,j)\in E\}} x_{ij} - \sum_{\{j:(j,i)\in E\}} x_{ji} = 0, \qquad i \in E \setminus \{s,t\} \tag{29}$$

$$\sum_{\{j:(i,j)\in E\}} x_{ij} - \sum_{\{j:(j,i)\in E\}} x_{ji} = -v, \qquad i = t \tag{30}$$

$$0 \leq x_{ij} \leq u_{ij} \,/\, (i,j) \in E \tag{31}$$

Onde:

x_{ij} − *Fluxo no arco*

u_{ij} − é *a capacidade de fluxo no arco*(i,j)

v − é *o fluxo máximo na rede*

A restrição (27) indica que deverá ser maximizado o problema, a restrição (28) indica o fluxo máximo na rede, já as restrições (29), (30) e (31) estão associadas à lei de conservação de fluxo.

Outro clássico dos problemas de fluxo de rede é o Problema de Fluxo com Custo Mínimo (PFCM), este problema considera um dado custo por unidade de fluxo em uma rede (nos seus arcos), quer dizer com arcos capacitados. A necessidade de enviar unidades de fluxo alocados em determinados nós (oferta/produção) para outros nós (demanda/consumo) irá demandar custos em cada fluxo, então questiona-se, como fazê-lo com o menor custo possível?

Percebe-se que nos problemas anteriores tratou-se de verificar a quantidade máxima de fluxo que poderia ser enviada de uma origem *s* para um destino *t,* não havendo custos envolvidos. Para solução desse tipo de problema recorre-se aos algoritmos de custo mínimo, esses tratam do problema de enviar uma quantidade qualquer de fluxo *v* de *s* para *t* numa rede na qual todos os arcos *(i,j)* tem uma capacidade ou limite superior *u(i,j)* tanto quanto um custo *c(i,j)* associado a estes arcos (CAMPOS, 1997).

Os custos associados ao problema podem ser de vários tipos como: tempo, valor monetário, distância, consumo de combustível ou até mesmo uma combinação destes. Em suma o objetivo dos algoritmos de custo mínimo é, então, encontrar os caminhos de fluxo que minimizam o custo total, ou seja:

$$\min y = \min\left(\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} f_{ij}\right)$$

onde c_{ij} representa o custo no arco (i,j) e f_{ij} o fluxo alocado neste mesmo arco.

As aplicações do PFCM são os mais diversos possíveis, como no planejamento de transportes, com a finalidade de diminuir o valor do custo associado ao frete, no transporte de cargas perecíveis que ajudaria na diminuição do tempo médio de transporte e assim incorreria menor risco de perda da carga em transportes urbanos, de modo a diminuir o valor do ingresso para determinadas rotas, dentre outras.

Alguns algoritmos foram desenvolvidos para solucionar esse tipo de problema como Ford/Fulkerson, "*Out of Kilter"* (*apud* Plane 1971, Minieka 1992) e o de Busacker e Gowen (*apud* Syslo 1983).

2.4. Modelos e métodos matemáticos

Os modelos são abstrações, representações de um sistema real. De acordo com Hillier e Lieberman (2002) um modelo é uma abstração do sistema real, na qual somente aspectos relevantes para uma determinada análise deste sistema são considerados. Visto que é praticamente impossível que um modelo contenha todos os detalhes de um sistema.

Dessa forma um modelo é a representação de um sistema a ser elaborado ou existente, com o intuito de se realizar uma investigação experimental ou tomada de decisão em um menor espaço de tempo e custo possível. Assim, percebe-se que os modelos, apesar de serem uma representação detêm de uma série de informações do sistema real. Portanto, assim diferentes modelos podem ser formulados em virtude da sua classificação.

Os modelos podem ser classificados em físicos, análogos, esquemáticos e matemáticos. Os modelos físicos são representações com equivalências geométricas, visuais, ou até mesmo ampliações ou duplicatas feitas em escala dos sistemas reais. Normalmente esses modelos são complementações de modelos matemáticos.

Os modelos análogos, deriva da palavra grega analogia, que significa proporção, semelhança visual do sistema real, em um sistema computacional os modelos são análogos ao sistema real. Já os modelos esquemáticos têm a característica de descrever de forma gráfica uma

situação ou processo, assim um modelo esquemático reduz um sistema real a um gráfico ou diagrama.

Os modelos matemáticos representam os princípios da situação sendo estudada. Embora os símbolos utilizados em sua representação sejam mais complexos que os verbais, eles fornecem um grau de abstração mais alto e com maior precisão (SODRÉ, 2007).

O mesmo ainda afirma que os modelos são de grande importância para a ciência e a engenharia em geral, especialmente a partir do instante em que o computador digital veio simplificar significativamente a tarefa do cálculo numérico.

Em sua maior parte, os modelos matemáticos são utilizados para predição ou controle. Modelos matemáticos dirigidos para o estudo de sistemas operacionais diferem dos usados tradicionalmente em ciências físicas, o primeiro geralmente estuda os fatores sociais e econômicos, para esses modelos utiliza-se os elementos probabilísticos para explicar seu comportamento, já o segundo são modelos para explicar operações, agregando duas classes de variáveis, as controladas pelo decisor e as que não são possíveis de controlar. Essas representações são denominadas de modelos matemáticos (D'AMBRÓSIO, 2003).

Fireman e Santos (2014) ratificam afirmando que um modelo matemático, é uma forma específica de representação, se vale de objetos matemáticos, tais como as funções, os vetores e as figuras geométricas.

Para Bassanezi (2002) os modelos matemáticos são representações que utilizam símbolos e relações matemáticas com a finalidade de analisar e representar um sistema real. Normalmente os modelos matemáticos são classificados em analítico, simulação e numérico.

O modelo analítico são formulas ou esquemas gráficos que mostram exatamente ou de forma aproximada como se comporta o sistema analisado, já os modelos de simulação são passos ou algoritmos que representam o comportamento do sistema em um intervalo de tempo, com início meio e fim, e por fim os modelos numéricos são procedimentos computacionais cuja as soluções são obtidas de forma interativa.

Percebe-se então que o modelo matemático em resumo é uma representação ou interpretação de um sistema real ou partes desse sistema, servindo para tomada de decisão, com diferentes classificações. Mas esses modelos necessitam de procedimentos para extrair informações importantes a tomada de decisão, assim esses se utilizam de métodos matemáticos.

Os métodos são um conjunto de passos, procedimentos, dessa forma os métodos matemáticos são rotinas ou conjunto de passos com a finalidade de solucionar um problema por meio matemático.

CAPÍTULO 3

CONTEXTO DO PROBLEMA E ABORDAGEM PROPOSTA

3.1. Definição do problema proposto

O Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP), referido na literatura inglesa como *Prize Collecting Traveling Salesman Problem* (PCTSP), é uma variação do Problema do Caixeiro Viajante (PCV), na qual um caixeiro coleta um prêmio (p_w) não negativo, em cada cidade *w* que visita e sofre uma penalidade (γ_t) para cada cidade *t* que não visita, assumindo também um custo c_{ij} de deslocamento entre as cidades i e j.

Desta forma o problema está em minimizar o somatório dos custos da viagem e penalidades, enquanto inclui na sua rota um número suficiente de cidades que o permita coletar um prêmio mínimo (p_{min}) pré-estabelecido.

O PCVCP foi desenvolvido inicialmente por Balas em 1989 para a programação de fábricas que produziam lâminas de aço. Em seguida, vieram outras aplicações com Goemans et al (1992) que desenvolveram um procedimento de aproximação para uma versão do PCVCP no qual o prêmio mínimo a ser coletado foi removido e o objetivo passou a ser simplesmente minimizar o custo e as penalidades.

Em consonância com este cenário está a rede metropolitana de conexão de alta velocidade da cidade de Mossoró, denominada GigaMossoró, que tem por objetivo interligar centros de pesquisa com uma conexão de alta velocidade, inserindo prioritariamente as três principais instituições públicas de ensino superior da cidade.

A proposta é que a rede seja construída em formato de anel, com tecnologia que fará a conexão em ambos os sentidos, garantindo a sua redundância. Com o orçamento inicial o projeto garante a construção de uma infovia com aproximadamente 24km, como pode ser observado na figura 2.

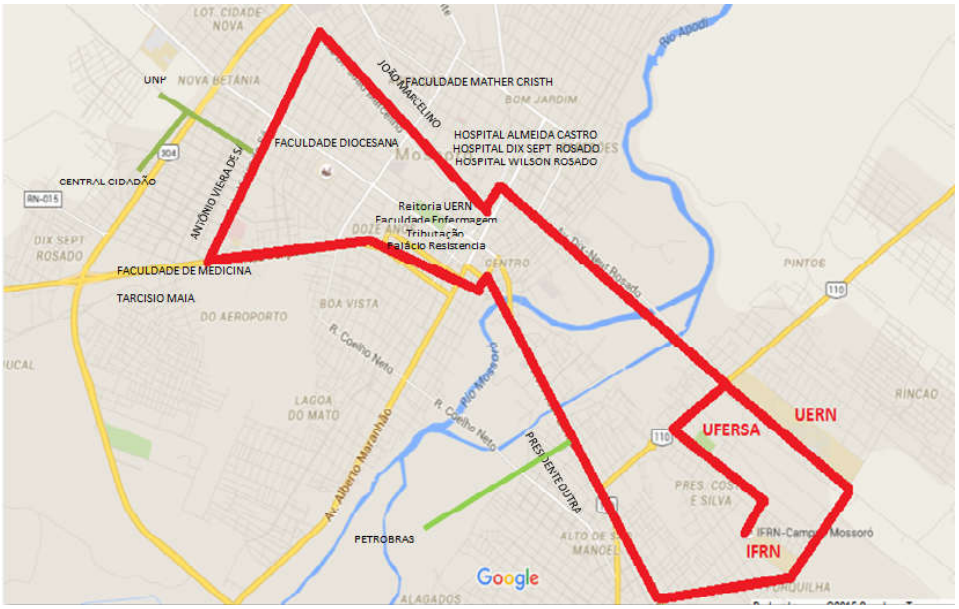

Figura 2 - Proposta inicial da rede GigaMossoró
Fonte: Conselho Diretor da Rede GigaMossoró, 2018.

O trajeto apresentado na Figura 2 é a proposta inicial da rede, de modo que a ideia é que parceiros se conectem à mesma, podendo assim usufruir de suas funcionalidades, além de contribuir na sua manutenção. São exemplos de potenciais parceiros: entidades públicas, associações, fundações, sociedades civis de fins não lucrativos e pessoas jurídicas de qualquer natureza, que tenham relevância para a oferta e demanda de tecnologia na região metropolitana da cidade de Mossoró.

A proposta inicial apresentada na figura 2 é uma idealização dos coordenadores da rede, onde os mesmos a projetaram a mesma sem qualquer fundamento cientifico, apresentando como base apenas suas vivências e expertises.

O problema como descrito apresenta características do problema do caixeiro viajante visto que os pontos (parceiros) se caracterizam como um conjunto de *n* cidades a serem visitadas, mas com uma variação, pois durante o percurso algumas anuências devem ser consideradas:

- A rede deve passar obrigatoriamente pelas instituições de ensino superior que integram o conselho gestor (UERN, UFERSA e IFRN), as quais se caracterizam como clientes obrigatórios da rede;
- A rede deve atingir o máximo de pontos de possíveis candidatos a clientes da rede;

- O problema deve levar em consideração a distância entre os candidatos a clientes e a rede;
- A rede não deve ultrapassar 24km;
- Deve haver uma ponderação mínima a ser coletada durante o percurso.

Diante das restrições que permeiam o problema entende-se que o mesmo tem as características de um PCVCP uma variante do PCV, visto que a cada ponto visitado será recebido um prêmio e para cada ponto não visitado será atribuído uma penalidade. Além disso deve-se durante seu percurso coletar um prêmio mínimo.

Os PCVCP como já descritos aqui apresentam alto nível de complexidade, não sendo possível obter uma solução com tempo computacional razoável. Dessa forma é necessário entender qual método de resolução é apropriado para o mesmo.

3.2. Modelo e método de resolução proposto

Como modelo matemático para o problema proposto nesta pesquisa será apresentada uma adaptação ao modelo originalmente concebido por Balas e modificado por Chaves (2003) para inclusão das variáveis de fluxos, conforme descrito na secção 2.4.

A adaptação aqui proposta diz respeito principalmente à necessidade de ponderação (punição ou bonificação) relativa à distância (ou grau de dificuldade de conexão) dos potenciais clientes, bem como, do estabelecimento do comprimento máximo do layout do anel da rede.

Dado que o problema proposto é reconhecidamente um problema de alta complexidade computacional, ou seja, comprovadamente NP-*difícil*, será utilizada uma abordagem via Metaheurística com base em um algoritmo genético (GA).

O conceito de GA surge de mecanismos de analogia entre a evolução da natureza e a necessidades de resolver problemas complexos, visto que um conjunto de soluções candidatas deveriam ser evoluídas (MEGALE, 2015).

Os GA aplicam o conceito de sobrevivência do mais apto para encontrar soluções ideais ou quase ótimas para problemas de otimização combinatória. Na prática cada indivíduo possui um cromossomo, uma série de genes, que codificam uma solução, e a partir desta solução o cromossomo associa-se com outros genes afim de melhorar o próximo cromossomo.

Desta forma os algoritmos genéticos desenvolvem um conjunto de indivíduos que compõem uma população ao longo de várias gerações. Em cada geração, uma nova população é

criada pela combinação de elementos da população atual para produzir descendentes que irão compor a próxima geração. A mutação aleatória também ocorre em algoritmos genéticos como um meio para escapar do aprisionamento em ótimos locais.

Dessa forma, com os AGs cada indivíduo da população irá representar uma solução viável para o problema proposto. A qualidade de cada uma das soluções é medida em função da aptidão, a busca segue através de um certo número de gerações, nas quais a contribuição de cada indivíduo para a geração seguinte é proporcional a sua aptidão (DOWSLAND, 1996).

Dentre os operadores tradicionais dos AGs pode-se destacar os seguintes (HOLLAND, 1992; BEAN, 1994; REEVES, 1997):

- Cromossomo: é uma solução candidata de um determinado problema. Normalmente o cromossomo é codificado como um vetor em $\mathbb{Z}^n$, onde n é o número de elementos que compõem a solução;
- Gene: é a unidade mínima de um cromossomo.
- População: é o conjunto de cromossomos;
- Prole: é o resultado da reprodução de 2 cromossomos;
- Reprodução: é o processo de geração de uma nova população;
- Operador de recombinação: realiza a troca de um conjunto de genes de um cromossomo por um conjunto de genes de um outro cromossomo. O operador *one point crossover*, gera duas proles a partir de dois cromossomos-pai e um ponto de *crossover* (POLI; LANGDON, 1998);
- Operador de seleção: é o método de escolha de cromossomos para recombinação. O método *roullete wheel* (Roleta), gera uma distribuição probabilística na qual a probabilidade de um cromossomo ser selecionado é proporcional à sua aptidão (KUMAR, 2012);
- Aptidão: indica a qualidade de um cromossomo em comparação com outros cromossomos dentro de sua população;
- Função objetivo: é a função responsável por atribuir uma aptidão a um cromossomo;
- Codificação: é a estratégia utilizada para construir um cromossomo a partir de informações específicas do problema a ser tratado;
- Decodificação: é a estratégia utilizada para criar um algoritmo determinístico que recebe como entrada um cromossomo e associa a ele uma solução para o problema em questão.
- *Crossover:* é o operador que procura obter uma solução de boa qualidade através da combinação de duas soluções da população.

- Mutação: é o operador que visa acrescentar diversidade à busca e é feita através de um método independente que procura gerar indivíduos distintos dos demais indivíduos da população.

Cabe ressaltar que o tamanho da população afeta a eficácia e o desempenho dos AG, visto que por um lado, aptidão das soluções encontradas tendem a ser muito baixa se a população for muito pequena e, por outro, o desempenho do algoritmo tende a ser ruim caso a população seja muito elevada. De fato, não existe um consenso sobre o tamanho ideal da população, porém alguns estudos empíricos demonstraram que uma população com tamanho variando entre 10 a 160 apresenta alta taxa de probabilidade de encontrar resultados aceitáveis (REEVES, 1997; FOGEL, 1995, WHITLEY, 1994).

Para o problema em questão, inicialmente foi confeccionado um algoritmo genético para realizar testes com base na matriz inicial dos pontos fixos do problema. O algoritmo foi construído conforme a estrutura descrita na figura 3.

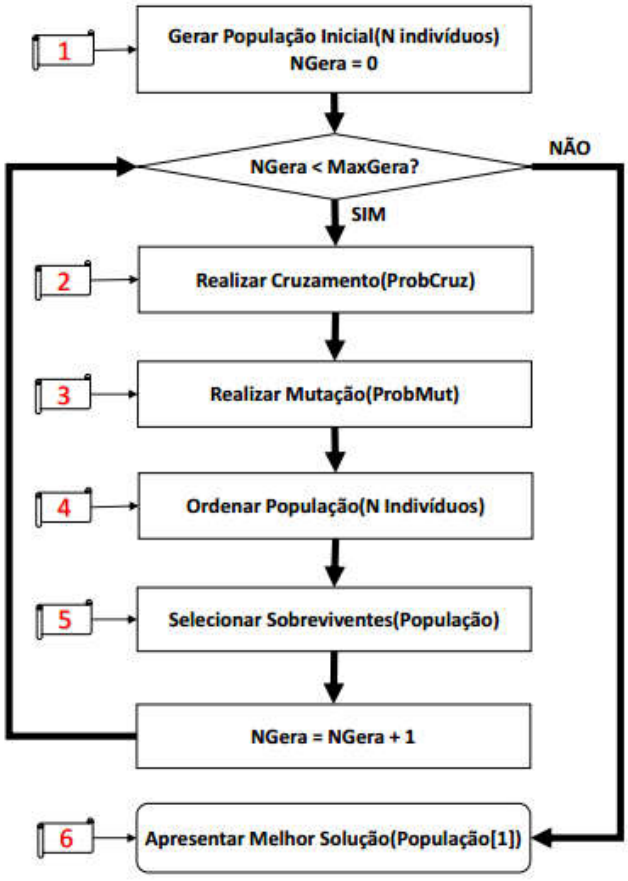

Figura 3 - Estrutura do Algoritmo Genético
Fonte: Autores 2018.

Dessa forma a população inicial pode ser gerada aleatoriamente e será composta por N indivíduos, onde cada indivíduo é uma solução para o problema. NGera é o contador da geração corrente (1). Em seguida o operador de cruzamento[1] deverá ser realizado com probabilidade ProbCruz, entre dois indivíduos escolhidos aleatoriamente na população. O cruzamento é uma forma de gerar soluções elitizadas (2). O operador de mutação[2] deverá ser realizado com probabilidade ProbMut, em um indivíduo escolhido aleatoriamente na população. A mutação é uma forma de dar chance para soluções aleatorizadas (3). Posteriormente a população deverá ser avaliada e ordenada com base no valor da função objetivo, de forma que o melhor indivíduo esteja na primeira posição da população (4), para que os sobreviventes da próxima geração sejam obtidos pelo uso de uma roleta viciada[3] (5), e assim, a melhor solução é o indivíduo no topo da população (6) (CHAVES, 2003).

Observa-se que o GA apresenta uma solução simples sem incremento, uma vez que não foram leva em consideração alguns parâmetros do problema. Dessa forma surge a necessidade de se estudar e entender variantes dos GAs, visto que esses apresentam resultados melhores e conseguem agregar mais detalhes aos modelos e problemas.

Uma variante do modelo de GAs são os Algoritmos Genéticos de Chaves Aleatórias (*Random Key Genetic Algorithm – RKGA*), os RKGA foram inicialmente introduzidos por Bean (1994) com o propósito de resolver problemas de sequenciamento.

Esses algoritmos têm sido aplicados para diversos problemas de otimização (NORONHA; RESENDE; RIBEIRO, 2011; RESENDE, 2012; GOULART et al., 2011; PRASETYO; FAUZA G.; LEE, 2015). Eles utilizam o conceito e a robustez de chaves aleatórias e um decodificador eficiente para a solução do problema de otimização, representando as possíveis soluções. Esse fato vem corrigir determinadas dificuldades que os AGs tradicionais costumam ter, que é a viabilidade das soluções durante a operação de *crossover.*

Nos algoritmos genéticos os indivíduos são chamados de cromossomos, esses são soluções geradas de forma aleatória para formar uma população. Os algoritmos utilizam o conceito de sobrevivência do mais adequado, onde são eliminadas as soluções em que a função de aptidão

[1] O cruzamento é realizando juntando-se as partes trocadas de dois indivíduos, tendo o cuidado para o caso do PCV de não ocorrer repetição de gene (cidade) no indivíduo resultante.
[2] A mutação é realizada trocando-se dois genes em um indivíduo.

[3] A roleta viciada é uma forma de sorteio "tendencioso" que permite que indivíduos com maior aptidão (melhor valor da função objetivo) tenha maior chance de ser sorteado e consequentemente, ser escolhido para sobreviver na próxima geração.

retorna os piores resultados. Cada iteração do método é chamada de “geração”, onde são aplicadas operações de mutação e cruzamento. Essas ocorrem em laço até que um critério de parada seja satisfeito (OLIVEIRA, 2016).

Outro ponto que merece destaque é a adaptabilidade do cromossomo, que é definido pelo custo da solução fornecida por uma heurística, que recebe como um de seus dados de entrada, o vetor de chave do cromossomo e devolve uma solução viável para o problema (SPEARS et al, 1991).

Algoritmos Genéticos com Chaves Aleatórias (RKGA) representam suas soluções como vetores em um determinado intervalo, dessa forma um algoritmo decodificador organiza os vetores e avalia a aptidão de cada solução. A estratégia de decodificação da solução é dependente do problema (TANGPATTANAKUL; JOZEFOWIEZ; LOPEZ, 2013).

CAPÍTULO 4

CONTRIBUIÇÃO CIENTÍFICA DA PESQUISA

A presente pesquisa apresenta-se de grande relevância para o estudo e implementação de redes de alta velocidade, visto que durante toda a pesquisa não foi encontrado estudos referentes a metodologia utilizada na escolha do percurso a ser percorrido pela rede. Outro fator de relevância da pesquisa é apresentar ao leitor a proposta de um modelo com elementos que garantem a sustentabilidade da rede, como o modelo utilizando entidades parceiras que financiarão a rede.

Além disso a pesquisa realiza a abordagem de um problema real com um alto nível de complexidade para a solução, utilizando-se de otimização combinatória para encontrar possíveis soluções para o problema. Dessa forma, gerando conhecimento para problemática abordada.

Outras contribuições coadjuvantes da pesquisa são a adaptação do modelo original do Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP) apresentado por Balas para um modelo de rede gigabytes e a implementação da metaheurística com algoritmo genético para solução do problema, além da confecção de instâncias de teste para o mesmo, gerando assim novos percursos para rede.

Assim o estudo também contribui como uma fonte de pesquisa para futuros trabalhos acerca do tema, visto que foi possível perceber a necessidade de gerar cada vez mais conhecimento sobre o tema, além de trazer uma abordagem diferente ao problema cerne do estudo.

CAPÍTULO 5

DETALHAMENTO DA PROPOSTA E INSTÂNCIAS UTILIZADAS

O objetivo da presente pesquisa é encontrar rotas para rede de alta velocidade Giga-Mossoró, utilizando-se de uma modelagem de otimização com abordagem metaheurística para a solução.

Definido o problema e suas características bem como abordagem para tentativa de resolução, é necessário apresentar os elementos matemáticos e a metaheurística. Apresenta-se inicialmente a formulação matemática proposta, definida inicialmente por Balas (2001), onde tem-se um grafo $G' = (V, A)$ direcionado, nesse caso correspondente a rota a ser percorrida pela rede GigaMossoró. Para cada arco (i, j) de A é dado um custo c_{ij}, e para cada vértice i de N, é associada uma penalidade $\mathbb{\Pi}_i$, a ser paga se o vértice i não compor a rota. Adicionalmente, para cada vértice i, existe um prêmio p_i agregado caso ele seja associado a rede. Os vértices são numerados de 0 até $n = |N|$, sendo que o vértice 0, sem perda de generalidade, determina o ponto inicial da rede, ou seja, onde ficará o provedor.

Determinando que y_i seja 1 se o vértice i for incluído na rota e 0 caso contrário, que x_{ij} seja o vetor de incidência associado à rota (ou seja, assume valor 1 caso a aresta i,j esteja inserido na rota, e 0 caso contrário), e que f garanta que a diferença entre o fluxo que chega ao vértice e que sai do vértice seja igual a 1, se o vértice for visitado, e 0 caso contrário, e também não permite que a quantidade de fluxo entre os vértices i e j ultrapasse o número de vértices possíveis de serem visitados (CHAVES, 2003).

Assim, como proposta para modelagem do problema foi adaptado um modelo matemático do PCVCP para o estudo, como pode ser observado a seguir:

$$\mathbf{Min} \sum_{i \in V} \sum_{j \in V} \boldsymbol{c_{ij} x_{ij}} + \sum_{i \in V} \boldsymbol{\pi_i} \cdot (\mathbf{1} - \boldsymbol{y_i}) \tag{32}$$

s.a.

$$\sum_{j \in V} \boldsymbol{x_{ij}} = \boldsymbol{y_i}, \qquad \boldsymbol{i} = \mathbf{1}, \ldots, \boldsymbol{n} \tag{33}$$

$$\sum_{i \in V} \boldsymbol{x_{ij}} = \boldsymbol{y_i}, \boldsymbol{j} = \mathbf{1}, \ldots, \boldsymbol{n} \tag{34}$$

$$\sum_{i \in V} p_i y_i \geq p_{min} \tag{35}$$

$$\sum_{j \in V} f_{ij} - \sum_{j \in V} f_{ji} = y_i, \quad \forall i \in V \tag{36}$$

$$f_{ij} \leq (n-1) \cdot x_{ij}, \quad \forall (i, j) \in A \tag{37}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n, \; i \neq j \tag{38}$$

$$y_i \in \{0, 1\}, \qquad i = 1, \ldots, n \tag{39}$$

$$f_{ij} \geq 0, \; \forall (i, j) \in A \tag{40}$$

O modelo acima apresentado pode ser descrito da seguinte forma.: a restrição (32) descreve o objetivo do problema, onde deve ser minimizado o custo c_{ij} e as penalidades Π_i, como descrito anteriormente. Para alcançar esse objetivo deve-se atender as restrições, a primeira restrição é a, está (33) determina que se o vértice i estiver na rota, o somatório das arestas que saem dele tem que ser igual a 1, e 0 de outro modo. A restrição (34) diz que se o vértice j estiver na rota, o somatório das arestas que chegam nele tem que ser igual a 1, e 0 caso contrário.

A restrição (35) assegura que o prêmio coletado na rota será maior ou igual ao prêmio mínimo pré-estabelecido anteriormente. No caso da rede GigaMossoró o prêmio será proporcional ao nível de importância que o candidato tem dentro do problema, visto que quanto maior a disponibilidade financeira e o interesse do candidato de se beneficiar da rede, maior será prêmio atrelado ao mesmo.

As restrições (36) e (37) são propostas nesta pesquisa para eliminar sub-rotas. A função (36) garante que o fluxo que chega e que sai do vértice *i* seja igual a 1 caso o vértice seja visitado, e 0 caso contrário; e a restrição (37) assegura que o fluxo máximo que pode passar por uma aresta é o número de vértices menos 1, ou seja, (*n*-1), pois como tenho *n* vértices o número máximo de arestas possíveis em uma solução é *n*-1. As restrições (38) e (39) asseguram a integralidade e a não negatividade das variáveis x_{ij} e y_i, respectivamente, e a função (40) assegura que o fluxo sempre será positivo.

O modelo matemático proposto foi construído para atender as demandas do problema em estudo na pesquisa. Realizado isso, há necessidade de entender a entrada de dados para que o mesmo seja executado. Essas entradas de dados são denominadas instâncias. Para construção dessas por sua vez foi necessário entender e realizar um exame sobre os dados de entrada que são as distâncias entre os clientes em potencial (chamados de pesos ou custos de deslocamento

entre um ponto e outro); Prêmio ou bonificação recebida ao visitar um cliente, assim foi atribuído peso/prêmio a cada nó e penalidades que serão acrescidas dentro do modelo caso um nó não seja visitado.

Para dado de entrada foi realizado um levantamento sobre o comportamento dos dados de entrada e identificado os parâmetros e características dos mesmos, após defini-los iniciou a elaboração e construção das instâncias utilizadas.

Para isso foi realizado um levantamento dos potenciais clientes para a rede construindo uma matriz de distância entre todos os pares. Para fazer parte da lista de clientes, utilizaram-se os seguintes parâmetros: estar em conformidade com os requisitos colocados pelo conselho gestor (instituições de ensino e fundações públicas), empresas públicas e/ou privadas com demanda tecnológica para a rede; estar dentro do perímetro urbano da cidade de Mossoró com aporte financeiro para contratar o serviço. A lista de possíveis clientes e os prioritários pode ser consultada através do Apêndice A.

Com base nos descritores acima já mencionados, foi possível fazer uma relação previa dos clientes em potencial. De posse dessas informações, construiu-se uma matriz de distâncias entre todos os clientes via ferramenta Google Maps para coleta de dados, assim foi construída uma matriz de distâncias entre todos os pares (clientes) com suas distâncias em metros (m), a matriz de distâncias entre todos os pares de clientes é a instancia que alimentará a função objetivo (33) assim como também a instancia de prêmios. A matriz de distância pode ser consultada no apêndice B bem como a matriz de pontos candidatos no apêndice A.

Definido os custos (matriz de distâncias), outra característica a ser observada e definida são os parâmetros de construção dos prêmios. Esses irão definir quão importante é a alocação do cliente na rede, de modo que a cada visita será coletado prêmios, os prêmios coletados são valores monetários que garantirão a manutenção da rede, portanto o modelo terá um prêmio mínimo a ser coletado ao termino do percurso.

Inicialmente foram estabelecidos os prêmios para os nós prioritários, que receberão valor 0 (zero) visto que não haveria perda de generalidade. Uma vez que são atribuídos prêmios para os mesmos, poderá haver uma distorção dos resultados. O que garantirá a passagem nos nós prioritários será a penalidade. Para os demais clientes da rede, foi estabelecido como escopo os valores entre 1 e 100, de modo que quanto maior o porte do cliente, ou seja, quanto mais requisitos a organização dispor (capacidade financeira, tecnologia e atributos determinados pelo conselho gestor) maior é sua nota, caso contrário será menor.

Porém foi observado que há necessidade de estabelecer outro critério em relação ao prêmio atribuído aos possíveis clientes, um fator a ser analisado é sua distância em relação ao arco da rede, onde quanto maior a distância do cliente menor será sua ponderação e uma pontuação para o porte da organização, para definir o porte foi analisado a sua capacidade financeira de pagar pelo serviço, demanda de internet e criticidade, que representa qual o impacto da disponibilidade da internet para o cliente.

Para o primeiro caso foi definido que como é difícil especificar a distância do cliente para o arco da rede, então foi estabelecido como parâmetro a distância entre todos os clientes candidatos e o ponto inicial da rede, que é a UERN. Definidas essas distâncias, foi possível entender que a ponderação nesse caso terá um fator inverso, visto que quando mais distante do ponto inicial, menor será seu peso (menor importância). Dessa forma calculou-se o peso inverso. Para efeitos de cálculo utilizou-se da equação (41) a seguir:

$$Pd_j = \frac{1000}{d_j} \tag{41}$$

A equação (41) pode ser lida da seguinte forma: Pd é o peso da distância inversa do cliente em relação a UERN. Nesse caso utilizou-se 1000 como base, visto que as distâncias estão em metros (m) e d_{ij} que corresponde a distância entre o nó inicial da rede e o cliente.

O outro parâmetro corresponde a ponderação é o porte da organização. Para isso foi levado em consideração a questão financeira, demanda e criticidade. Assim foi atribuído notas entre 0 (zero) e 100, para os clientes. Depois de definido todos os parâmetros, foi possível encontrar o prêmio correspondente a cada cliente utilizando-se a equação (42), para efeito de construção da instancia os valores finais forma multiplicados por 100 para serem convertidos em números inteiros, como pode ser observado a seguir:

$$Prêmio\ final_j = Pd_j * Porte_j * 100 \tag{42}$$

Então o prêmio final (42) será a relação entre o peso da distância versus porte do cliente. Assim foram obtidos os resultados apresentados na figura 6:

Cálculo do prêmio ao visitar um cliente				
Candidatos	**Distância em relação a UERN**	**Peso Distância**	**Porte (Financeiro / Demanda / Criticidade)**	**Prêmio final**
UERN (Campus central)	0	0	0	0
UFERSA	0	0	0	0
IFRN	0	0	0	0
UERN (REITORIA)	0	0	0	0
UERN (FACS)	0	0	0	0
FAC. MATHER CRISTH	4.400	0,23	70	1589
FAC. DIOCESANA	4.500	0,22	70	1554
UNP	7.900	0,13	90	1143
UNIT	3.400	0,29	65	1911
Unopar	4.000	0,25	65	1625
Hosp. Wilson Rosado	4.200	0,24	88	2094
Rebouças Albeto M.	3.600	0,28	60	1668
Rebouças Av. Pres. Dutra	2.700	0,37	60	2220
Thermas	6.900	0,15	50	725
Garbos	6.300	0,16	50	795
Villa Oeste	3.400	0,29	50	1470
Partage Shopping	11.300	0,09	88	774
BB Alberto Maranhão	4.300	0,23	75	1748
BB Centro	3.700	0,27	75	2025
BB Ilha de SL	3.400	0,29	75	2205
BB Stilo	3.900	0,26	75	1920
Caixa Economica Centro	4.000	0,25	75	1875
Caixa Economica Av. Pres. Dutra	3.200	0,31	75	2348
Bradesco	3.900	0,26	75	1920
Santander	3.700	0,27	75	2025
Policia Civil 2ª DP	4.100	0,24	70	1680
Central do Cidadão/DETRAN-RN	8.800	0,11	85	935
Ministério Público Federal	1.800	0,56	80	4480
Ministério Publico do RN	3.300	0,30	80	2400
Forúm Desembargador Silveira Martins	2.900	0,34	70	2380
Fórum Trabalhista de Mossoró, 21ª Região	2.800	0,36	70	2520
Aeroporto	6.600	0,15	85	1275
Secretaria Municipal da Fazenda de Mossoró	5.200	0,19	75	1425
Secretaria da Receita Federal do Brasil	4.200	0,24	75	1800
Prefeitura de Mossoró	4.200	0,24	70	1680
Centro Administrativo da Prefeitura de Mossoró	6.400	0,16	80	1280
Intertv Costa Branca	1.900	0,53	85	4505
Policia Federal	5.800	0,17	80	1360
Atacadão	7.900	0,13	70	910
Maxxi Atacado	7.100	0,14	70	980

Figura 4 - Prêmio por cliente
Fonte: Autores 2018.

A matriz de prêmio definida será a instância que será utilizada pela equação (35) do modelo matemático, assim a cada ponto visitado pela rede receberá uma bonificação. Definido o prêmio, temos por último o parâmetro penalidade, este por sua vez tem a característica de atribuir punições (penalidades) caso um determinado nó não seja visitado, isto é, não seja inserido na rota. Então foram criados parâmetros para determinar qual a penalidade em cada nó. Inicialmente foi pensado como garantir que os clientes prioritários iriam compor a rota. Para isso foi atribuído um valor elevado para esses clientes como pode ser observado na figura 7, de modo a garantir a inserção desses dentro do percurso da rede, uma vez que o objetivo do modelo é minimização de custos e penalidades.

Para os demais nós da rede utilizou-se uma escala de 1 a 100 (pesos). Neste caso a ponderação de penalidade seguirá a mesma lógica dos valores altos para os pontos com maior capacidade técnica como já mencionado acima. Assim, quanto maior o potencial do cliente para ser inserido na rede, maior também será o peso da penalidade, visto que cada ponto não inserido na rede elevará o valor final da função objetivo, assim quantos mais pontos for inserido, mais a capacidade de minimização da função e maior o prêmio coletado.

Tomando como base os descritores, foi construída a seguinte matriz de penalidades, de modo que cada pontos apresente uma penalidade, como pode ser observada na figura 7:

Penalidades	
UERN (Campus central)	1000000
UFERSA	1000000
IFRN	1000000
UERN (REITORIA)	1000000
UERN (FACS)	1000000
FAC. MATHER CRISTH	90
FAC. DIOCESANA	90
UNP	90
UNIT	90
Unopar	90
Hosp. Wilson Rosado	80
Rebouças Albeto M.	70
Rebouças Av. Pres. Dutra	70
Thermas	60
Garbos	60
Villa Oeste	60
Partage Shopping	70
BB Alberto Maranhão	75
BB Centro	75
BB Ilha de SL	75
BB Stilo	75
Caixa Economica Centro	75
Caixa Economica Av. Pres. Dutra	75
Bradesco	75
Santander	75
Policia Civil 2ª DP	70
Central do Cidadão/DETRAN-RN	70
Ministério Público Federal	65
Ministério Publico do RN	65
Forúm Desembargador Silveira Martins	65
Fórum Trabalhista de Mossoró, 21ª Região	65
Aeroporto	80
Secretaria Municipal da Fazenda de Mossoró	75
Secretaria da Receita Federal do Brasil	75
Prefeitura de Mossoró	75
Centro Administrativo da Prefeitura de Mossoró	75
Intertv Costa Branca	80
Policia Federal	80
Atacadão	65
Maxxi Atacado	65

Figura 5 - Penalidade atribuídas a cada cliente
Fonte: Autores 2018.

As penalidades informadas na matriz será a instancia que alimentará o componente $\sum_{i \in V} \boldsymbol{\pi_i} \cdot (\mathbf{1} - \boldsymbol{y_i})$ da função objetivo. Essas informações geradas, estabeleceu o conjunto de instâncias utilizadas para execução do algoritmo desenvolvido.

Definido a matriz de distância, os descritos de prêmios e penalidades, se fez necessário estabelecer o prêmio mínimo para a ser coletado durante o percurso da rede, assim ficou estabelecido que os prêmios deveriam ser 75% e 90% do prêmio total da rede. Uma vez que quanto maior o percentual do prêmio maior o número de pontos inseridos no percurso.

CAPÍTULO 6
VALIDAÇÃO E RESULTADOS OBTIDOS

6.1 Validação do Modelo e Testes Computacionais

O presente capítulo apresenta informações relevantes com relação a validação do Problema do Caixeiro Viajante com Coleta de Prêmios, apesar do PCVCP apresentar diversas aplicações no mundo real, pouco ainda é explorado no sentido de criar bibliotecas públicas de teste. Dessa forma para avaliar a heurística proposta na presente pesquisa, forma utilizadas instâncias obtidas da seguinte forma: inicialmente foi validado o problema com instâncias que estão disponíveis no seguinte endereço: http://www.decom.ufop.br/prof/marcone/Orientacoes/Orientacoes-Concluidas.htm, por conseguinte, foi utilizado as instâncias confeccionadas durante a pesquisa conforme foi demonstrado no capítulo anterior.

O modelo matemático proposto é classificado como um modelo de Programação Linear, uma vez que, tanto a função objetivo como as restrições são inequações lineares. Como as variáveis do modelo só admitem valores inteiros então este é um modelo de Programação Linear Inteira (exata), ou simplesmente de Programação Inteira (PRADO, 1999).

Para solução/validar o modelo matemático exato aqui proposto, utilizou-se o software da IBM ILOG CPLEX ® *Optimization Studio* versão V12.6.3, versão *student*. Para execução deste foi utilizado uma máquina com processador Intel(R) Core(TM) i5-3337U CPU 1.80GHz, com 4 GB de memória RAM, com Sistema Operacional Windows 64 bits - processador x64.

Para validar o modelo matemático foi elaborado um algoritmo dentro do CPLEX ® *Optimization Studio* como pode ser consultado no ANEXO E. As instâncias inicialmente utilizadas dentro do modelo exato foram as do trabalho de Chaves (2003), visto que esse apresentou resultados consolidados para as mesmas. As instâncias podem ser consultadas através do link http://www.decom.ufop.br/prof/marcone/Orientacoes/OrientacoesConcluidas.htm, para efeitos de comparação utilizou-se inicialmente as PCV_CP10 (11 pontos) e PCV_CP20 (21 pontos) com ótimos 1765 e 2302 respectivamente, utilizando de 75% do prêmio total como parâmetro, os resultados obtidos são apresentados na Tabela 1.

EXECUÇÃO COM PRÊMIO MÍNIMO: 75% DO SOMATÓRIO DOS PRÊMIOS						
Instância	Função objetivo	Tempo (s)	Prêmio mínimo	Coletado	Penalidade	Não visitados
PCV_CP11	1765.00	1.02	287.25	332.00	10	2

PCV_CP21	2302.00	2.67	756.00	920.00	101.00	2

Tabela 1 - Resultado das instâncias do trabalho de Chaves (2003) para 75% do prêmio total, modelo exato
Fonte: Autor 2018.

Observado os valores obtidos e comparados com os encontrados no trabalho de Chaves (2003) constatou-se que são os mesmos, dessa forma valida-se o modelo proposto. Como parâmetro de análise resolveu-se aumentar o percentual do prêmio de 75% para 90% do total. Os resultados podem ser observados na Tabela 2.

EXECUÇÃO COM PRÊMIO MÍNIMO: 90% DO SOMATÓRIO DOS PRÊMIOS						
Instância	Função objetivo	Tempo (s)	Prêmio mínimo	Coletado	Penalidade	Não visitados
PCV_CP11	2244.00	1.03	344.70	363.00	8	1
PCV_CP21	2302.00	3.38	907.20	920.00	101.00	2

Tabela 2 - Resultado das instâncias do trabalho de Chaves (2003) para 90% do prêmio total, modelo exato
Fonte: Autor 2018.

Comparando os resultados apresentados nas tabelas 1 e 2, pode-se verificar que leves alterações, uma vez que a mais significativa foi em relação a instância PCV_CP11 para 90% do prêmio que não visitou apenas 1 ponto. Os resultados quando comparados com os resultados da pesquisa do Chaves (2003), apresentam as mesmas características.

Com o modelo validado foi necessário testar as instâncias construídas com os dados da pesquisa. Essas foram distribuídas da seguinte forma: PCV_CP5, para os 5 pontos prioritários da rede (não podem ser visitados); o PCV_CP10, com 10 pontos da rede; o PCV_CP20, com 20 pontos da rede e o PCV_CP27, com 27 pontos da rede. Não foi possível inserir o PCV_CP40, com todos os pontos da rede, visto que o CPLEX ® *Optimization Studio*, na sua versão *student* não pode utilizar mais que 27 pontos. Inicialmente utilizou-se 75% do prêmio total e os resultados podem ser observador na Tabela 3.

EXECUÇÃO COM PRÊMIO MÍNIMO: 75% DO SOMATÓRIO DOS PRÊMIOS						
Instância	Função objetivo	Tempo (s)	Prêmio mínimo	Coletado	Penalidade	Não visitados
PCV_CP5	18400.00	0.63	0	0	0	0
PCV_CP10	19670.00	1.16	5866.50	6679.00	90.00	1
PCV_CP20	20091.00	93.08	17659.50	18441.00	350.00	5
PCV_CP27	20611.00	5882.74	27186.75	28529.00	490.00	7

Tabela 3 - Resultado das instâncias do trabalho para 75% do prêmio total, modelo exato
Fonte: Autor 2018.

Para efeitos de comparação foi também obtido resultados com prêmio mínimo de 90% do prêmio total dos vértices. Como pode ser observado na Tabela 4.

EXECUÇÃO COM PRÊMIO MÍNIMO: 90% DO SOMATÓRIO DOS PRÊMIOS						
Instância	Função objetivo	Tempo (s)	Prêmio mínimo	Coletado	Penalidade	Não visitados
PCV_CP5	18400.00	0.95	0	0	0	0
PCV_CP10	22880.00	1.59	7039.80	7822.00	0	0
PCV_CP20	23981.00	83.32	21191.40	21252.00	190.00	3
PCV_CP27	23351.00	4077.17	32624.10	32812.00	280.00	4

Tabela 4 - Resultado das instâncias do trabalho para 75% do prêmio total, modelo exato
Fonte: Autor 2018.

Comparando os resultados obtidos para as instâncias com 75% e 90%, pode-se observar que os resultados da função objetivo para 90% melhoram em relação a 75%, e os pontos não visitados para 90% são menores que 75%, isso se dá em virtude da elevação do prêmio mínimo estabelecido. O fator que merece destaque para ambos os casos é o tempo, tanto para 75% quanto para 90% da instância PCV_CP27, os tempos são de aproximadamente 1h e 38min e 1h e 8min respectivamente, esses valores só demonstram que quanto maior o nível de complexidade e de vértices, há a necessidade de elaborar uma metaheurística para solução do problema, uma vez que quanto maior a complexidade da rede, maior será o número de vértices (clientes) candidatos, elevando assim o tempo computacional, uma vez que o mesmo cresceu de forma exponencial e tornando-se ineficiente para tomada de decisão.

Diante dos resultados apresentados e das dificuldades apresentadas, utilizou-se como proposta de metaheuristica o algoritmo genético para encontrar soluções ótimas para o problema em análise. Para desenvolvimento do algoritmo foi escolhido a linguagem de programação Python, está se deu em virtude de ser uma linguagem de alto nível, orientada a objeto e funcional e para o desenvolvimento do algoritmo utilizou-se o software PyCharm versão 2018.2.2 e compilador Python 3.7 64 bits, instalado em uma máquina com processador Intel® Core™ i5-7200u, com 2.5GHz e com 4 Giga de memória RAM, com Sistema Operacional Windows 64 bits, que foi a mesma utilizada para executar as instâncias.

O algoritmo genético desenvolvido para presente pesquisa pode ser consultado no APENDICE F, o mesmo está dividido em 4 conjuntos de instruções, são eles: Tools, Cidadecandidata, rota e main. Tools faz toda a parte de parametrização do genético, gerando uma população inicial, realizando os cruzamentos e mutações entre os vetores, por fim ordenando e selecionando os sobreviventes, gerando assim um conjunto cada vez melhor a cada nova interação. Na

cidade candidata e rota, é realizado o cálculo do custo das cidades visitas e define o percurso a ser seguido, levando em consideração os prêmios e penalidades respectivamente, por fim apresenta-se o main, nesse é definido os parâmetros como o nome das instâncias a serem lidas e executadas, uma vez que são instâncias de tamanhos diferentes com características diferentes para serem executadas, define-se o tamanho da população inicial, percentual de corte e o prêmio mínimo a ser coletado durante o percurso da rota.

Definido e desenvolvido o algoritmo, houve a necessidade de realizar experimentos para as instâncias, assim como foi realizado para o modelo exato. Dessa forma, utilizou-se o mesmo parâmetro para o prêmio mínimo a ser coletado, que são 75% e 90% do prêmio total, as instâncias são as mesmas, com uma leve mudança, uma vez que devido restrições do CPLEX ® não foi possível encontrar resultados para instâncias com 40 (quarenta) vértices, já com a metaheuristica foi possível.

Também ficou definido que a geração da população inicial para todas as instâncias seria de 100 (cem), o percentual de corte de 50% da população a cada nova interação. Para todos os resultados obtidos foram realizadas 10 execuções do algoritmo de modo que para cada parâmetro foi levado em consideração sua média, com excessão de do número de pontos (clientes) não visitados, para esse utilizou-se a moda.

Inicialmente obteve-se os resultados das instâncias utilizadas por Chaves (2003), como forma de comparação e observar o comportamento do algoritmo, assim foram gerados resultados para 75% como pode ser observado na Tabela 5.

EXECUÇÃO COM PRÊMIO MÍNIMO: 75% DO SOMATÓRIO DOS PRÊMIOS						
Instância	Função objetivo	Tempo (s)	Prêmio mínimo	Coletado	Penalidade	Não visitados
PCV_CP11	1989.00	8.95	287.25	310.3	12,9	2
PCV_CP21	3853.00	16.51	756.00	363.00	670.3	4

Tabela 5 - Resultado das instâncias do trabalho de Chaves (2003) para 75% do prêmio total utilizando Metaheuristica
Fonte: Autor 2018.

Da mesma forma, foram gerados resultados para as mesmas instâncias com 90% como pode ser observado na Tabela 6.

EXECUÇÃO COM PRÊMIO MÍNIMO: 90% DO SOMATÓRIO DOS PRÊMIOS						
Instância	Função objetivo	Tempo (s)	Prêmio mínimo	Coletado	Penalidade	Não visitados
PCV_CP11	2244.00	8.90	344.7	363.00	8	1
PCV_CP21	3710.00	17.9	907.2	947.1	155.9	2

Tabela 6 - Resultado das instâncias do trabalho de Chaves (2003) para 90% do prêmio total utilizando Metaheuristica

Fonte: Autor 2018.

Com a mesma parametrização foram obtidos os resultados para as instâncias do trabalho como pode ser observado na Tabela 7 e Tabela 8. Foram utilizados para esses resultados os dados descritos anteriormente.

EXECUÇÃO COM PRÊMIO MÍNIMO: 75% DO SOMATÓRIO DOS PRÊMIOS						
Instância	Função objetivo	Tempo (s)	Prêmio mínimo	Coletado	Penalidade	Não visitados
PCV_CP10	23432.00	8.47	5866.5	6093.00	90	1
PCV_CP20	25921.00	14.68	17659.5	25567.5	353	5
PCV_CP27	30889.20	28.72	27186.75	28987.7	499	7
PCV_CP40	43609.40	39.22	47433.00	48868.6	666	12

Tabela 7 - Resultado das instâncias do trabalho para 75% do prêmio total utilizando Metaheuristica.
Fonte: Autor 2018.

EXECUÇÃO COM PRÊMIO MÍNIMO: 90% DO SOMATÓRIO DOS PRÊMIOS						
Instância	Função objetivo	Tempo (s)	Prêmio mínimo	Coletado	Penalidade	Não visitados
PCV_CP10	24080.00	9.3	7039.80	7822.00	0	0
PCV_CP20	30865.10	15.76	21191.40	21682.80	142	2
PCV_CP27	37113.70	26.22	32624.10	33175.50	228	3
PCV_CP40	47433.00	42.83	56919.60	57760.70	242	4

Tabela 8 - Resultado das instâncias do trabalho para 75% do prêmio total utilizando Metaheuristica.
Fonte: Autores 2018.

Diante dos resultados obtidos foi possível validar o algoritmo genetico visto que os resultados gerados atenderam os requisitos de prêmios mínimos e foi inserido todos os pontos prioritários dentro do percurso.

Com os resultados das instâncias geradas pela metaheuristica da presente pesquisa, surgiu à necessidade de realizar uma analise comparativa entre os resultados obtidos no modelo exato e com a metaheuristica com a finalidade de observar melhorias ou não diante dos resultados encontrados.

6.2 Resultados Obtidos

Neste capítulo, apresentam-se os resultados computacionais obtidos com testes para as instâncias geradas no estudo, como mencionado anteriormente, e um comparativo entre os resultados encontrados nas modelagens exatas e a metaheuristica. A apresentação dos experimentos computacionais através do comparativo tem como finalidade verificar a qualidade do algorítmo desenvolvido para solução.

Para realizar o comparativo, foram levados em consideração as instâncias PCV_CP11 e PCV_CP21 do trabalho do chaves para 75%, bem como as instâncias PCV_CP10, PCV_CP20 e PCV_CP27, com os respectivos percentuais. Não foram inseridas as instâncias com 5 e 40 vértices, uma vez que estão apenas os pontos prioritários e esses não influenciaram na manutenção da rede e o fato do CPLEX não apresentar licença para rodar uma instância com 40 vértices, respectivamente. Assim ordenou os dados comparando todos os elementos, como pode ser observado na Tabela 9. Para efeitos de comparação foram levados em consideração a função objetivo, o tempo, prêmio coletado e penalidades.

Para realizar a análise de elevação percentual como comparativo entre os resultados encontrados, utilizou-se os valores da metaheriticas como referência.

EXECUÇÃO COM PRÊMIO MÍNIMO: 75% DO SOMATÓRIO DOS PRÊMIOS				
Instância		Exato	Metaheuristica	Elevação
PCV_CP10	Função objetivo	19670	23432	16,05%
	Tempo (s)	1.16	8.47	86,30%
	Prêmio Coletado	6679	6093	- 9,62%
	Penalidade	90	90	0,00%
PCV_CP11	Função objetivo	1765	1989	11,26%
	Tempo (s)	1.02	8.95	88,60%
	Prêmio Coletado	332	310	- 7,10%
	Penalidade	10	8	- 25,00%
PCV_CP20	Função objetivo	20091	25921	22,49%
	Tempo (s)	93.08	14.68	- 534,06%
	Prêmio Coletado	18441	25567	27,87%
	Penalidade	350	353	0,85%
PCV_CP21	Função objetivo	2302	3853	40,25%
	Tempo (s)	2.67	16.51	83,83%
	Prêmio Coletado	920	363	- 153,44%
	Penalidade	101	670	84,93%
PCV_CP27	Função objetivo	20611	30889	33,27%
	Tempo (s)	5882.74	28.72	- 20383,08%
	Prêmio Coletado	28529	28987	1,58%
	Penalidade	490	499	1,80%

Tabela 9 - Analise comparativa dos resultados das instâncias para 75% do prêmio total

Fonte: Autor 2018.

Observando os dados descritos na Tabela 9 pode-se observar que na função objetivo e no prêmio coleta, houveram elevações dos resultados da metaheuristica em relação ao método exato, isso se dá em virtude dos parâmetros de ajustes como corte e mutação que ainda não foram bem refinados, e devido ao pouco tempo para gerar um número maior de resultados e realizar analises comparativas.

Outro parâmetro que merece destaque na análise é o tempo de execução, quando comparado os resultados o ganho de tempo com a metaheuristica é de mais de 2.000%, isso significa que o problema poderá elevar o número de vértices, por conseguinte a complexidade e mesmo assim a solução será apresentada dentro de um tempo computacional baixo, justificando-se a utilização de metaheuristicas para solução desse tipo de problema.

CAPÍTULO 7

7.1 CONSIDERAÇÕES FINAIS

A definição do percurso ideal para uma rede de alta velocidade é um processo de extrema importância do ponto de vista de sua performance, estabilidade e manutenção (sobrevivência) da mesma. Definir o percurso nem sempre é uma tarefa simples e de fácil compreensão, uma vez que a sua instalação deve atender requisitos predeterminados pelos atores envolvidos e requisitos ótimos de configuração, de modo abranger o maior número de clientes, mas não é qualquer cliente, são aqueles que querem, podem e vão usufruir da rede.

Assim tal problema mostra-se como um problema complexo de abordagem prático, para isso foi necessário conhecer os parâmetros e necessidades da rede, de modo a garantir que todos os elementos estejam dentro do conhecimento aqui gerado. Antes de saber quais dados irão gerar informações, é essencial conhecer quais variáveis são pertinentes para a problemática do estudo.

O presente estudo contemplou essa análise, pois foi conhecido quais os elementos que afetam a inserção ou não de um cliente a rede. De posse dessas informações foi estabelecido um modelo matemático para solução do problema. No estudo optou-se por utilizar um modelo do PCVCP, uma vez que as características dos dados da pesquisa se assemelham com modelo.

Partindo desse pressuposto, definiu-se um modelo matemático para a problemática aqui abordada. Esse modelo foi validado uma vez que para se gerar resultados para o estudo, há, portanto, a necessidade de entender o nível de exatidão do mesmo. Com o modelo validado, foi desenvolvido uma metaheuritica com base no mesmo para propor soluções para o problema aqui abordado, gerando resultados satisfatórios.

Os resultados obtidos com o modelo e a metaheuristica se mostram satisfatório, visto que os mesmos foram gerados com base em uma analise detalhada dos elementos que fazem a rede, além de estabelecer métricas e atender as restrições para rede.

Além disso, a pesquisa trouxe benefícios para quem trabalha definições de caminhos de rede de alta velocidade, pois formalizou um pensamento tácito em um modelo matemático passível de aplicação para tomada de decisão, servindo de base para trabalhos futuros.

7.2 TRABALHOS FUTUROS

Recomenda-se que para aprimorar o modelo aqui elaborado, seja realizado uma análise mais profunda dos parâmetros de refinamento do algoritmo geneéico, onde deve-se agregar mais conhecimento de seu comportamento e influência sobre o resultado. Para isso é interessante realizar mais interações e analisar os resultados obtidos de modo a refiná-los cada vez mais.

Outro ponto que merece atenção é a possibilidade de aumentar o número de vértices em análise no estudo, uma vez que quanto maior esse número, maior a probabilidade de caminhos para a rede e, por conseguinte os resultados obtidos serem ótimos.

Como ponto de destaque é a elaboração de outras metaheuristicas como o GRASP e BRKGA, para análise e comparação. Uma vez que quanto maior a solides dos resultados, melhor a tomada de decisão.

REFERÊNCIAS

ARAKAKI, R. G. I. Heurística de localização-alocação para problemas de localização de facilidades. Tese de Doutorado em Computação Aplicada, INPE- Instituto Nacional de Pesquisas Espaciais, São José dos Campos, São Paulo, Brasil, 2002.

ARAÚJO, M. D. Uso da metaheurística GRASP na solução do problema de localização de facilidades: uma aplicação no setor de segurança pública. Trabalho de Conclusão de Curso, Ciência da Computação UERN, p. 53, 2013.

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. Pesquisa Operacional. Ed. Campus, 2006.

BALACHANDRAN, V. *An integer generalized transportation model for optimal job assignment in computer networks.* Operations Research, v. 24, p. 742–759. 1976.

BALAS, E. *The Prize Collecting Traveling Salesman Problem and Its Applications.* Management Science Research Report, MSRR-664, 2001.

BALLOU, R. H. Gerenciamento da cadeia de suprimentos: logística empresarial. 5ªed. Porto Alegre: Bookman, 2006.

BANSAL, N.; BUCHBINDER, N.; MADRY, A.; NAOR, J. A. *Polylogarithmic-Competitive Algorithm for the k-Server Problem.* Foundations of Computer Science (FOCS), 52nd Annual Symposium on IEEE, 2011.

BARTAL, Y.; KOUTSOUPIAS, E., *On the Competitive Ratio of the Work Function Algorithm for the k-Server Problem*. Theor. Comput. Sci. v. 324, p. 337 – 345, 2004.

BASSANEZI, R. C. Ensino – aprendizagem com modelagem matemática. São Paulo: Editora Contexto, 2002.

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. ORSA journal on computing, n.2, p.154-160, 1994.

BORODIN, A.; EL-YANIV, R. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

BORODIN, A.; LINIAL, N.; SAKS, M. *An optimal online algorithm for metrical task systems*. Journal of the ACM, v. 39, p. 745 – 763, 1992.

BRADLEY, R. E.; SANDIFER, C. E. *Leonhard Euler: Life, work and legacy*. Oxford: Elsevier, 2007.

BRANDEAU, M. L.; CHIU, S. S. *An overview of representative problems in facility location*. Management Science. The Institute of Management Science. v.35, n.6, 1989.

BURKARD, R. E. *Selected topics on assignment problems.* Discrete Applied Mathematics, v. 123, n. 3, p. 257-302, 2002.

C.A.S.; PARDALOS, P.M. *On the average case behavior of the multidimensional assignment problem.* Pacific Journal of Optimization, v. 1, n. 1, p. 39-57, 2005.

CAMPELLO, R. E.; MACULAN, N. Algoritmos e Heurísticas: desenvolvimento e avaliação de performance. Editora da Universidade Federal Fluminense – EDUFF, Niterói, 1994.

CAMPOS, V. B. G. Algoritmos para Resolução de Problemas em Redes. INSTITUTO MILITAR DE ENGENHARIA - IME, Rio de Janeiro - RJ, 1997.

CATTRYSSE, D. G.; SALOMOM, M.; WASSENHOVE, L. N. V. *A set partitioning heuristic for the generalized assignment problem.* European Journal of Operational Research, v. 72, p.167 – 174. 1994.

CHAVES, A. A.; et al. *Modelagens Exata e Heurística para Resolução do Problema do Caixeiro Viajante com Coleta de Prêmios.* Relatório Técnico – Universidade Federal de Ouro Preto, Ouro Preto, 2003. Disponível em < http://www.decom.ufop.br/prof/marcone/Orientacoes/OrientacoesConcluidas.htm > Acessado em 15 de fev de 2018.

CHAVES, A. A.; et al. *Metaheurísticas híbridas para resolução do problema do caixeiro viajante com coleta de prêmios*. Produção, v. 17, p. 263 – 272, 2007.

CHRISTOFIDES, N. *Graph Theory: Na Algorithimic Approach*. New York: Academic Press, 1975.

CHU, P. C.; BEASLEY, J. E. *A genetic algorithm for the generalized assignment problem*. Computers and Operations Research, v. 24, p. 17 – 23. 1997.

CORDENONSI, A. Z. Ambientes, objetos e dialogicidade: uma Estratégia de Ensino Superior em Heurísticas e Metaheurísticas. Tese de Doutorado: Programa de Pós-Graduação em Informática na Educação – UFRGS, 228 f., 2008.

DAMBRÓSIO, U. Dos fatos reais à modelagem matemática: uma proposta de conhecimento matemático. Disponível em: <http://vello.site.uol.com.br/modelos.htm.> Acesso em: 02 jan. 2018.

DANTZIG, G.; FULKERSON, R.; JOHNSON, S. *Solution of a Large-Scale Traveling Salesman Problem*. Journal of the operation Res. Soc. Am., v. 2, p. 393-410, 1954.

DASKIN, M. *Network and Discrete Location: Models, Algorithms, and Applications.* Wiley Interscience, New York, 1995.

DASKIN, M. *What You Should Know About Location Modeling*. In: Naval Research Logistics. New York: Wiley Interscience, v.55, p.283-294. 2008.

DAWANDE, M.; KALAGNANAM, J. *The multiple knapsack problem with color constraints*. IBM T. J. Watson Research, Tech. Rep. 1998.

DELL'AMICO, M.; MAFFIOLI, F. & SCIOMANCHEN, A.A Lagrangian Heuristicfor the Prize Collecting Travelling Salesman Problem." Annals of Operations Research, 81, p.289–305. 1998.

DEB, K. Multi-Objective Optimization Using Evolutionary Algorithms. New York: Wiley. p. 518, 2001.

DIAZ, J. A., FERNANDEZ, E. *A Tabu search heuristic for the generalized assignment problem.* European Journal of Operational Research, v. 132, p. 22–38. 2001.

DOWSLAND, K. A. Genetic algorithms, a tool for OR? Journal of the Operacional Research Society, 47, 550-561, 1996.

FARAHANI, R. Z.; STEADIESEIFI, M.; ASGARI, N. *Multiple criteria facility location problems: a survey*. Applied Mathematical Modelling, v. 34, p. 1689 – 1709, 2010.

FIREMAN, E. C.; SANTOS, J. R. G. dos. O ENSINO DE MATEMÁTICA E DE FÍSICA INTEGRADO PELA MODELAGEM COMPUTACIONAL. UFAL - Alagoa, 2014.

FISCHETTI, M.; LEPSCHY, C.; MINERVA, G.; ROMANIN-JACUR, G.; TOTO, E. *Frequency assignment in mobile radio systems using branch-and-cut techniques.* European Journal of Operational Research, v. 123, p. 241-255, 2000.

FISHER, M. L.; JAIKUMAR, R. *A generalized assignment heuristic for vehicle routing.* Networks, v. 11, p. 109 – 124. 2006.

FORD, L. R.; FULKERSON, D. R. *Flows in Networks.* Princeton Univ., New Jersey. 1962.

FRANCIS, R. L.; WHITE, J. A. *Facility Layout and Location: An Analytical Approach.* Prentice-Hall, Englewood Cliffs, New Jersey. 1974.

GARCIA, S. F. A.; LIMA, G. B.; CARVALHO, D. T. Redes Interorganizacionais de Cooperação para a Internacionalização. REGE, São Paulo, v. 17, n. 2, p. 209-224, 2010.

GOEMANS, M.; WILLIANSON, D. A General Approximation Tchnique for Constrained Forest Problems, In Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms, p.307-315, 1992.

GOLDBARG, M. C.; LUNA, H. P. Otimização Combinatória e Programação Linear - Modelos e Algoritmos. Editora Campus, 2000.

GOLDBERG, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, 1989.

GONÇALVES J.F., RESENDE M.G.C. Biased random-key genetic algorithms for combinatorial optimization. Journal of Heuristics, 17, p.487-525, 2011.

GOULART, N. et al. Biased random-key genetic algorithm for fiber installation *in optical network optimization.* In: 2011 IEEE Congress of Evolutionary Computation (CEC). p. 2267–2271. 2011.

GRUNDEL, D.A.; KROKHMAL, P.; OLIVEIRA, *for the generalized assignment problem.* European Journal of Operational Research, v. 169, p. 548–569. 2006.

GUIGNARD, M.; ROSENWEIN, M. *An improved dual-based algorithm for the generalized assignment problem.* Operations Research, v. 37, p. 658 – 663. 1989.

HAKIMI, S. L. *Optimum location of switching centers and the absolute centers and the medians.* Operations Research, v. 12, p. 450 – 459, 1964.

HILLIER, F. S.; LIEBERMAN, G. J. *Introduction to Operations Research*. 8° editon,Nova York, Mac-Graw Hill, 2004

HILLIER, F. S.; LIEBERMAN, G. J. *Introduction to Operations Research.* New York (NY): McGraw Hill, Seventh Edition. p. 805, 2002.

HOLLAND, J. H. Adaptation in Natural and Artificial Systems. Cambridge, MA, ISBN 0-262-58111-6, USA: MIT Press, 1992.

HOLLAND, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, 1975.

HUNG, M. S.; FISK, J. C. *A heuristic routine for solving large loading problems*. Naval Research Logistic Quarterly, vol. 26, 643–650. 1979.

HUNG, M. S.; ROM, W. O. *Solving the assignment problem by relaxation.* Operations Research, v. 28, n. 4, p. 969-982, 1980.

JANSON B. N. *Dynamic Traffic Assignment for Urban Road Networks*. Trans. Research, v. 25b, n. 2/3. 1991.

KARIV, O.; HAKIMI, S. L. *An algorithmic approach to network location problems,* Part I: The p-centers. In: SIAM Journal of Applied Mathematics. v.37, p.513–538. 1979.

KARP, R. M. *On the Computational Complexity of Combinatorial Problems*. Networks 5, 45-68, 1975.

KIRUBARAJAN, T.; BAR-SHALOM, Y.; PATTIPATI, K. R. *Multiassignment for tracking a large number of overlapping objects.* IEEE Transactions on Aerospace and Electronic Systems, v. 37, n. 1, p. 2-21, 2001.

KLOSE, A.; DREXL, A. *Facility location models for distribution system design*. European Journal of Operational Research,. v.162, p. 4 - 25, 2005.

KOOPMANS, T.C.; BECKMANN, M.J. Assignment problems and the location of economic activities. Econometrica, v. 25, p. 53-76, 1957.

KRASNOGOR, N. *Studies on the theory and design space of memetic algorithms.* Faculty of computing, engineering and mathematical sciences. University of the West of England, UK. PhD thesis, 2002.

KUHN, H. W. *The Hungarian method for the assignment problem.* Naval Research Logistic Quaterly, v. 2, p. 83-97, 1955.

KUMAR, R. Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms. International Journal of Machine Learning and Computing, 2012.

LAGUNA, M.; KELLY, J. P.; GONZALEZ-VELARDE, J. I.; Glover, F. Tabu search for the multilevel generalized assignment problem. European Journal of Operations Research, v. 42, p. 677–687. 1995.

LAPORTE, G.; MARTELLO, S. *The Selective Traveling Salesman Problem*. Discrete Appl. Math, v. 26, p. 193 - 207, 1990.

LARSON, R. C.; ODONI, A. R. *Urban Operations Research*. New Jersey: Prentice-Hall, 1981.

LAWLER, E.; LENSTRA, J.; RINNOOY, K. A., SHMOYS, D. The traveling salesman

LENSTRA, J. K.; SHMOYS, D. B. & Tardos, E. Approximation algorithms for scheduling unrelated parallel machines. Mathematical Programming: Series A and B, v. 46, p. 259 – 271. 1990.

LOIOLA, E. M.; ABREU, N. M. M.; BOAVENTURA NETTO, P. O. Uma revisão comentada das abordagens do problema quadrático de alocação. Pesquisa Operacional, v. 24, n. 1, p. 73-109, 2004.

MAINIERI, G. B. Metaheurística BRKGA aplicada a um problema de programação de tarefas no ambiente flowshop híbrido. São Paulo 2014, Tese de doutorado - Escola Politécnica da USP, p. 110, SP, 2014.

MANASSE, M. S.; MCGEOCH, L. A.; SLEATOR, D. D. Competitive algorithms for online problems. Proceedings of the twentieth annual ACM symposium on Theory of computing, ACM Press, p. 322–333. 1988.

MAZZOLA, J. B.; NEEBE, A. W.; DUNN, C. V. R. *Production planning of a flexible manufacturing system in a material requirements planning environment.* International Journal of Flexible Manufacturing Systems, v. 1, p. 115–142. 1989.

MEGALE, G. L. B. *Algoritmo genético de chaves aleatórias viciadas aplicado ao problema de clusterização de módulos de software.* Dissertação (Mestrado em Informática)-UFRJ, p. 102, Rio de Janeiro, 2015.

MEIRA Jr., W. Redes Metropolitanas de Alta Velocidade. Boletim bimestral sobre tecnologia de redes produzido e publicado pela RNP, v. 3, n. 6, 1999. Disponível em < https://memoria.rnp.br/newsgen/9911/rmav.html >. Acesso 02 de jan. de 2018.

MINIEKA, E. *The Centers and Medians of a graph.* Operations Research, v. 25, p. 641 – 650, 1977.

MIYAZAWA, F. K. Otimização Combinatória. UNICAMP. Disponível em < https://www.ic.unicamp.br/~fkm/problems/combopt.html >. Acesso 02 de dez. de 2017.

MITCHELL, M. *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1996.

MUNHOZ, P. L. A.; OCHI, L. S.; SOUZA, M. J. F. Um algoritmo baseado em *iterated* local *search* para o problema de roteamento de veículos periódico. Anais. In: XXXII Encontro Nacional de Engenharia de Produção (ENEGEP), Bento Gonçalves, 2012.

NARCISO, M. G.; LORENA, L. A. N. *Lagrangean / surrogate relaxation for generalized assignment problems.* European Journal of Operational Research, v. 114, p. 165 – 177. 1999.

NAUSS, R. M. *Solving the generalized assignment problem: An optimizing and heuristic approach.* INFORMS Journal on Computing, v. 15, p. 249 – 266. 2003.

NILS, J. N. Principles of Artificial Inteligence. Morgan kaufmann, first edition. 1982.

NORONHA, T. F.; RESENDE, M. G. C.; RIBEIRO, C. C. A biased random-key genetic algorithm for routing and wavelength assignment. Journal of Global Optimization, n.17, p.487–525, 2011.

OLIVEIRA, E. S. de. *Um Algoritmo Genético de Chaves Aleatórias Viciadas para o problema de Atracamento Molecular*. Dissertação de mestrado, p. 98, Porto Alegre: PPGC da UFRGS, 2016.

O'Reilly, T. What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. O'Reilly. Disponível em <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>. Acesso 04 de jan. de 2018.

OSMAN, I. H. *Heuristics for the generalized assignment problem: Simulated Annealing and Tabu Search approaches.* OR Spektrum, v. 17, p. 211 – 225. 1995.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial optimization: algorithms and complexity*. Upper Saddle Rive, NJ, USA: Prendice-Hall, Inc., 1982.

PEARL, J. Heuristics-itelligent search strategies for computer problem solving. Boston: Reading, Addison-Wesley, 1984.

PEDRO, O. R. Uma abordagem de Busca Tabu para o Problema do Caixeiro Viajante com Coleta de Prêmios. Dissertação (Mestrado em Engenharia Elétrica), Universidade Federal de Minas Gerais, 2013.

PEREIRA, M. A. Um método Branch-and-Price para problemas de localização de p-medianas. Tese de Doutorado. INPE, São José dos Campos, 2005.

PIERSKALLA, W. P. *The multidimensional assignment problem.* Operations Research, v. 16, p. 422 – 431, 1968.

PIZZOLATO, N. D. Problemas de localização. Rio de Janeiro, 2000.

POLI, R.; LANGDON, W. Genetic Programming with One-Point Crossover. In: CHAWDHRY, P.; ROY, R.; PANT, R. (Ed.). Soft Computing in Engineering Design and Manufacturing. Springer London, p. 180–189. 1998.

PRASETYO, H.; FAUZA G., A. Y.; LEE, S. H. Survey on applications of biased-random key genetic algorithms for solving optimization problems. In: Industrial Engineering and Engineering Management (IEEM). p. 863–870. 2015.

PUSZTASZERI, J.; RENSING, P. E.; LIEBLING, T. M. *Tracking elementar particles near their primary vertex: A combinatorial approach.* Journal of Global Optimization, v. 9, n. 1, p. 41-64, 1996.

REEVES, M. Modern heuristic techniques for combinatorial problems. New York: John Wiley & Sons, 1993.

REEVES, C. R. Genetic Algorithms for the Operation Researcher. INFORMS Journal on Computing, v. 9, n. 3, p. 231–250, 1997.

RESENDE, M. G. C. Biased random-key genetic algorithms with applications in telecommunications. J. Span. Soc. of Stat. Oper. Res., v. 20, n. 1, p. 130–153, 2012.

ROSS, G. T.; SOLAND, R. M. Modeling facility location problems as generalized assignment problems. Management Science, v. 24, p345–357. 1977.

SAHNI, S.; GONZALEZ, T. P-complete approximation problems. Journal of the ACM, v. 23, p. 555 – 565.1976.

SANCHES, B. C. S.; MANIC, G. Algoritmos Para Fluxo Máximo em Redes. Centro de Matemática, Computação e Cognição, Universidade Federal do ABC. Santo André-SP, 2008.

SENNE, Edon Luiz França;LORENA, Luiz Antonio Nogueira; SALOMÃO, Silvely Nogueira de Almeida. Métodos de geração de colunas para problemas de atribuição. Rev. Produção, v. 17, n. 1, p. 071-083, São Paulo-SP, 2007.

SILVA, A. Optimização da Recolha de Resíduos Urbanos. Dissertação para a obtenção do grau de Mestre em Engenharia do Ambiente, Universidade de Aveiro. 2009.

SILVEIRA, R. M. Redes de Alta Velocidade e as Aplicações Multimídia. USP, São Paulo, 2006. Disponível em < http://rmav-sp.larc.usp.br/Documentos/RAVeAplic.pdf >. Acesso em 02 de jan. de 2018.

SLEATOR, D.; TARJAN, R. *Armotized Efficiency of List Update and Paging Rules*. Communications of the ACM, v. 28, p. 202 – 208, 1988.

SODRÉ, Ulissys. Modelos matemáticos. Londrina PR, 2007. Disponível em: < http://www.uel.br/projetos/matessencial/superior/pdfs/modelos.pdf > . Acesso 25 de dez. de 2017.

SOUZA, M. J. F. Inteligência Computacional para Otimização. Instituto de Ciências exatas e biológicas, Dep. de computação, p. 59, 2008.

SPEARS, W.; DEJONG, K. On the virtues of parameterized uniform crossover. In Belew, R. e BOOKER, L., Proceedings of the Fourth International Conference on Genetic Algorithms, pages 230–236, San Mateo, Italy. 1991.

STEENBRINK, P. A. *Optimization of Transport Networks.* Ed. John Wiley & Sons, 1974.

STEINBERG, L. *The backboard wiring problem: A placement algorithm*. SIAM Review, v. 3, p. 37 – 50. 1961.

SYSLO, M. M.; NARSINGH, D.; KOWALIK, J.S. *Discrete Optimization Algorithms With* Pascal Programs. Prentice Hall, Inc. Englewood Cliffs. 1983.

TANGPATTANAKUL, P.; JOZEFOWIEZ, N.; LOPEZ, P. Biased random key genetic algorithm with hybrid decoding for multi-objective optimization. In: FedCSIS, 2013 Fede-rated Conference on. p. 393–400. 2013.

TRAGANTALERNGSAK, S.; HOLT, J.; RONNQVIST, M. An exact method for the twoechelon, single-source, capacitated facility location problem. European Journal of Operational Research, v.123, n.3, p.473 – 489, 2000.

TUREK, J.; WOLF, J. L.; YU, P. S. *Approximate algorithms scheduling parallelizable tasks*. Proceedings of ACM Symposium on Parallel Algorithms and Architectures, San Diego, USA, p. 323 – 332. 1992.

WANG, L.; GU, W. *Genetic algorithm with stochastic ranking for optimal channel assignment in mobile communications*. Lecture Notes in Computer Science, v. 3314, p. 154-159, 2004.

WRIGHT, M. B. *Speeding up the Hungarian algorithm.* Computers and Operations Research, v. 17, n. 1, p. 95 – 96, 1990.

YAGIURA, M.; IBARAKI, T.; GLOVER, F. *A path relinking approach for the generalized assignment problem*. in: Proceedings of the International Symposium on Scheduling, Japan, June 4–6, pp. 105–108. 2002.

APENDICE A

Clientes prioritários e Candidatos
UERN (Campus central)
UFERSA
IFRN
UERN (REITORIA)
UERN (FACS)
FAC. MATHER CRISTH
FAC. DIOCESANA
UNP
UNIT
Unopar
Hosp. Wilson Rosado
Rebouças Albeto M.
Rebouças Av. Pres. Dutra
Thermas
Garbos
Villa Oeste
Partage Shopping
BB Alberto Maranhão
BB Centro
BB Ilha de SL
BB Stilo
Caixa Centro
Caixa Av. Pres. Dutra
Bradesco
Santander

APENDICE B

DADOS EM METROS

	UERN (Camp. central)	UFERSA	IFRN	UERN (REITORIA)	UERN (FACS)	FAC. MATHER CRISTH	FAC. DIOCESANA	UNP	UNIT	Unopar	Hosp. Wilson Rosado	Rebouças Albeto M.	Rebouças Av. Pres. Dutra	Hotel Thermas	Hotel Garbos	Hotel Villa Oeste	Partage Shopping	BB Alberto Maranhão	BB Centro	BB Ilha de SL	BB Stilo	Caixa Economica Centro	Caixa Economica Av. Pres. Dutra	Bradesco	Santander	Polícia Civil 2ª DP	Central do Cidadão/DETRAN-RN	Ministério Público Federal	Ministério Publico do RN	Forúm Desembargador Silveira Martins	Fórum Trabalhista de Mossoró, 21ª Região	Aeroporto	Secretaria Municipal da Fazenda de Mossoró	Secretaria da Receita Federal do Brasil	Prefeitura de Mossoró	Centro Administrativo da Prefeitura de Mossoró	Intertv Costa Branca	Polícia Federal	Atacadão	Maxxi Atacado
UERN (Campus central)	-	1200	2800	3500	7200	4400	4500	7900	3400	4000	4200	3600	2700	6900	6300	3400	11300	4300	3700	3400	3900	4000	3200	3900	3700	4100	8800	1800	3300	2900	2800	6600	5200	4200	4200	6400	1900	5800	7900	7100
UFERSA	1200	-	1600	2900	6600	4200	3900	7300	2400	3000	3800	3200	1700	3500	5900	2500	11000	3800	3200	2400	2700	3600	2200	3400	3300	3000	8200	650	2200	1700	1600	5900	4000	3700	3700	5900	700	5200	7700	6900
IFRN	2800	1600	-	4500	8300	5500	5500	11000	4100	4300	5300	4700	2000	7900	7400	2900	12300	4700	4200	3000	3800	5000	2900	4200	4100	1300	9500	1000	300	400	280	7200	5200	4700	4700	7200	950	6500	9000	8100
UERN (REITORIA)	3500	2000	4500	-	3000	1300	1200	4600	1700	1100	900	750	3200	4000	3400	1700	4800	1000	700	1700	2000	1600	2000	750	600	4100	5500	3600	5100	4700	1600	3100	1300	700	700	3100	3700	2500	4600	3800
UERN (FACS)	7200	6600	8300	3900	-	3400	3500	3100	4500	3900	3600	4500	5900	4500	4700	4400	3300	3200	4600	4400	4000	3800	4800	3800	4000	7000	2100	6700	8200	7800	7700	1200	3400	4000	4000	1400	6800	1900	3200	2300
FAC. MATHER CRISTH	4400	4200	5500	1300	3400	-	130	3600	3000	2400	900	1200	4500	2400	1900	3000	3600	2200	2200	3000	2500	2400	3200	2300	2300	5100	4100	4400	5900	5500	5400	2400	2000	1400	1400	2200	4900	1500	3700	2800
FAC. DIOCESANA	4500	3900	5500	1200	3500	130	-	3600	2400	1800	300	1100	3900	2800	2200	2300	4100	1500	1500	2400	2000	1700	2700	1700	1700	5000	4700	4500	6000	5600	5400	2300	1800	1200	1200	2100	4600	1600	3600	2600
UNP	7900	7300	11000	4600	3100	3600	3600	-	4800	4200	3300	4200	6300	2800	3200	4800	800	3700	4200	4800	4300	4100	5100	4100	4200	8000	1700	7500	9000	8100	8000	3100	3900	3700	3700	2300	7100	1600	600	300
UNIT	3400	2400	4100	1700	4500	3000	2400	4800	-	650	2100	2200	1400	4900	4800	73	5600	1400	1000	55	600	1200	230	900	950	2600	8000	2200	3700	3300	3200	5400	3400	4000	4000	5500	2300	5100	7200	6300
Unopar	4000	3000	4300	1100	3900	2400	1800	4200	650	-	1500	1400	2100	4300	4200	600	5000	800	700	600	100	650	900	1300	1350	3200	5800	2900	4400	4000	3800	3200	1300	1000	1000	3100	2900	2900	4800	4000
Hosp. Wilson Rosado	4200	3800	5300	900	3600	900	300	3300	2100	1500	-	950	3600	3100	2800	2100	4000	1200	1100	2000	1500	1400	2400	1400	1450	4600	4600	4300	5900	5300	5200	2500	1500	950	950	2300	4300	1600	3800	4000
Rebouças Albeto M.	3600	3200	4700	750	4500	1200	1100	4200	2200	1400	950	-	3600	3300	2900	2100	5000	1100	1000	1800	1400	1400	2100	1500	1300	5700	5400	3600	5100	4700	1300	3100	1400	850	850	3100	3700	2400	4600	4700
Rebouças Av. Pres. Dutra	2700	1700	2000	3200	5900	4500	3900	6300	1400	2100	3600	3600	-	6400	6200	1500	7100	2800	2700	1500	2000	2700	1200	2700	2800	1200	7900	1500	2500	2600	2400	5200	3600	3100	3100	5300	1500	4900	7100	6200
Thermas	6900	3500	7900	4000	4500	2400	2800	2800	4900	4300	3100	3300	6400	-	500	4900	3700	4100	4200	4400	4900	4300	5400	4300	4500	7800	3700	6600	8700	7700	7600	4200	4400	3800	3800	4000	6700	6700	3500	2700
Garbos	6300	5900	7400	3400	4700	1900	2200	3200	4800	4200	2800	2900	6200	500	-	3900	4100	4100	4300	5000	4500	4200	5200	4300	4400	7200	3900	6800	8900	7900	7800	4400	4600	4000	4000	4200	7300	6900	3700	2900
Villa Oeste	3400	2500	2900	1700	4400	3000	2300	4800	73	600	2100	2100	1500	4900	3900	-	5500	1400	1200	56	450	1200	250	900	1000	2600	2300	2300	3800	3400	3200	5600	3500	4100	4100	5600	2300	5100	7300	6500
Partage Shopping	11300	11000	12300	4800	3300	3600	4100	800	5600	5000	4000	5000	7100	3700	4100	5500	-	4500	4900	5600	5300	5100	5800	4900	5000	8500	2700	8000	9500	9100	9000	3900	4900	4600	4600	3100	8100	2700	300	1300
BB Alberto Maranhão	4300	3800	4700	1000	3200	2200	1500	3700	1400	800	1200	1100	2800	4100	4100	1400	4500	-	700	1400	900	260	1600	700	750	4100	5600	3600	5100	4700	4500	2200	270	290	290	2200	3600	2500	4600	3700
BB Centro	3700	3200	4200	700	4600	2200	1500	4200	1000	700	1100	1000	2700	4200	4300	1200	4900	700	-	1000	650	550	1500	190	140	3700	8100	2200	3800	3400	3200	5600	1700	1400	1400	3500	3200	3300	5300	4600
BB Ilha de SL	3400	2400	3000	1700	4400	3000	2400	4800	55	600	2000	1800	1500	4400	5000	56	5600	1400	1000	-	500	1200	300	1200	1300	2600	5700	3700	5200	4800	4600	3300	1300	1200	3300	3100	3400	2700	4800	4000
BB Stilo	3900	2700	3800	2000	4000	2500	2000	4300	600	100	1500	1400	2000	4900	4500	450	5300	900	650	500	-	650	850	1300	1200	3200	5700	3300	4900	4800	4600	3300	1300	1200	1200	3300	3400	2700	4800	4000
Caixa Economica Centro	4000	3600	5000	1600	3800	2400	1700	4100	1200	650	1400	1400	2700	4300	4200	1200	5100	260	550	1200	650	-	1500	650	700	3900	4900	3400	5400	4500	4400	2400	550	800	800	2400	3500	3000	4800	3900
Caixa Economica Av. Pres. Dutra	3200	2200	2900	2000	4800	3200	2700	5100	230	900	2400	2100	1200	5400	5200	250	5800	1600	1500	300	850	1500	-	1100	1150	2400	6500	3400	5300	5600	4400	3900	2000	1700	1700	3800	3500	3500	5600	4800
Bradesco	3900	3400	4200	750	3800	2300	1700	4100	900	1300	1400	1500	2700	4300	4300	900	4900	700	190	1200	1300	650	1100	-	50	3500	5400	3400	5000	4500	4400	3100	1000	600	600	2900	3600	2400	4300	3500
Santander	3700	3300	4100	600	4000	2300	1700	4200	950	1350	1450	1300	2800	4500	4400	1000	5000	750	140	1300	1200	700	1150	50	-	3600	5600	3500	5100	4600	4500	3000	1200	750	750	3000	3600	2600	4700	3900
Polícia Civil 2ª DP	4100	3000	1300	4100	7000	5100	5000	8000	2600	3200	4600	5700	1200	7800	7200	2600	8500	4100	3700	2600	3200	3900	2400	3500	3600	0	8300	2300	1800	1700	1600	6200	4400	4100	4100	6200	2600	5900	8100	7200
Central do Cidadão/DETRAN-RN	8800	8200	9500	5500	2100	4100	4700	1700	8000	5800	4600	5400	7900	3700	3900	2300	2700	5600	8100	5700	5700	4900	6500	5400	5600	8300	0	9000	10600	10200	10000	3600	5400	5100	5100	2800	9100	3200	2400	1400
Ministério Público Federal	1800	650	1000	3600	6700	4400	4500	7500	2200	2900	4300	3600	1500	6600	6800	2300	8000	3600	2200	3700	3300	3400	3400	3400	3500	2300	9000	0	1500	1100	1000	5800	4000	3600	3600	5800	82	5800	7600	7000
Ministério Público do RN	3300	2200	300	5100	8200	5900	6000	9000	3700	4400	5900	5100	2500	8700	8900	3800	9500	5100	3800	5200	4900	5400	5300	5000	5100	1800	10600	1500	0	400	550	7300	5500	5200	5200	7400	1400	7200	9100	8100
Forúm Desembargador Silveira Martins	2900	1700	400	4700	7800	5500	5600	8100	3300	4000	5300	4700	2600	7700	7900	3400	9100	4700	3400	4800	4800	4500	5600	4500	4600	1700	10200	1100	400	0	130	6900	5100	4700	6900	6900	1000	6600	8700	7700
Fórum Trabalhista de Mossoró, 21ª Região	2800	1600	280	4600	7700	5400	5400	8000	3200	3800	5200	1300	2400	7600	7800	3200	9000	4500	3200	4600	4600	4400	4400	4400	4500	1600	10000	1000	550	130	0	6800	4900	4600	4600	6700	900	6400	8600	7600
Aeroporto	6600	5900	7200	3100	1200	2400	2300	3100	5400	3200	2500	3100	5200	4200	4400	5600	3900	2200	5600	3300	3300	2400	3900	3100	3000	6200	3600	5800	7300	6900	6800	0	2300	2300	2300	750	5500	2400	3800	3000
Secretaria Municipal da Fazenda de Mossoró	5200	4000	5200	1300	3400	2000	1800	3900	3400	1300	1500	1400	3600	4400	4600	3500	4900	270	1700	1300	1300	550	2000	1000	1200	4400	5400	4000	5500	5100	4900	2300	0	550	550	2200	3900	2900	4600	3800
Secretaria da Receita Federal do Brasil	4200	3700	4700	700	4000	1400	1200	3700	4000	1000	950	850	3100	3800	4000	4100	4600	290	1400	1200	1200	800	1700	600	750	4100	5100	3600	5200	4700	4600	2300	550	0	10	2400	3600	2200	4300	3400
Prefeitura de Mossoró	4200	3700	5200	700	4000	1400	1200	3700	4000	1000	950	850	3100	3800	4000	4100	4600	290	1400	3300	1200	800	1700	600	750	4100	5100	3600	5200	6900	4600	2300	550	10	0	2400	3600	2200	4300	3400
Centro Administrativo da Prefeitura de Mossoró	6400	5900	7200	3100	1400	2200	2100	2300	5500	3100	2300	3100	5300	4000	4200	5600	3100	2200	3500	3100	3300	2400	3800	2900	3000	6200	2800	5800	7400	6900	6700	750	2200	2400	2400	0	6300	1300	3200	2300
Intertv Costa Branca	1900	700	950	3700	6800	4900	4600	7100	2300	2900	4300	3700	1500	6700	7300	2300	8100	3600	3200	3400	3400	3500	3500	3600	3600	2600	9100	82	1400	1000	900	5500	3900	3600	3600	6300	0	5600	7700	6800
Polícia Federal	5800	5200	6500	2500	1900	1500	1600	1600	5100	2900	1600	2400	4900	6700	6900	5100	2700	2500	3300	2700	2700	3000	3500	2400	2600	5900	3200	5800	7200	6600	6400	2400	2900	2200	2200	1300	5600	0	2200	1300
Atacadão	7900	7700	9000	4600	3200	3700	3600	600	7200	4800	3800	4600	7100	3500	3700	7300	300	4600	5300	4800	4800	4800	5600	4300	4700	8100	2400	7600	9100	8700	8600	3800	4600	4300	4300	3200	7700	2200	0	850
Maxxi Atacado	7100	6900	8100	3800	2300	2800	2600	300	6300	4000	4000	4700	6200	2700	2900	6500	1300	3700	4600	4000	4000	3900	4800	3500	3900	7200	1400	7000	8100	7700	7600	3000	3800	3400	3400	2300	6800	1300	850	0

APENDICE C

Penalidades	
UERN (Campus central)	1.000
UFERSA	1.000
IFRN	1.000
UERN (REITORIA)	1.000
UERN (FACS)	1.000
FAC. MATHER CRISTH	90
FAC. DIOCESANA	90
UNP	90
UNIT	90
Unopar	90
Hosp. Wilson Rosado	80
Rebouças Albeto M.	70
Rebouças Av. Pres. Dutra	70
Thermas	60
Garbos	60
Villa Oeste	60
Partage Shopping	70
BB Alberto Maranhão	75
BB Centro	75
BB Ilha de SL	75
BB Stilo	75
Caixa Centro	75
Caixa Av. Pres. Dutra	75
Bradesco	75
Santander	75

APENDICE D

Cálculo do prêmio ao visitar um cliente				
Candidatos	**Distância em relação a UERN**	**Peso Distância**	**Porte (Financeiro / Demanda / Criticidade)**	**Prêmio final**
FAC. MATHER CRISTH	4.400	0,227	70	15,89
FAC. DIOCESANA	4.500	0,222	70	15,54
UNP	7.900	0,127	90	11,43
UNIT	3.400	0,294	65	19,11
Unopar	4.000	0,250	65	16,25
Hosp. Wilson Rosado	4.200	0,238	88	20,94
Rebouças Albeto M.	3.600	0,278	60	16,68
Rebouças Av. Pres. Dutra	2.700	0,370	60	22,20
Thermas	6.900	0,145	50	7,25
Garbos	6.300	0,159	50	7,95
Villa Oeste	3.400	0,294	50	14,70
Partage Shopping	11.300	0,088	88	7,74
BB Alberto Maranhão	4.300	0,233	75	17,48
BB Centro	3.700	0,270	75	20,25
BB Ilha de SL	3.400	0,294	75	22,05
BB Stilo	3.900	0,256	75	19,20
Caixa Centro	4.000	0,250	75	18,75
Caixa Av. Pres. Dutra	3.200	0,313	75	23,48
Bradesco	3.900	0,256	75	19,20
Santander	3.700	0,270	75	20,25

APENDICE E

C:\TSP_OK.cpp 1

```
//=====================================================================
=====================
// PROGRAMA: TSP_OK.cpp -- Resolve o problema do caixeiro viajante com coleta de pre-
mios
// DADOS DE ENTRADA:
// dist[n][n] - Matriz simétrica de distancia entre as cidade a serem visistadas;
// penal[n] - Vetor de penalidades (valor infinito para cidades consideradas obrigatórias);
// premio[n] - Vetor de permiação (valor zero para cidades consideradas obrigatórias).
// DADOS DE SAÍDA:
// Valor da Função Objetivo;
// Lista das cidades visitadas;
// Lista das cidades não visitadas;
// Valor do Prêmio coletado;
// Valor da penalidade paga;
// Distância percorrida.
// MODELO - Modelagem matemática proposta por Egon Balas.
//=====================================================================
=====================
#include <ilcplex/ilocplex.h>
#include "multiDimvarMap.h"
#include <time.h>
#ifdef FULLTEST
#include <assert.h>
IloBool cutCalled = IloFalse;
#endif
ILOSTLBEGIN
int main(int argc, char **argv) {
IloEnv env; IloInt n;
char D[15],LP[15];
char R[15];
char NArq[5];
try {
const char* file;
if ( argc == 2 ) file = argv[1];
cout<< "QUAL O TAMANHO DA INSTANCIA [5, 10, 11, 20, 21, 27, 40]? ";
cin>>NArq;
n=atoi(NArq);
strcpy(D,"PCV_CP");
strcat(D,NArq);
strcpy(LP,D);
strcpy(R,D);
strcat(R,"_90%");
strcat(D,".dat");
strcat(LP,".LP");
strcat(R,".txt");
file=D;
clock_t TempoExecucao;
```

```
IloNumArray2 dist(env,n);
IloNumArray premio(env,n);
IloNumArray penal(env,n);
IloNum premioMinimo, soma;
// ------------------------------------------------------------
// Leitura de dados
// ------------------------------------------------------------
ifstream data(file);
if ( !data ) throw(-1);
data >> dist >> penal >> premio;
n = dist.getSize();
IloInt i, j;
for(j=0; j<n; j++)
{
soma+=premio[j];
}
premioMinimo=soma*0.9; // Definido o premio acumulado mínimo
// ------------------------------------------------------------
// Descrição do Modelo
// ------------------------------------------------------------
// VARIÁVEL BINÁRIA x[i][j] para todo i,j = 0, ..., n
// x[i][j] == 1 Se o arco (i,j) é utilizado
// x[i][j] == 0 Caso contrário
//
// VARIÁVEL BINÁRIA y[i] para todo i = 0, ..., n
C:\TSP_OK.cpp 2
// y[i] == 1 Se a cidade i é visitada
// y[i] == 0 Caso contrário
//
// VARIÁVEL INTEIRA t[i] Para todo i = 0, ..., n - 1
// u[i] == k Se o nó i é o k-éssimo nó na rota
// u[0] == 1
// u[i] em [2, ..., n] Para todo i = 1, ... n - 1
//
// FUNÇÃO OBJETIVO
// MIN Somatório((i,j), d[i][j] * x[i][j]) + p[i]*(1-y[i])
//
// RESTRIÇÕES
// 1) Somatório(j, x[i][j]) == y[j] para todo i
// 2) Somatório(i, x[i][j]) == y[i] para todo j
// 3) u[i] - u[j] + n * x[i][j] <= n - 1 para todo i,j = 1, ..., n - 1
// 4) Somatório(i, w[i])>= PremioMinimo para todo i
// ------------------------------------------------------------------------
// ------------------------------------------------------------
// Criação do Modelo
// ------------------------------------------------------------
IloModel pcvcp(env);
#ifdef FULLTEST
IloCplex cpxtest(pcvcp);
```

```
#endif
// VARIÁVEIS PARA O MODELO
// ------------------------------------------------------------
IloArray<IloNumVarArray> x(env, n);
IloNumVarArray u(env, n), y(env,n);
// RESTRIÇÕES COM USO DE ILORANGE NO MODELO
// ------------------------------------------------------------
IloArray<IloRangeArray> fluxo(env, n); // Restrição 3)
std::stringstream name;
// CRIANDO A ESTRUTURA PARA AS VARIÁVEIS DO MODELO
// ------------------------------------------------------------
// Variável u: u[0] tem valor fixo = 1
u[0] = IloNumVar(env, 1, 1, IloNumVar::Int, "u_0");
// Variável u: demais posições u[1], ..., u[n]
for(auto i = 0; i < n; ++i) {
name << "u_" << i;
u[i] = IloNumVar(env, 2, n, IloNumVar::Int, name.str().c_str());
name.str(""); // limpando nome fantasia
}
//Criando variável y: y[0], ..., y[n]
for(auto i = 0; i < n; ++i) {
name << "y_" << i;
y[i] = IloNumVar(env, 0, 1, IloNumVar::Bool, name.str().c_str());
name.str(""); // limpando nome fantasia
}
// Criando variável x: x[1,..,n][1,..,n]
for(auto i = 0; i < n; ++i) {
x[i] = IloNumVarArray(env, n);
for(auto j = 0; j < n; ++j) {
name << "x_" << i << "_" << j;
x[i][j] = IloNumVar(env, 0, 1, IloNumVar::Bool, name.str().c_str());
name.str(""); // limpando nome fantasia
}
}
// CRIANDO AS RESTRIÇÕES PARA O MODELO
// ------------------------------------------------------------
IloExpr expr(env);
// Restrição 1: Somatório(j, x[i][j]) == y[j] para todo i,j: 1,..,n
for(auto j = 0; j < n; ++j) {
for(auto i = 0; i < n; ++i) {
expr += x[i][j];
}
C:\TSP_OK.cpp 3
pcvcp.add(expr==y[j]);
expr.clear(); // limpando a varável expr
}
// Restrição 2: Somatório(i, x[i][j]) == y[i] para todo i,j: 1,..,n
for(auto i = 0; i < n; ++i) {
for(auto j = 0; j < n; ++j) {
```

```
expr += x[i][j];
}
pcvcp.add(expr==y[i]);
expr.clear(); // limpando a varável expr
}
// Restrição 3: Eliminação de subrotas e continuidade de fluxo (Uso de IloRange)
fluxo[0] = IloRangeArray(env); // Para i == 0, restrição vazia.
for(auto i = 1; i < n; ++i) {
fluxo[i] = IloRangeArray(env, n);
for(auto j = 1; j < n; ++j) {
expr = u[i] - u[j] + static_cast<int>(n) * x[i][j];
name << "fluxo_" << i << "_" << j;
fluxo[i][j] = IloRange(env, -IloInfinity, expr, n - 1, name.str().c_str());
name.str(""); // limpando nome fantasia
expr.clear(); // limpando a varável expr
}
pcvcp.add(fluxo[i]);
}
// Restrição 4: Somatório(i, w[i])>= PremioMinimo para todo i
IloExpr SomaPremio(env);
for (i = 0; i < n; i++)
SomaPremio+=premio[i]*y[i];
pcvcp.add(SomaPremio >= premioMinimo);
// CRIANDO A FUNÇÃO OBJETIVO
// ------------------------------------------------------------
IloExpr Pena(env);
IloExpr Distancia(env);
for(auto i = 0; i < n; ++i) {
Pena += penal[i]*(1-y[i]);
for(auto j = 0; j < n; ++j) {
Distancia+= x[i][j]*dist[i][j];
}
expr = Distancia + Pena;
}
IloObjective obj(env, expr, IloObjective::Minimize);
pcvcp.add(obj);
expr.end();
TempoExecucao = clock();
// RESOLUÇÃO DO MODELO
// ------------------------------------------------------------
IloCplex cplex(pcvcp);
cplex.exportModel(LP);
bool resolvido = false;
try {
resolvido = cplex.solve();
} catch(const IloException& e) {
std::cerr << std::endl << std::endl;
std::cerr << "CPLEX Raised an exception:" << std::endl;
std::cerr << e << std::endl;
```

```
env.end();
throw;
}
if(resolvido){
std::cout << std::endl << std::endl;
std::cout << "\nMODELO RESOLVIDO COM SUCESSO!" << std::endl;
std::cout << "\nSTATUS: " << cplex.getStatus() << std::endl;
std::cout << "\nVALOR DA FUNCAO OBJETIVO: " << cplex.getObjValue() << std::endl;
} else {
std::cerr << "\nERRO NA RESOLUCAO!" << std::endl;
std::cerr << "\nSTATUS: " << cplex.getStatus() << std::endl;
std::cerr << "\nSOLUCIONAR: " << cplex.getCplexStatus() << std::endl;
C:\TSP_OK.cpp 4
}
TempoExecucao = clock() - TempoExecucao;
float Tempo_Seg=((float)TempoExecucao)/CLOCKS_PER_SEC;
float Tempo_Min = ((float)TempoExecucao)/CLOCKS_PER_SEC/60;
// TELA DE RELATÓRIO
// ------------------------------------------------------------
cplex.out()<<"============================================================
================= "<<endl;
cplex.out()<<" RESULTADO DETALHADO "<<endl;
cplex.out()<<"============================================================
================= "<<endl;
cplex.out()<<"\n CIDADES VISITADAS: ";
for(i=0; i<n; i++)
if (cplex.getValue(y[i])==1) cplex.out()<<i<<", ";
cplex.out()<<"\n\n CIDADES NAO VISITADAS: ";
for(i=0; i<n; i++)
if (cplex.getValue(y[i])==0) cplex.out()<<i<<", ";
cplex.out()<<endl;
cplex.out()<<"\n PREMIO MINIMO PERMITIDO = "<<premioMinimo<<endl;
cplex.out()<<endl;
cplex.out()<<" PREMIO COLETADO = "<<cplex.getValue(SomaPremio)<<endl;
cplex.out()<<endl;
cplex.out()<<" PENALIDADE PAGA = "<<cplex.getValue(Pena)<<endl;
cplex.out()<<endl;
cplex.out()<<" DISTANCIA PERCORRIDA = "<<cplex.getValue(Distancia)<<endl<<"\n";
cplex.out()<<" TEMPO DE EXECUCAO EM SEGUNDOS = "<<Tempo_Seg<<endl;
cplex.out()<<endl;
cplex.out()<<" TEMPO DE EXECUCAO EM MINUTOS = "<<Tempo_Min<<endl;
cplex.out()<<endl;
cplex.out()<<"============================================================
================= "<<endl;
//-------------------------------------------------------------------------------------
// GRAVANDO RESULTADOS WM ARQUIVO
//-------------------------------------------------------------------------------------
FILE *arq;
arq = fopen(R, "w");
```

```
if(arq == NULL)
printf("Erro, nao foi possivel abrir o arquivo\n");
else{
fprintf(arq, "\n RESULTADO OTIMO DA INSTANCIA %s",D);
fprintf(arq, "\n VALOR DA FUNCAO OBJETIVO = %8.2f",cplex.getObjValue());
fprintf(arq, "\n CIDADES VISITADAS :");
for(i=0; i<n; i++)
if (cplex.getValue(y[i])==1) fprintf(arq, "%d , ",i);
fprintf(arq, "\n CIDADES NAO VISITADAS :");
for(i=0; i<n; i++)
if (cplex.getValue(y[i])==0) fprintf(arq, "%d , ",i);
fprintf(arq,"\n PREMIO MINIMO PERMITIDO = %8.2f",premioMinimo);
fprintf(arq,"\n PREMIO COLETADO = %8.2f",cplex.getValue(SomaPremio));
fprintf(arq,"\n PENALIDADE PAGA = %8.2f",cplex.getValue(Pena));
fprintf(arq,"\n DISTANCIA PERCORRIDA = %8.2f",cplex.getValue(Distancia));
fprintf(arq,"\n TEMPO DE EXECUCAO EM SEGUNDOS = %8.2f",Tempo_Seg);
}
fclose(arq);
//------------------------------------------------------------------------------------
Pena.end();
Distancia.end();
SomaPremio.end();
#ifdef FULLTEST
assert ( cplex.getImpl()->isConsistent() );
assert ( cpxtest.getImpl()->isConsistent() );
assert ( cplex.getStatus() == IloAlgorithm::Optimal );
assert ( fabs (cplex.getObjValue() - 11461.0) < 1e-6 );
assert (cutCalled);
env.out() << "Test completed successfully" << endl;
#endif
}
catch (const IloException& e) {
cerr << "Exception caught: " << e << endl;
#ifdef FULLTEST
assert (0);
#endif
}
catch (...) {
C:\TSP_OK.cpp 5
cerr << "Unknown exception caught!" << endl;
#ifdef FULLTEST
assert (0);
#endif
}
env.end();
system("pause");
return 0;
}
```

APENDICE F

```
Tools
import os
from numpy import *
import random
import rota
import timeit
from copy import copy
import statistics
from fractions import Fraction as F
from decimal import Decimal as D


def lerarquivo(nomearquivo):
    ref_arquivo = open(nomearquivo, "r")
    linhas = ref_arquivo.readlines()
    ref_arquivo.close()
    return linhas


def criarmatrizdistancia(stringsarquivo, ncidades):
    matrizdistancia = []
    for l in range(0, ncidades):
        colunas = []
        linhastring = stringsarquivo[l].replace("\n", "").split(';')
        for c in range(0, ncidades):
            colunas.append(linhastring[c])
        matrizdistancia.append(colunas)
    return matrizdistancia


def criarvetorvalores(stringsarquivo, ncidades):
    colunas = []
    linhastring = stringsarquivo[0].replace("\n", "").split(';')
    for c in range(0, ncidades):
        colunas.append(linhastring[c])

    return colunas


def solucaogulosaaleatoria(porcentagemcorte, ncidades, distancia, premiomi-
nimo, premios):
    # Sorteando a primeira cidade
    # cidadeinicial = random.choice(range(ncidades))
    cidadeinicial = 0
    premioatual = float(premios[0])
    # print(cidadeinicial)


def calculacusto(caminho, matrizdistancia):
    custo = 0
    for i in range(len(caminho) - 1):
        custo += float(matrizdistancia[caminho[i]][caminho[i + 1]])

    custo += float(matrizdistancia[caminho[len(caminho) - 1]][caminho[0]])
    return custo
```

```
def gerarcaminhoaleatorio(ncidades):
    caminhoaleatiorio = random.sample(range(ncidades), ncidades)
    return caminhoaleatiorio


def gerarpopulacaoaleatoria(ncidades, tampop, matrizdistancia, porcenta-
gemcorte, premiominimo, premios, penalidades):
    populacao = []
    for i in range(tampop):
        solucao = solucaogulosaaleatoria(porcentagemcorte, ncidades, ma-
trizdistancia, premiominimo, premios)
        rotasolucao = rota.Rota(solucao, matrizdistancia, premios, penali-
dades)
        populacao.append(rotasolucao)
    return populacao


def split(w):
    x = w[0]
    left = []
    right = []
    for i in range(1, len(w)):
        if w[i].custo <= x.custo:
            left += [w[i]]
        else:
            right += [w[i]]
    return left, x, right


def quicksort(w):
    if len(w) < 2:
        return w
    else:
        w1, x, w2 = split(w)
        return quicksort(w1) + [x] + quicksort(w2)


def printpop(populacao):
    for i in populacao:
        printrota(i)


def printrota(rota):
    print(rota.caminho)
    print(rota.custo)
    print(rota.premio)


def caminhofilhovazio(ncidades):
    filhocaminho = []
    for i in range(ncidades):
        filhocaminho.append(False)
    return filhocaminho


def tamfatia(ncidades, porcentagemdafatia):
    return int(ncidades * porcentagemdafatia)


def cortarpai2(pai2, fimcorte, ncidades, cortepai1):
```

```
    primeiravez = 0
    cortepai2 = []
    rodar = True
    i = fimcorte
    while rodar:
        if i == fimcorte and primeiravez == 0 or i != fimcorte:

            if not pai2.caminho[i] in cortepai1:
                cortepai2.append(pai2.caminho[i])
        if i == fimcorte:
            if primeiravez == 0:
                primeiravez = 1
            else:
                rodar = False
        if i + 1 == ncidades:
            i = -1
        i += 1
    return cortepai2


def porcorte2nofilho(fimcorte, cortepai2, filhocaminho, ncidades):
    primeiravez = 0
    rodar = True
    i = fimcorte
    j = 0
    for corte in cortepai2:
        filhocaminho[i] = corte
        i += 1
        if i == ncidades:
            i = 0


def cruzamento(entradapai1, entradapai2, ncidades, distancia, premios, pe-
nalidades):
    pai1 = entradapai1
    pai2 = entradapai2
    pai1.caminho = entradapai1.caminho.copy()
    pai2.caminho = entradapai2.caminho.copy()
    diferencatam = ncidades - len(pai1.caminho)
    for i in range(diferencatam):
        pai1.caminho.append(None)
    diferencatam = ncidades - len(pai2.caminho)
    for i in range(diferencatam):
        pai2.caminho.append(None)

    # Criando caminho filho vazio
    filhocaminho = caminhofilhovazio(ncidades)

    # Escolhendo qual fatia do pai 1
    tamslice = tamfatia(ncidades, 0.4)
    iniciocorte = random.sample(range(ncidades - tamslice), 1)[
        0]  # random sample ira pegar uma casa valida para iniciar o corte.
    fimcorte = iniciocorte + tamslice

    # Corte no pai 1
    cortepai1 = pai1.caminho[iniciocorte:fimcorte]

    # Corte no pai 2
    cortepai2 = cortarpai2(pai2, fimcorte, ncidades, cortepai1)
```

```
    # Pondo no caminho filho
    filhocaminho[iniciocorte:fimcorte] = cortepai1
    porcorte2nofilho(fimcorte, cortepai2, filhocaminho, ncidades)

    # Criando filho
    filho = rota.Rota(filhocaminho, distancia, premios, penalidades)

    return filho


def getquantelite(tampop):
    quantelite = int(tampop * 0.1)
    if quantelite < 1:
        quantelite = 1
    return quantelite


def gerarfilhos(tampop, populacao, ncidades, distancia, nfilhos, premios,
penalidades, premiominimo):
    # Selecionar pais

    quantelite = getquantelite(tampop)
    elite = populacao[:quantelite]
    nelite = populacao[quantelite:]
    filhos = []
    # Cruzamentos
    for i in range(0, nfilhos):
        pai1 = elite[random.randrange(0, quantelite)]
        pai2 = nelite[random.randrange(quantelite, len(nelite))]
        filho = cruzamento(populacao[0], populacao[1], ncidades, distancia,
premios, penalidades)
        if filho.premio >= premiominimo:
            filhos.append(filho)
        else:
            i = i - 1
    return filhos


def gerarmutantes(tampop, ncidades, distancia, porcentagemcorte, premiomi-
nimo, premios, penalidades):
    nmutantes = int(tampop * 0.3) #0.3
    mutantes = []
    for i in range(0, nmutantes):
        solucao = solucaogulosaaleatoria(porcentagemcorte, ncidades, dis-
tancia, premiominimo, premios)
        mutante = rota.Rota(solucao, distancia, premios, penalidades)
        mutantes.append(mutante)
    return mutantes


# retorna o i de um individuo
def roulettewheel(possivelpopulacao):
    #
    somatotalfitness = 0
    for individuo in possivelpopulacao:
        somatotalfitness += individuo.fitness
    #
    seletor = random.uniform(0, somatotalfitness)
    roleta = 0
    i = 0
```

```
    for individuo in possivelpopulacao:
        roleta += individuo.fitness
        if roleta > seletor:
            return i
        i += 1


def sobreviver(populacao, tampop, filhos, mutantes):
    # Ver quem passa para proxima geracao:
    # Elite
    quantelite = getquantelite(tampop)
    elite = populacao[:quantelite]
    nelite = populacao[quantelite:]
    # 1) Juntar elite,melhores da pop passada e pop nova num grande vetor
    possivelpopulacao = []
    possivelpopulacao = concatenate((elite, nelite[:int(tampop / 2)]),
axis=0)
    possivelpopulacao = concatenate((possivelpopulacao, filhos), axis=0)
    # 2) Ordenar esse grande vetor e cortar solucoes ruins
    possivelpopulacao = sorted(possivelpopulacao, key=lambda rota:
rota.custo)
    possivelpopulacao = possivelpopulacao[:tampop - len(mutantes)]
    # 3) Adicionar as mutações e ordenar outra vez
    possivelpopulacao = concatenate((possivelpopulacao, mutantes), axis=0)
    possivelpopulacao = sorted(possivelpopulacao, key=lambda rota:
rota.custo)
    sobreviventes = possivelpopulacao
    return sobreviventes

    return sobreviventes


def atualizardistanciaparaatualcitycands(citycands, cidadeatual, matrizdis-
tancia):
    for city in citycands:
        city.atualizardistanciaparacidadeatual(cidadeatual, matrizdistan-
cia)
    return citycands


from cidadecandidata import CidadeCandidata


def getslicesorteavel(porcentagem, ncidades, citycands):
    porcentagemcorte = porcentagem
    slicesorteavel = int(1 + porcentagemcorte * (len(citycands) - 1))
    return slicesorteavel


def gerarlistacidadescandidatas(ncidades, cidadeinicial, matrizdistancia):
    citycands = []
    for i in range(ncidades):
        if not i == cidadeinicial:
            citycand = CidadeCandidata(i, cidadeinicial, matrizdistancia)
            citycands.append(citycand)

    # Ordenando pela distancia para a distancia atual
    citycands = sorted(citycands, key=lambda CidadeCandidata: CidadeCandi-
data.distanciaparacidadeatual)
    # printcidadescandidatas(citycands)
```

```
    return citycands


def solucaogulosaaleatoria(porcentagemcorte, ncidades, distancia, premiomi-
nimo, premios):
    # Sorteando a primeira cidade
    # cidadeinicial = random.choice(range(ncidades))
    cidadeinicial = 0
    premioatual = float(premios[0])
    # print(cidadeinicial)

    # Criando uma lista com cidades candidatas.
    citycands = gerarlistacidadescandidatas(ncidades, cidadeinicial, dis-
tancia)

    # Vetor de solução
    solucaogulosa = []
    solucaogulosa.append(cidadeinicial)
    # Fazer ate completar solucao
    while len(solucaogulosa) < ncidades and premioatual <= premiominimo:
        # Ver quantas cidades entram no sorteio
        slicesorteavel = getslicesorteavel(porcentagemcorte, ncidades,
citycands)
        citycands[:slicesorteavel]
        # Sorteio da nova cidade atual

        cidadeatual = random.choice(citycands[:slicesorteavel])
        # Adicionar cidade sorteada ao vetor de solucao e removê-lo das
candida-tas
        premioatual += float(premios[cidadeatual.numero])
        solucaogulosa.append(cidadeatual.numero)
        citycands.remove(cidadeatual)

        # Atualizar lista de distancias para cidade atual e reordenar por
essa distância
        citycands = atualizardistanciaparaatualcitycands(citycands, cidade-
atual, distancia)
        citycands = sorted(citycands, key=lambda CidadeCandidata: Cidade-
Candidata.distanciaparacidadeatual)

    return solucaogulosa


import os

class CidadeCandidata(object):
    def __init__(self, numero, cidadeatual,matrizdistancia):
        self.numero = numero
        self.distanciaparacidadeatual = self.calculacustoentrecidades(cida-
deatual, numero, matrizdistancia)

    def calculacustoentrecidades(self,cidadeatual, cidadeacomparar, ma-
trizdistancia):
        custo = float(matrizdistancia[cidadeatual][cidadeacomparar])
        return custo

    def atualizardistanciaparacidadeatual(self, cidadeatual,matrizdistan-
cia):
        self.distanciaparacidadeatual = self.calculacustoentrecidades(cida-
deatual.numero, self.numero, matrizdistancia)
```

```
import os
from numpy import *
import random


class Rota(object):

    def __init__(self, caminho, matrizdistancia, premios, penalidades):
        self.caminho = caminho
        self.custo = self.calculacusto(caminho, matrizdistancia, penalida-
des)
        self.premio = self.calculapremio(caminho, premios)
        self.penalidade = self.calculapenalidade(caminho, penalidades)

    def calculacusto(self, caminhoentradanaotratado, matrizdistancia, pena-
lidades):
        custo = 0
        caminhoentrada = []
        for espaco in caminhoentradanaotratado:
            if not espaco is None:
                caminhoentrada.append(espaco)
        # interar i vezes, sendo i o numero de elementos do caminho de en-
trada-1.
        # Ou seja, i vai receber as posicoes do caminho e caminhoentrada[i]
o valor daquela posicao
        for i in range(len(caminhoentrada) - 1):
            custo += float(matrizdistancia[caminhoentrada[i]][caminhoen-
trada[i + 1]])
        custo += float(matrizdistancia[caminhoentrada[len(caminhoentrada) -
1]][caminhoentrada[0]])
        #calcula o numero de cidades.
        ncidades = len(matrizdistancia)
        penalidade = 0
        # iterar entre 0 e ncidades-1
        for i in range(ncidades):
            if not i in caminhoentrada: #caso a posicao nao esteja no cami-
nho:
                penalidade+=float(penalidades[i])  # adicionar a penalidade
daquela posicao
        custo += penalidade # somar penalidade a custo

        return custo

    def calculapremio(self, caminhoentradanaotratado, premios):
        premio = 0
        caminhoentrada = []
        for espaco in caminhoentradanaotratado:
            if not espaco is None:
                caminhoentrada.append(espaco)
        for index in caminhoentrada:
            premio += float(premios[index])
        return premio

    def calculapenalidade(self, caminhoentradanaotratado, penalidades):
        penalidade = 0
        caminhoentrada = []
        for espaco in caminhoentradanaotratado:
            if not espaco is None:
                caminhoentrada.append(espaco)
        for index in caminhoentrada:
```

```
            penalidade += float(penalidades[index])
        return penalidade

    def intersecao (self, lista1, lista2):
        lista3 = []
        for n in lista1:
            if not n in lista2:
                lista3.append(n)
        return lista3

from tools import *
import time

from rota import Rota

def main():
    # Lendo arquivos e gerando matriz de distancias
    stringsarquivo = lerarquivo('PCP_CP40d.csv') #parametro ajustável de
acordo com a instancia utilizada
    ncidades = len(stringsarquivo)
    distancia = criarmatrizdistancia(stringsarquivo, ncidades)
    stringsarquivo = lerarquivo('PCP_CP40pr.csv') #parametro ajustável de
acordo com a instancia utilizada
    premios = criarvetorvalores(stringsarquivo, ncidades)
    stringsarquivo = lerarquivo('PCP_CP40p.csv') #parametro ajustável de
acordo com a instancia utilizada
    penalidades = criarvetorvalores(stringsarquivo, ncidades)

    inicio = time.time()

    # Gerando populacao inicial
    tampop = 100
    porcentagemcorte = 0.5
    premiominimo = 56919 #parametro do prêmio minímo ajustável de acordo
com a instancia utilizada
    populacao = gerarpopulacaoaleatoria(ncidades, tampop, distancia, por-
centagemcorte, premiominimo, premios,
                                        penalidades)
    populacao = sorted(populacao, key=lambda rota: rota.custo)  # Ordenacao
da populacao

    # Fazendo as geracoes
    melhor = None
    ngeracoes = 1000
    for i in range(0, ngeracoes):
        print('it:' + str(i))
        # Cruzamentos
        nfilhos = tampop
        filhos = gerarfilhos(tampop, populacao, ncidades, distancia, nfi-
lhos, premios, penalidades, premiominimo)
        filhos = sorted(filhos, key=lambda rota: rota.custo)
        # Mutações
        mutantes = gerarmutantes(tampop, ncidades, distancia, porcenta-
gemcorte, premiominimo, premios, penalidades)
        mutantes = sorted(mutantes, key=lambda rota: rota.custo)

        # Ver quem passa para proxima geracao:
        populacao = sobreviver(populacao, tampop, filhos, mutantes)
        populacao = sorted(populacao, key=lambda rota: rota.custo)
```

```
        # Salvando e exibindo melhor
        if melhor is None:
            melhor = populacao[0]

            print('Melhor custo: ' + str(melhor.custo))
        else:
            if populacao[0].custo < melhor.custo:
                melhor = populacao[0]

                print('Melhor custo: ' + str(melhor.custo))
                print('Caminho: ' + str(melhor.caminho))
    # Tempo final para execucao
    fim = time.time()
    final = fim - inicio

    print('Melhor solucao encontrada: ' + str(melhor.custo))
    print('Melhor solucao encontrada[caminho]: ' + str(melhor.caminho))
    print('Premio encontrado[caminho]: ' + str(melhor.premio))

    caminhoentrada = [x for x in range(0,21)]
    penalidadeTest = 0
    caminhoentrada = Rota.intersecao('__main__', caminhoentrada, melhor.ca-
minho)
    for index in caminhoentrada:
        penalidadeTest += float(penalidades[index])
    print('penalidades encontrado[caminho]: ' +str(penalidadeTest))
    print('Custo da Rota: ' +str(melhor.custo - penalidadeTest))


    print('Tempo total de execução: ' +str(final))


if __name__ == '__main__':
    main()
```