UNIVERSIDADE ESTADUAL PAULISTA

FACULDADE DE CIÊNCIAS

CAMPUS DE BAURU

MATEUS GOMES CABANA

META-HEURÍSTICAS HÍBRIDAS PARA PROBLEMAS DE ROTEAMENTO

BAURU

2018

MATEUS GOMES CABANA

META-HEURÍSTICAS HÍBRIDAS PARA PROBLEMAS DE ROTEAMENTO

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Estadual Paulista "Julio de Mesquita Filho", Campus Bauru, como requisito para a obtenção do grau de Cientista da Computação.
Orientadora: Prof[a] Dr[a] Andréa Carla Gonçalves Vianna

BAURU

2018

Mateus Gomes Cabana. Meta-heurísticasHíbridas para Problema de Roteamento./Mateus Gomes Cabana – Bauru, Novembro/2018 - 62.p.: il.(algumas color); 30 cm.

Orientadora: Profª Drª Andrea Carla Gonçalves Vianna

Trabalho de Conclusão de Curso – Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Ciência da Computação, Novembro /2018

1. Tags 2. Para 3. A 4. Ficha 5. Catalográfica

Dedico esta, bem como todas as minhas demais conquistas, aos meus amados pais e amigos que ajudaram nessa caminhada.

AGRADECIMENTOS

Quero agradecer, em primeiro lugar, a Deus, pela força e coragem durante toda esta longa caminhada. A Professora Doutora Andréa Carla Gonçalves Vianna, com quem partilhei o que era o broto daquilo que veio a ser esse trabalho. Nossas conversas durante estudos foram fundamentais. E a minha família, por todo apoio que me deram.

RESUMO

Problemas que envolvem a roteirizarão de veículos aparecem frequentemente na área de logística. Um problema clássico é o Problema do Caixeiro Viajante (PCV) que determina a melhor rota para percorrer uma sequência de cidades, visitando todas elas uma única vez, e retornando à cidade inicial. Por se tratar de um problema de otimização combinatória, esse trabalho propõe estratégias de solução denominadas meta-heurísticas. Dessa forma, diferentes técnicas foram estudadas e implementadas, entre elas Algoritmos Genéticos, Busca Tabu e VNS. Apesar disso, as meta-heurísticas podem apresentar alguns problemas, que podem ser facilmente resolvidas através de uma Rede Neural Artificial, que são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural. Assim, através desta junção, é possível criar um sistema hibrido para corrigir eventuais problemas das meta-heurísticas. Analises foram feitas para estudar vantagens e desvantagens de cada método, bem como um estudo aprofundado com o sistema híbrido.

Palavras-Chaves: Otimização; Caixeiro Viajante; Meta-heurísticas; Rede Neural.

ABSTRACT

Problems involving vehicle routing often appear in the logistics area. A classic problem is the Traveling Salesman Problem (TSP) that determines the best route to travel through a sequence of cities, visiting all of them once, and returning to the starting city. Due to the fact that it is a combinatorial optimization problem, this work proposes solution strategies called metaheuristics. In this way, different techniques were studied and implemented, among them Genetic Algorithms, Tabu Search and Neighborhood Search. Nevertheless, meta-heuristics can present some problems, which can be easily solved through a Artificial Neural Network that are computational techniques that present a mathematical model inspired by the neural structure. Only then, through this join, will it be possible to create a hybrid system to correct any meta-heuristic problems. Analyzes were made to study the advantages and disadvantages of each method, as well as an in-depth study with the hybrid system.

Keywords: Optimization; Traveling Salesman Problem; Metaheuristics; Artificial Neural Network.

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Problemas envolvendo roteirização de veículos ocorrem frequentemente na área de logística, principalmente na minimização dos custos de uma rota e, na determinação do menor percurso com objetivo de minimizar a distância total percorrida. Um problema clássico de roteirização é o Problema do Caixeiro Viajante (*Traveling Salesman Problem*) (Cunha, 2000) que consiste em determinar a sequência de cidades (ou clientes) a serem visitadas por um caixeiro viajante que minimize a distância total percorrida e garanta que cada cidade (ou cliente) seja visitada exatamente uma vez.

Por ser um dos problemas com diversas aplicações, por se tratar de uma otimização combinatória, tornando-se possível encontrar muitas questões apesar da simplicidade de sua formulação, pertence à classe NP-completo. Desta forma, tem-se buscado estratégias de solução exatas e heurísticas para o problema.

As meta-heurísticas vem suprir esta deficiência e tem como objetivo principal explorar um espaço de pesquisa de forma inteligente, ou seja, encontrar soluções de alta qualidade movendo-se para áreas não exploradas quando necessário. Exemplos disso são os Algoritmos Genéticos(AG) (Goldberg, 1989) que utilizam as ideias evolutivas de Darwin para encontrar a solução (Silva e Oliveira, 2006), a Busca por Vizinhança (Mladenovic e Hansen, 1997) que realiza buscas em regiões de solução factível (Carvalho *et al*, 2008), Busca Tabu (Glover, 1996) que é uma variação da busca por vizinhança, porém possui uma memória adaptativa, entre outras (Pedro, 2013).

Assim, meta-heurísticas são procedimentos de busca, com capacidade de escapar de ótimos locais, focando na eficiência e uma maior exploração do espaço de busca, destinados a resolver problemas de otimização.

Apesar disso, as meta-heurísticas podem apresentar alguns problemas, que podem ser, muitas vezes, resolvidos utilizando uma rede neural. As Redes Neurais Artificiais (RNA) são técnicas computacionais que apresentam um modelo matemático inspirados no modelo do sistema nervoso e neurônios

biológicos, que irão auxiliar na conversão através de seu comportamento “inteligente”, que vem das interações entre as unidades de processamento da rede (Haykin, 2001). Só assim, através desta junção, será possível criar um sistema híbrido inteligente que busca unir duas ou mais técnicas a fim de obter melhorias nos resultados finais (Melo e Martinhon, 2004). Por isso, são muito relevantes nos dias atuais, visto que apresentam problemas, os quais requerem diversos tipos de processamento para atingir sua finalidade.

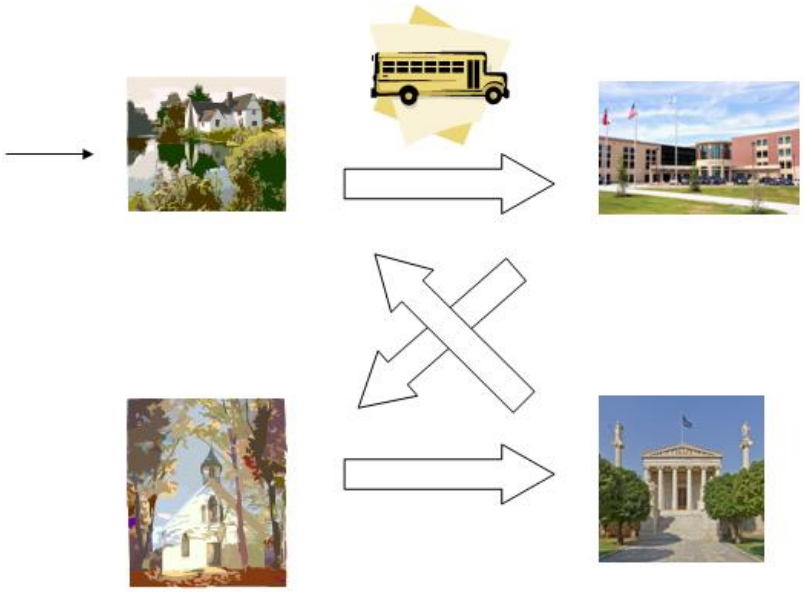
Este projeto tem como objetivo implementar computacionalmente algoritmos de meta-heurísticas e aplicá-los na resolução do Problema do Caixeiro Viajante, e estudar vantagens e desvantagens para este tipo de problema. Além disso será utilizado uma RNA para corrigir eventuais problemas da meta-heurísticas, criando assim um sistema híbrido.

No capítulo 2 é feito um levantamento bibliográfico sobre o assunto. Nos capítulos 3 e 4 são descrito como as meta-heurísticas foram aplicadas ao problema de roteamento, bem como a RNA. O desenvolvimento de todo projeto e resultados computacionais estão nos capítulos 6 e 7, respectivamente.

2 REVISÃO BIBLIOGRÁFICA

O Problema do Caixeiro Viajante (*Traveling Salesman Problem*) consiste em determinar a sequência de cidades (ou clientes) a serem visitadas por um caixeiro viajante tal que minimize a distância total percorrida e garanta que cada cidade (ou cliente) seja visitada exatamente uma vez. Apesar de ser um problema muito antigo, ainda consegue ser de grande relevância, tanto no campo das ciências, atraindo centenas de pesquisadores, não somente aqueles da área de otimização, mas de diversos campos: biologia, física, inteligência artificial, entre outros. A Figura 1 apresenta, de forma bastante simples, um modelo de problema do caixeiro viajante.

Figura 1 – Exemplo de um Problema do Caixeiro Viajante.



Fonte: Elaborada pelo Autor.

Por se tratar de um problema que envolve questões combinatórias (Laporte, 1992), o algoritmo do caixeiro viajante é classificado como NP-

completo. O grande desafio é determinar o melhor método a ser aplicado a este tipo problema, visto que, por ser um problema dessa natureza, a quantidade de cidades envolvida no problema afeta diretamente o desempenho e resultado final obtido.

Para determinar sua solução, é possível utilizar estratégias que busquem uma solução próxima ao ótimo, denominada heurísticas.

Plous (1993) conceitua as heurísticas como regras gerais de influência utilizadas pelos sujeitos para chegar aos seus julgamentos em tarefas decisórias de incerteza, as heurísticas também podem utilizadas como um métodos de otimização que tiram proveito de características e informações do próprio problema a ser explorado, facilitando o encontro de um ótimo global no espaço de busca. As heurísticas são limitadas e fornecem sempre a mesma solução quando iniciadas de um mesmo ponto de partida.

As meta-heurísticas vem para suprir esta deficiência tem como objetivo principal explorar um espaço de pesquisa de forma inteligente, ou seja, encontrar soluções de alta qualidade movendo-se para áreas não exploradas quando necessário.

Toda tarefa de busca e otimização possui componentes importantes, entre eles: o espaço de busca, onde são consideradas todas as possibilidades de solução de um determinado problema e a função de avaliação (ou função de custo), uma maneira de avaliar os membros do espaço de busca. Existem diferentes métodos de busca e funções de avaliação que foram objeto de estudo neste trabalho. Estes algoritmos foram adaptados ao Problema do Caixeiro Viajante e implementados computacionalmente, dentre eles pode-se destacar o Algoritmo Genético, Busca em Vizinhança e Busca tabu.

Os AG pertencem a uma classe particular de algoritmos evolutivos que usam técnicas inspiradas pela biologia dentre elas a evolução natural de Darwin (Goldberg, 1989) para encontrar a solução (Silva e Oliveira, 2006). Segundo a teoria evolutiva de Charles Darwin, os indivíduos que apresentam características mais aptas para o ambiente têm mais chance de sobreviver e

deixar seus herdeiros. Assim, pelo decorrer das gerações, a população de uma espécie evolui para melhor adaptar ao ambiente onde reside.

O método consiste em solucionar problemas de forma a encontrar o resultado quase ótimo, sendo aplicável em diversas áreas. Para entender o funcionamento faz-se necessário entender as analogias dos processos evolutivos de Darwin (Figura 2). Assim, de acordo com (Goldberg,1989) pode-se dizer que:

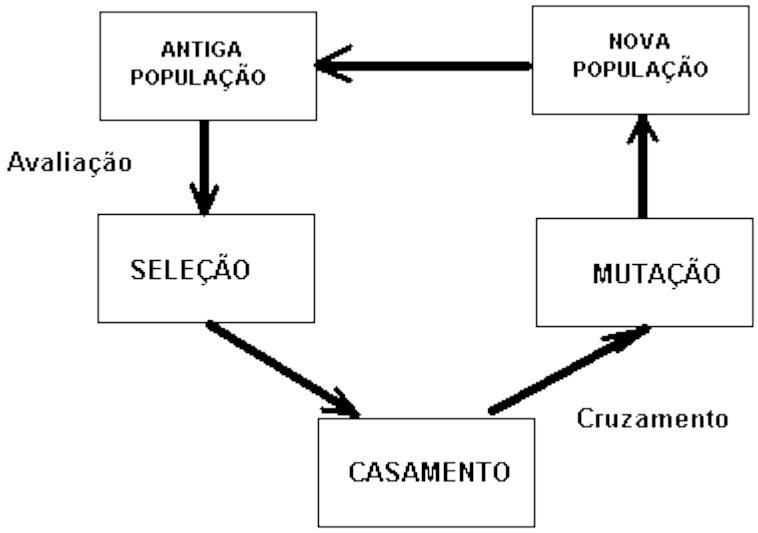
- Inicialmente é gerada uma população com um conjunto finito de indivíduos, em que cada indivíduo é uma solução viável ao problema.

- A cada iteração, ocorrem processos evolutivos, como *crossover* entre os indivíduos e mutação em um individuo existente na população, gerando assim novos “filhos”. Esta população é avaliada, ou seja, cada indivíduo recebe uma nota.

- O princípio básico do funcionamento dos AG é que um critério de seleção vai fazer com que, depois de muitas gerações, neste projeto optou-se escolher pelo método do elitismo em que uma porcentagem dos indivíduos mais adaptados é mantida, enquanto os outros são descartados.

- Os membros que foram mantidos sofrem novamente os processos evolutivos. Este processo reprodutivo é repetido até que a solução seja satisfatória.

Figura 2 - Algoritmos Genéticos.



Fonte:<http://www.lem.ep.usp.br/Pef411/~Cristiano%20Oliveira/CristianoOliveira/Paginas/Fundamentacao.htm> Acesso em 29 de Outubro de 2018

Busca em Vizinhança Variável foi proposta (Mladenović e Hansen, 1997) em meados da década de 90, é uma meta-heurística que se baseia na ideia de "mudanças sistemáticas" da vizinhança a qual a solução atual se encontra, nesse aspecto essa meta-heurística é diferente das demais, com isso O algoritmo muda a vizinhança como uma forma de sair de soluções ótimas locais. Nesse processo a solução corrente também é a incumbente o que é o outro ponto que se difere dos demais métodos.

A Busca Tabu é uma meta-heurística que tem por finalidade encontrar uma solução através de uma busca heurística local como semelhante a Busca pro vizinhança, o diferencial desse método e que o mesmo guarda informações sobre soluções já visitadas, buscando fugir de mínimos locais (Pedro, 2013). A seção 3.3 apresentará o funcionamento dessa meta-heurística aplicado ao problema do caixeiro viajante

Apesar de encontrarem ótimos resultados, as meta-heurísticas podem apresentar alguns problemas, e isto pode ser resolvido através de uma rede neural Artificial. As Redes Neurais Artificiais (RNA) são técnicas

computacionais que apresentam um modelo matemático inspirado no modelo do sistema nervoso e neurônios biológicos. Assim, é possível criar um sistema híbrido utilizando uma RNA e uma meta-heurística.

Os sistemas híbridos podem ser caracterizados como uma combinação de sub-sistemas, em que cada um possui uma identidade individual e não apresentem características não-coincidentes entre si (GHOSH,1997). Essa combinação deve originar um método que possua um potencial de solucionar o problema maior do que a soma dos potencias de seus sub-sistemas isolados. Uma das grandes motivações de se implantar um sistema híbrido é que não existe um único método que seja ótimo para qualquer problema. Além disso, deixar um único método dedicado à solução pode ser um processo mais custoso do que a obtenção de uma solução pela combinação de múltiplos métodos como serão apresentado na sessão de resultados computacionais.

3 META-HEURISTICAS UTILIZADAS

3.1 ALGORITMOS GENÉTICOS

AG são técnicas inspiradas na seleção e evolução natural de Darwin, como também outras teorias evolutivas, tais como teorias genéticas de Gregor Mendel (Goldberg, 1989).

Os AG diferem muito dos algoritmos tradicionais em vários aspectos, entre eles:

- Trabalha com a codificação com um conjunto de soluções possíveis.

- O resultado é apresentado como uma população e não como uma solução única.

- Usam transições probabilísticas e não determinística, entre outras.

Para compreender o funcionamento dos AG para o Problema do Caixeiro Viajante nesse projeto faz-se necessário realizar uma analogia à explicação sobre a evolução das espécies. Assim, para o Problema do Caixeiro Viajante tem-se:

1- Inicialmente é gerada uma população formada por um conjunto aleatório de indivíduos (possíveis rotas), que podem ser vistos como possíveis soluções para o problema.

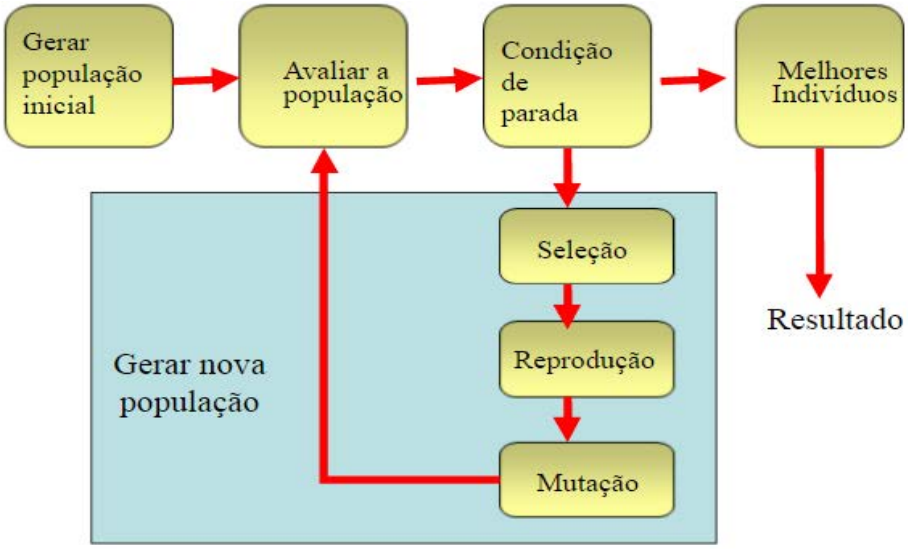
2- Durante o processo evolutivo, esta população é avaliada, sendo que para cada indivíduo (rota) é atribuída uma nota (distancia total do percurso), ou índice, que reflete sua habilidade de adaptação a determinado ambiente. Como o objetivo é minimização, as maiores notas tendem sempre a sair.

3- A cada iteração os membros mantidos pela seleção do elitismo podem sofrer modificações em suas características fundamentais por meio de cruzamentos (*crossover*) e mutações, gerando descendentes para a próxima geração, que serão inseridos se sua pontuação for menor do que a maior pontuação da população.

4- Este processo, chamado de reprodução, é repetido até que uma solução satisfatória seja encontrada, ou o número de iterações atinja o máximo estabelecido. Embora possam parecer simplistas do ponto de vista biológico, estes algoritmos são suficientemente complexos para fornecer mecanismos de busca adaptativos poderosos e robustos.

Alguns parâmetros precisam ser estabelecidos para que o algoritmo seja executado corretamente, entre os quais pode-se citar o tamanho da população (deve ser proporcional ao número de cidades existentes), a taxa de cruzamento utilizado no processo inteiro, sendo que a cada iteração é gerado valor,se o valor gerado for menor ou igual ao valor da taxa pré estabelecido é feito o cruzamento, a taxa de mutação que segue o mesmo principio da taxa de cruzamento, porém, o processo de geração de novos filhos é diferente, e o critério de parada que foi estabelecido é um número máximo de passo. A Figura 3 ilustra o funcionamento dos Algoritmos Genéricos.

Figura 3 - Funcionamento dos Algoritmos Genéticos.



Fonte: Elaborado pelo Autor.

3.2 BUSCA EM VIZINHANÇA VARIÁVEL

O algoritmo de Busca em Vizinhança Variável (VNS) (*Variable Neighbourhood Search*) é uma meta-heurística que se baseia na ideia de "mudanças sistemáticas" da vizinhança a qual a solução atual se encontra (Mladenović e Hansen, 1997). O algoritmo VNS muda a vizinhança sempre como uma forma de sair da solução ótima local e alcançar o ótimo global.

Contrariamente a outras meta-heurísticas, baseadas em métodos de busca local, o método VNS não segue uma trajetória, mas sim explora vizinhanças gradativamente mais "distantes" da solução corrente. O método focaliza a busca em torno de uma nova solução se e somente se um movimento de melhora é realizado. Assim, essa meta-heurística realiza um conjunto de trocas no espaço de busca até que se encontre o ótimo local, para que então ele possa realizar a mudança de vizinhança para "fugir" desse ótimo local. Com isso, ao encontrar o ótimo global então a busca fica parada sem possibilidade de sair desse ponto. Assim, nesse estudo, foi implementado um

algoritmo de força bruta (consiste em enumerar todos os possíveis candidatos da solução e checar cada candidato para saber se ele satisfaz o enunciado do problema).

O algoritmo Busca em Vizinhança consiste nos seguintes passos:

1 Inicialmente são criadas soluções (rotas) de forma aleatória.

2 Seleciona-se aleatoriamente um vizinho *s'* dentro da vizinhança N(k)(*s*) da solução *s* corrente.

3 Esse vizinho é então submetido a um procedimento de busca local (trocas sistemáticas). Se a nova solução ótima local, *s"*, for melhor que a solução *s* corrente, a busca continua de *s"* recomeçando da primeira estrutura de vizinhança N(1)(*s*). Caso contrário, continua-se a busca a partir da próxima estrutura de vizinhança N(*k*+1)(*s*).

4 Este procedimento é encerrado quando uma condição de parada for atingida, no caso, o número máximo de iterações.

3.3 BUSCA TABU

A Busca Tabu é uma meta-heurística que tem por finalidade encontrar um melhor resultado através de uma busca heurística local. Por isso é muito semelhante ao VNS. A grande diferença é que o método guarda informações sobre soluções já visitadas, buscando fugir de mínimos locais, onde essa memória (lista) adaptativa vai gerar uma estratégia de busca. A memória adaptativa armazena os melhores elementos encontrados na busca, podendo ser memória de curta ou longa duração.

A memória de curta duração armazena as melhores soluções por um tempo pequeno. Assim que, novas soluções surgem e são melhores que as armazenadas anteriormente, a solução armazenada é retirada da memória

cedendo local para a nova solução, como exemplificado na Figura 4. Caso a solução corrente caia em um ponto baixo, qualquer movimentação melhorará o resultado fazendo com que esse elemento possa ser excluído da lista, visto que há um resultado que melhora a solução.

Figura 4 - Gráfico que representam a possível solução do problema.

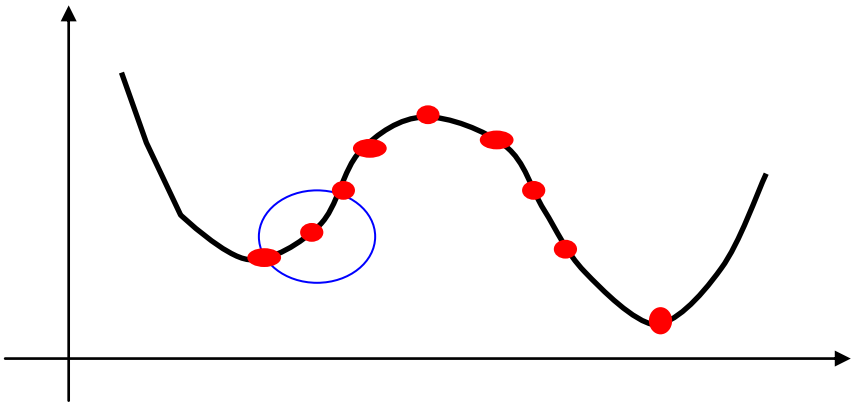
Fonte: Disponível em
<http://www.inf.ufpr.br/aurora/disciplinas/topicosia2/downloads/trabalhos/BuscaTabuTSP.pdf>
Acessado em 1 de outubro de 2018

A memória de longa duração, por ser mais longa, armazena um número de soluções maiores que a memória de curta duração. Por isso, as soluções ficam um tempo maior alocadas na lista, que são fundamentais para que o resultado não fique estagnado na planície como mostrado na Figura 4. Essas planícies representam que o valor da função objetivo não varia com a mudança de configuração dos atributos.

Além disso, a lista tabu evita que o programa execute movimentos proibidos, já que o mesmo cria uma lista com os movimentos possíveis. Por outro lado pode proibir que movimentos para soluções que ainda não foram visitadas. Assim ela evita riscos de ciclagem, em que a solução fica presa em um ótimo local.

As estratégias de busca baseadas em memória visam explorar as boas soluções e as regiões promissoras. Em cada iteração do algoritmo é gerada uma vizinhança de tamanho N, que podem conter qualquer solução válida, menos as soluções que estão presentes na Lista Tabu. Desta vizinhança é escolhida a melhor solução. Desse modo, escolhe-se sempre as melhores soluções para serem armazenadas, enquanto as outras podem ser descartadas, evitando assim que futuramente o programa escolha soluções de baixa qualidade**.** A Figura 5 exemplifica a importância de criar a memória, para se obter a melhor solução possível.

Figura 5 - Exemplo do funcionamento da memória do algoritmo Busca Tabu, os quais os pontos indicam o que foi salvo na memória para chegar a melhor solução.



Fonte:< http://www.decom.ufop.br/marcone/Disciplinas/InteligenciaComputacional/Ico-BT.ppt> Acesso em 1 de Outubro 2018

Com base nessas ideias e nas informações dadas pelo problema, o este projeto formulou o programa de Busca Tabu para o Problema do Caixeiro Viajante que funciona da seguinte forma:

1- Inicialmente são criadas soluções (rotas) de forma aleatória.

2- A partir das cidades da solução inicial, explora-se a vizinhança dessas cidades como no Passo 2 da VNS. Se a solução for melhor que a corrente, essa vizinhança é salva na Lista Tabu.

3- Se a solução ficar presa em uma planície, à memória (lista), aumenta seu tamanho para guardar mais soluções, com o intuito de fugir da mesma. Caso isso não acontecer o programa fica com uma memória (lista) curta e de mesmo tamanho.

OBS: Pela possibilidade de utilizar muita memória, no programa computacional utilizou-se lista com alocação dinâmica, apesar de que a memória dinâmica se restringir ao tamanho da memória total.

4- A cada nova iteração é sempre consultado a lista tabu e, se possível, realiza a movimentação, senão for possível volta ao Passo 2.

5- Este procedimento é encerrado quando uma condição de parada for atingida, no caso, o número máximo de iterações.

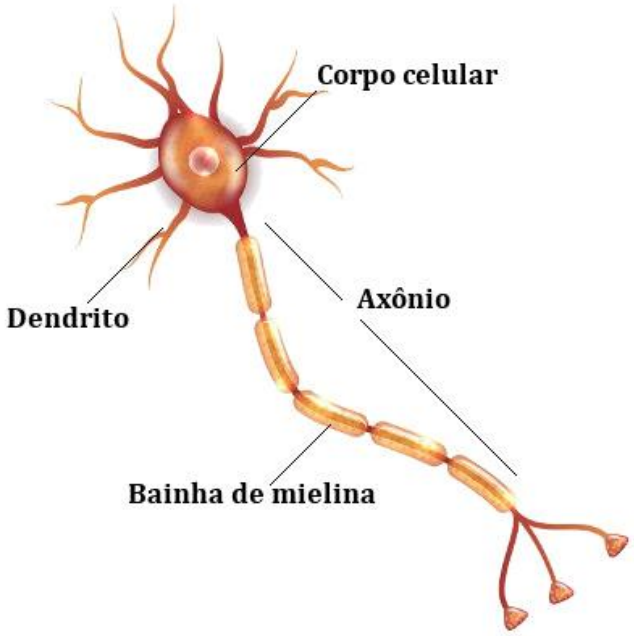
4 REDE NEURAL ARTIFICIAL

As RNA inspiram-se em modelos biológicos, ou seja, são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural (Haykin, 2001) que é capaz de adquirir conhecimento.

Os neurônios se comunicam através de sinapses. A sinapse é uma região onde dois neurônios entram em contato e são transmitidos os impulsos nervosos entre eles. Essas sinapses podem ser excitatórias ou inibitórias. As sinapses excitatórias enviam sinais a outros neurônios que recebem o estímulo e tendem a também ficar excitados e a passar o estímulo adiante favorecendo disparo do neurônio enquanto que as sinapses inibitórias inibem o disparo dos neurônios.

Os impulsos recebidos por um neurônio, em um determinado momento, são processados e assim, esse neurônio produz uma substância neurotransmissora que flui do corpo celular para o axônio, sendo enviado finalmente a outro neurônio ligados a ele pelo axônio. A Figura 6 demonstra a estrutura básica de um neurônio.

Figura 6 – Composição de um neurônio.



Fonte: <https://brasilescola.uol.com.br/o-que-e/biologia/o-que-e-neuronio.html> Acessado em 1 de outubro de 2018

Esses impulsos são uma espécie de mensagem que é transmitido a todas as células do sistema até que se chega ao cérebro, e ele interprete a mensagem e execute determinada função no organismos. A sessão 4.1 apresenta o funcionamento da RNA desenvolvida utilizada para criar um sistema híbrido.

Levando em conta esse funcionamento dos modelos biológicos explicado na sessão levantamento bibliográfico, as RNA são um tipo de sistema composto por unidades de processamento simples que possuem uma capacidade de armazenar e utilizar conhecimento (Haykin, 2001). O comportamento inteligente de uma RNA vem das interações entre as unidades de processamento da RNA. As operações mínimas que uma rede deve executar de acordo com McCulloch e Pitts (1943) podem ser resumidas como segue:

1- Os neurônios recebem dados de entrada (chamados de *neurônio de entrada*), que correspondem aos neurônios dos órgãos dos sentidos, em um sistema. Essas entradas simulam a captação de estímulos que estão conectadas a outros neurônios.

2- Cada entrada é multiplicado por um número, ou peso, que indica a sua influência na saída da unidade como pode ser representado na equação 1. Se o valor dessa função ultrapassar um valor limite estabelecido pela função de transferência, um sinal é propagado pela saída.

$$u = \sum_{i=0}^{n} W_i X_i \qquad (1)$$

cujo X_i representa sinais de entrada.

W_i representa sinapses.

u representa sinal de saída.

Algumas função de transferências podem ser estabelecidas por:

1- Linear:

$$f(x) = \alpha x$$

2- Degrau:

$$f(x) = \begin{cases} \beta & x \geq \omega \\ \alpha & x < \omega \end{cases}$$

3- Gaussiana

$$f(x) = e^{\frac{-x^2}{v}}$$

4- Sigmóide

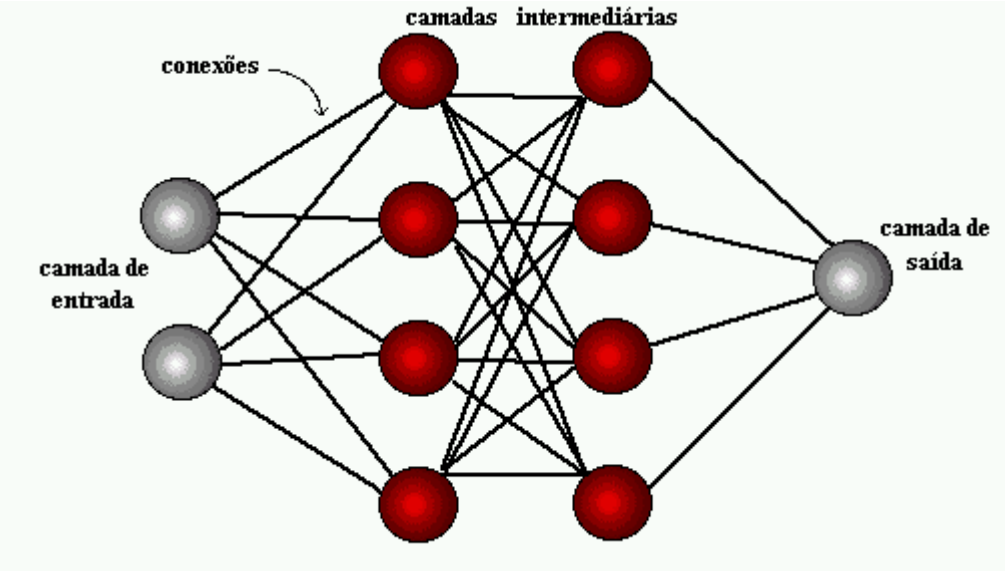
$$f(x) = \frac{1}{1 + e^{-ax}}$$

5- Sinal

$$f(x) = \begin{cases} 1\,, x \geq 0 \\ -1\,, x < 0 \end{cases}$$

Esse processamento geralmente envolve camadas em um sistema, que podem ser classificadas em três grupos representadas pela Figura 7.

Figura 7 – Estrutura básica de uma RNA.



Fonte:< http://conteudo.icmc.usp.br/pessoas/andre/research/neural/> Acessado 1 de outubro de 2018

- Camadas de entrada: São onde as entradas ou padrões da rede são inseridos.

- Camadas intermediárias (*Hidden*): Essa camada é uma das partes mais importantes, pois é nela que se realiza a maior parte do processamento, podendo ser considerada como extratora de características.

- Camadas de saída: Camada na qual a saída é gerada e apresentada, as quais podem ser comparadas aos neurônios que excitam os músculos (moto- neurônios).

Uma RNA pode ser montada com várias camadas de acordo com o problema a ser resolvido,porém em cada arquitetura deve determinar o número de camadas bem como o número de neurônios que as formam dependendo do problema a ser resolvido.

Outro ponto importante das RNA é em relação à forma como essa rede vai aprender: no processo de aprendizado ocorre mudanças significativas nas sinapses (saídas dos neurônios de cada camada). Se determinadas conexões entre os neurônios são mais usadas, elas são mais reforçadas enquanto as demais são enfraquecidas. Por causa desse fator, é necessário que a rede implementada seja treinada para que as devidas conexões fiquem bem definidas. Com base nisso existem basicamente três tipos de aprendizado:

- Supervisionado: neste tipo de aprendizado são apresentados à rede os padrões de entrada e saída. O treinamento supervisionado consiste em verificar se a saída obtida da rede foi a mesma da desejada ou próxima o suficiente para que possa ser aceita por um erro mínimo. Se o erro for maior que o aceitável, um ajuste de pesos é feito na rede.

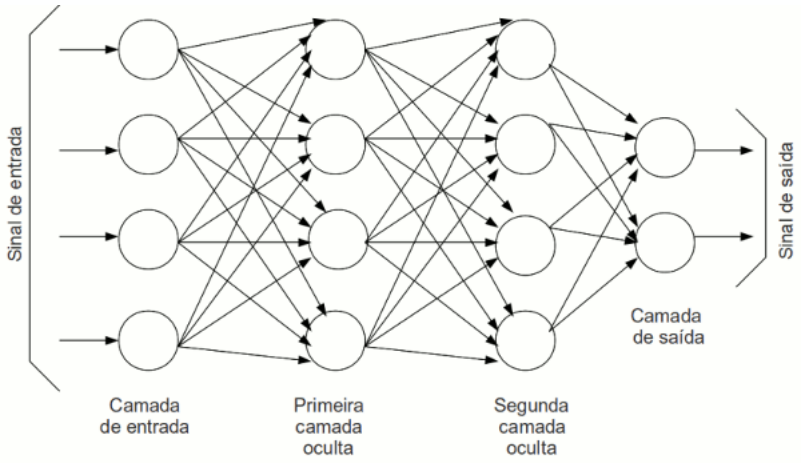
- Não-supervisionado: não existe a indicação da resposta desejada para o padrão da entrada, e sim a indicação de quão semelhante. Assim, a rede trabalha os dados de forma a determinar algumas propriedades do conjunto de dados. A partir destas propriedades é que o aprendizado é constituído.

- Híbrido: neste tipo ocorre uma "mistura" dos tipos supervisionado e não-supervisionado, sendo que uma camada trabalha com um tipo de aprendizado enquanto outra realiza o outro aprendizado.

Em relação às conexões entre as camadas a RNA pode apresentar dois comportamentos:

- Redes diretas (*feedforward*): Esses tipos de conexões geralmente não possuem nenhuma forma de ciclo. Geralmente apresentam as camadas de entrada, camadas internas (*hidden*) e as camadas de saída. A Figura 8 apresenta o esquema deste tipo de conexão.

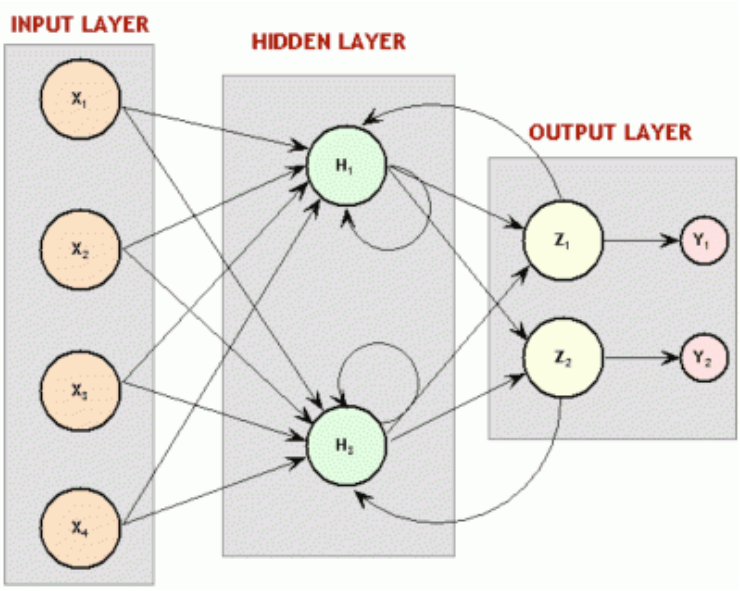
Figura 8 - Esquema de uma RNA direta.



Fonte: https://www.monolitonimbus.com.br/redes-neurais-artificiais/ Acesso em 1 de outubro de 2018

- Redes com *feedback*: São redes que possuem ciclos em suas conexões, como apresentado na Figura 9. Nesse modelo existem as conexões simétricas que é um caso particular em que possuem uma matriz de conectividade simétrica.

Figura 9 - Modelo de uma RNA com *Feedback*.



Fonte: http://deeplearningbook.com.br/as-10-principais-arquiteturas-de-redes-neurais/Acesso 16 de Novembro de 2018

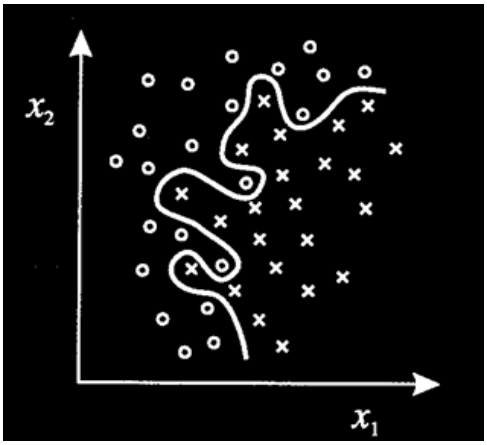
4.1 REDE NEURAL *BACKPROPAGATION*

A rede *backpropagation,* ou retropopagação, é uma RNA que possui conexões do tipo *feedforward* ou direta e possui uma arquitetura multicamada com treinamento supervisionado.

Este tipo de rede tem por objetivo melhorar os sinais das sinapses entre as camadas em que os pesos da rede são ajustados baseados no erro (Rumelhart *et al.*, 1986). O sinal de erro é propagado em sentido oposto ao de propagação do sinal funcional, reajustando os pesos entre as camadas.

A *backpropagatrion* é uma rede muito utilizada devido a suas características de trabalhar com redes multi-camadas e resolver problemas não-linearmente separáveis como demonstrado na Figura 10.

Figura 10 - Modelo de uma RNA com *Feedback*.



Fonte: https://www.devmedia.com.br/redes-neurais-artificiais-algoritmo-backpropagation/28559 Acesso 1 de Outubro de 2018

A ideia do algoritmo é formada basicamente por dois passos: o processo direto, expresso no capítulo 4, as quais uma entrada tem seu efeito aplicado em toda rede e os pesos não são alterados, e no ajuste de pesos no processo reverso, no qual o erro calculado na saída da rede é propagado no sentido reverso e com isso os pesos são reajustados de acordo com esse erro, que pode ser expresso pela equação 2:

Fórmula geral de atualização dos pesos da RNA:

$$w = w - n\frac{\partial E}{\partial w} \qquad (2)$$

em que n representa a taxa de aprendizado da RNA;

w representa o peso da iteração;

$\frac{\partial E}{\partial w}$ representam derivadas parciais da função de erro E em relação a cada peso do vetor w.

Considere a RNA representado pela Figura 11. Sendo y a saída esperada e ŷ a saída, definimos o erro como:

$$E(y,\hat{y}) = \sum_{i=1}^{n}(\boldsymbol{y_i} - \hat{y}_{\boldsymbol{i}})^{\mathbf{2}} \tag{3}$$

Agora, considere a derivada parcial do erro em relação a camada 2:

$$\frac{\partial E}{\partial \hat{y}_{\boldsymbol{k}}} = \frac{\partial}{\partial \hat{y}_{\boldsymbol{k}}}(\sum_{i=1}^{n}(\boldsymbol{y_i} - \hat{y}_{\boldsymbol{i}})^{\mathbf{2}}) = \frac{\partial}{\partial \hat{y}_{\boldsymbol{k}}}(\boldsymbol{y_i} - \hat{y}_{\boldsymbol{i}})^{\mathbf{2}} = -\mathbf{2}(\boldsymbol{y_i} - \hat{y}_{\boldsymbol{i}}) \tag{4}$$

Seguindo a estrutura da rede para a esquerda, continua-se realizando o calculo da derivada do erro em relação a z da camada 2, realizando a Regra da Cadeia.

$$\frac{\partial E}{\partial \boldsymbol{z_i^{(2)}}} = \frac{\partial E}{\partial \hat{y}_{\boldsymbol{i}}}\frac{\partial \hat{y}_{\boldsymbol{i}}}{\partial \boldsymbol{z_i^{(2)}}} = -\mathbf{2}(\boldsymbol{y_i} - \hat{y}_{\boldsymbol{i}})\sigma' z_{\boldsymbol{i}}^{(\mathbf{2})} \tag{5}$$

Sendo σ′ é a derivada da função de ativação, este valor é o gradiente local em relação ao *i*-ésimo neurônio da camada 2 e, sendo assim indica-se com δ:

$$\frac{\partial E}{\partial \boldsymbol{z_i^{(2)}}} = \delta_{\mathrm{i}}^{(2)} \tag{6}$$

Aplicando a Regra da Cadeia realiza-se a derivada parcial do erro em função de um peso w[j,i] da camada 2:

$$\frac{\partial E}{\partial \boldsymbol{w_{j,i}^{(2)}}} = \frac{\partial E}{\partial \boldsymbol{z_i^{(2)}}}\frac{\partial \boldsymbol{z_i^{(2)}}}{\partial \boldsymbol{w_{j,i}^{(2)}}} = \delta_{\mathrm{i}}^{(2)}\frac{\partial}{\partial \boldsymbol{w_{j,i}^{(2)}}}\left(\sum_{k=1}^{m}\left(\boldsymbol{w_{j,i}^{(2)}}y_{\boldsymbol{k}}^{(\mathbf{1})} + b_{\boldsymbol{i}}^{(\mathbf{2})}\right)\right) = \delta_{\mathrm{i}}^{(2)}y_{\boldsymbol{j}}^{(\mathbf{1})} \tag{7}$$

O cálculo de bias é análogo, resultando em:

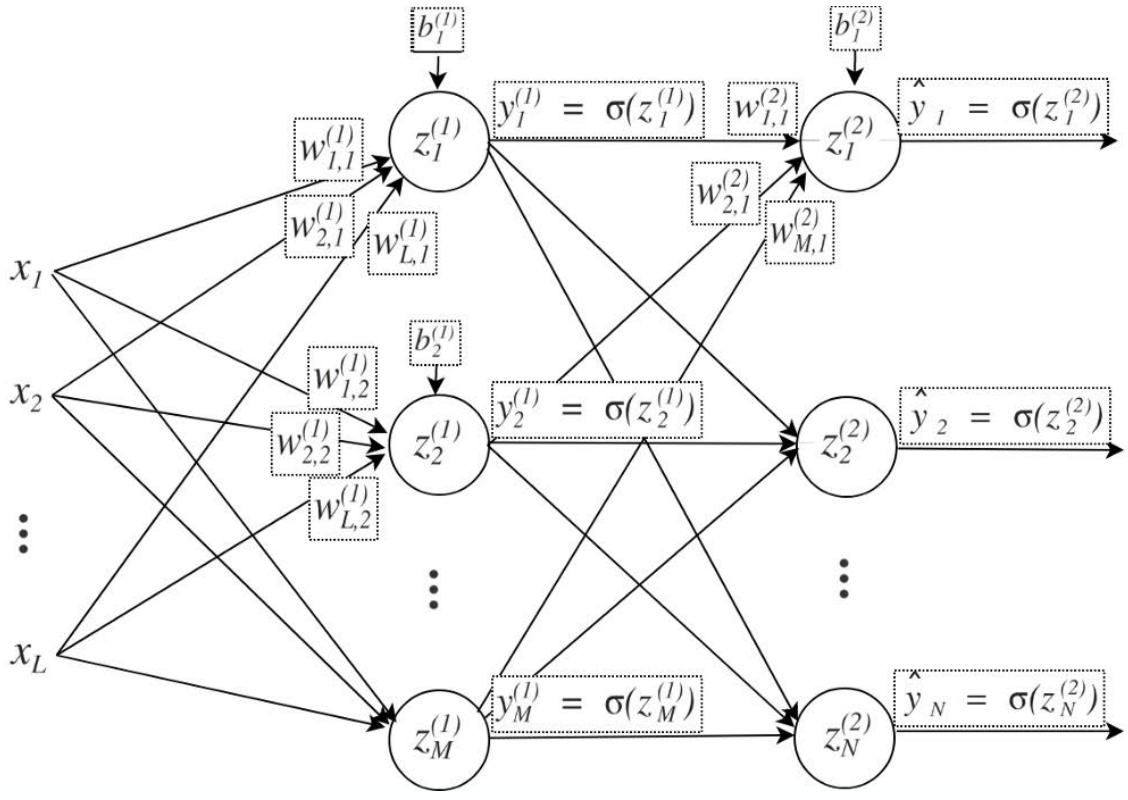
$$\frac{\partial E}{\partial \boldsymbol{b_i^{(2)}}} = \delta_{\mathrm{i}}^{(2)} \tag{8}$$

Com isso é possível aplicar a fórmula geral de atualização dos pesos da última camada tem-se:

$$\boldsymbol{w_{j,i}^{(2)}} = \boldsymbol{w_{j,i}^{(2)}} - \boldsymbol{n}\delta_{\mathrm{i}}^{(2)} y_{\boldsymbol{j}}^{(\mathbf{1})} \tag{9}$$

Nas demais camadas realizam-se os mesmos passos.

Figura 11 - Modelo de uma RNA.



Fonte: https://medium.com/@14prakash/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c Acesso em 1 de outubro de 2018

5 DESENVOLVIMENTO DO PROJETO

Inicialmente foi feito uma revisão bibliográfica e estudos para aplicações de meta-heurísticas no Problema do Caixeiro Viajante. Em seguida, na fase de implementação, alguns problemas surgiram na implementação computacional, entre eles:

- Evitar repetições de soluções nas soluções finais (a maioria das meta-heurísticas trabalha com um conjunto de soluções possíveis), pois caso contrário, a solução final tende à uma população formada por rotas iguais.

- No caso específico dos AG, desenvolver um algoritmo inteligente de forma que as operações de cruzamento e mutação sempre originasse rotas possíveis para a população.

- Problemas com o consumo de memória. Algumas meta-heurísticas usam uma estrutura estática em seus processos, o que pode gerar um tempo maior de processamento, visto que ela utiliza um espaço pré determinado mesmo não sendo utilizado todo espaço.

- Tempo excessivamente longo na execução das meta-heurísticas.

Apesar da fase de testes não ser iniciada, a Busca Tabu apresenta uma melhor resposta do que as outras meta-heurísticas, devido ao fato de que ela foi implementada utilizando estrutura dinâmica e possui um algoritmo que guarda a melhor vizinhança (caminho) que deve ser seguida, o que facilita para encontrar no resultado de boa qualidade. Assim, como o algoritmo de VNS, utiliza o princípio de força bruta, aumenta o tempo de processamento,

dependendo do número de cidades do teste. Além disso, os AGs também apresentam problemas por ser um algoritmo determinístico, ou seja, a população existente vai determinar se será possível chegar ao ótimo global, através dos processos de mutação e cruzamento, o que muitas vezes não ocorre.

Problemas como valores obtidos com as meta-heuristicas estavam longe do ótimo, quando comparado aos resultados ótimos das instancias da literatura, devido ao fato de escolhas prévias de variáveis como, o número máximo de iterações deve ser utilizada para um número específico de cidades, numero de indivíduos na população para o algoritmo genético. Ainda continuavam impactando nos resultados finais. Esse problema foi resolvido graças à implementação de uma RNA *Perceptron* com *Backpropagation* com o auxílio da biblioteca *pybrain* como mostrado na Figura 12. A rede foi treinada até que se obtivesse um erro menor que 0.001 conseguindo melhorar muito os resultados finais como apresentado no capítulo 6.

Figura 12 - Exemplo de uso do *pybrain*.

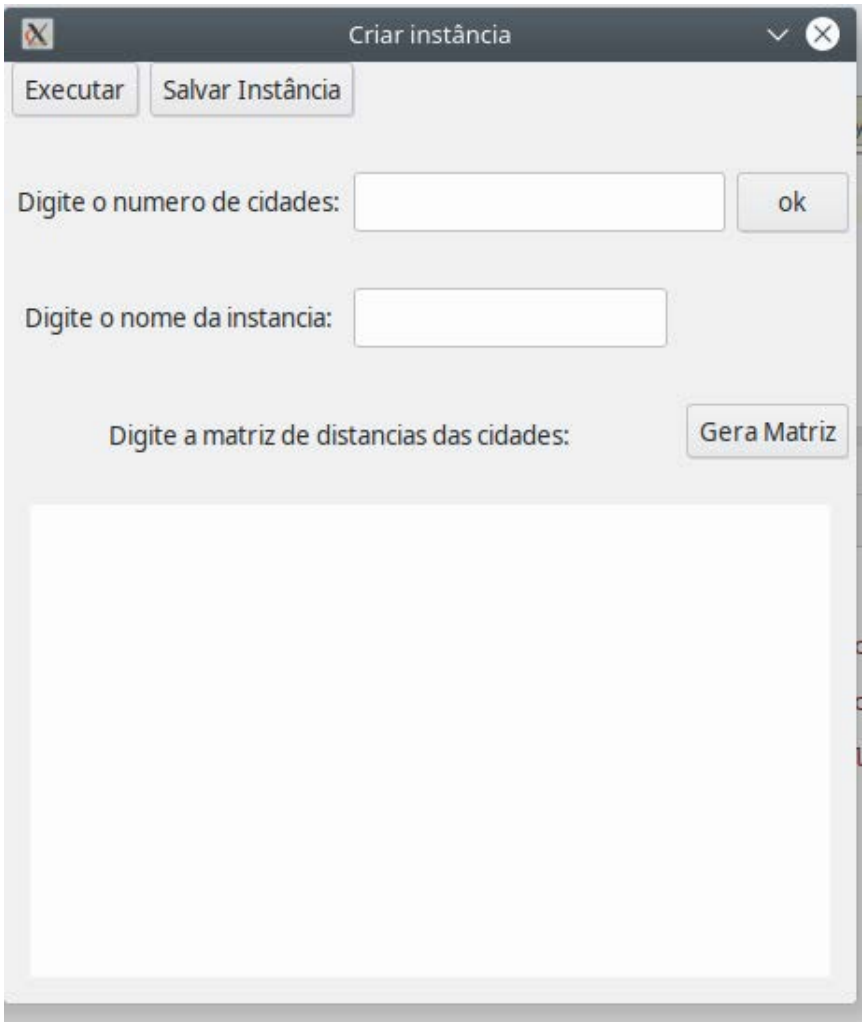
```
from pybrain.tools.shortcuts import buildNetwork
#from pybrain.datasets import SupervisedDataSet
from pybrain.datasets import SupervisedDataSet
#from pybrain.tools.shortcuts import buildNetwork
from pybrain.supervised.trainers.backprop import BackpropTrainer
class rede():
    def redea(self):
        rede = buildNetwork(2,4,1,bias=True)

        base = SupervisedDataSet(2, 1)
        .....
        base.addSample((redevalores[i],1), (rederesultado[i]))

        treinamento = BackpropTrainer(rede, base)
        for i in range(1000):
            print(treinamento.train())
        print('entrada')
        print(base['input'])
        print('\nsaida\n')
        print(base['target'])
        while True:
            numerodecidade = float(input('\nNumero de cidades: '))
            z = rede.activate((numerodecidade,1))
```

Fonte Elaborado pelo autor.

O último estágio foi escolher quais meta-heuristicas são mais vantajosas para resolver o Problema do Caixeiro Viajante e juntá-las aproveitando o melhor de cada método, criando assim, um Sistema Híbrido.

Assim, foi necessário a implementação de uma interface gráfica para que fosse intuitivo utilizar o programa. Tal aplicação foi auxiliada através de um *wrapper* para *python* chamado *PyGtk*. Através dele as interfaces foram todas desenvolvidas como mostrado na Figura 13.

Figura 13 - Interface feita em *Pygtk*.



Fonte: Elaborado pelo autor.

6 RESULTADOS COMPUTACIONAIS

Os algoritmos apresentados anteriormente foram implementados utilizando a linguagem de programação Python 3.0 e, foram realizados testes com instâncias da literatura retiradas em http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html acessado em 1 de Outubro de 2018

O quadro 1 apresenta todos os resultados obtidos com essas instâncias sem o uso *de* RNA. O Quadro 2 apresenta os resultados com a RNA. Os quadros 3 e 4 apresentam o *gap* do resultado ótimo das instancias para o resultado obtido em cada teste

Quadro 1 – Resultados Computacionais sem RNA.

Instância	Número de cidades	Valor ótimo	Algoritmos Genéticos		Busca por Vizinhança		Busca Tabu	
			Valor	Tempo (s)	Valor	Tempo (s)	Valor	Tempo (s)
burma14	14	3323	3423	39	3632	51	3323	12
ulysses16	16	6859	6859	32	6859	65	6859	25
br17	17	39	39	52	49	43	39	36
gr17	17	2085	2423	62	2097	75	2092	24
gr21	21	2707	2707	24	2707	55	2721	23
ulysses22	22	7013	7015	56	7115	87	7132	52
FRI26	26	937	974	76	1272	93	956	32
bayg29	29	1610	1650	149	1630	112	1712	40
bays29	29	2020	2457	158	2272	124	2028	47
ftv33	34	1286	1404	142	2681	115	1286	44
ftv35	36	1473	1650	280	1480	122	1490	58
ftv38	39	1530	1960	331	1685	109	1610	87
dantzig42	42	699	720	298	762	198	715	281
swiss42	42	1273	2393	467	2298	98	1379	120
p43	43	5620	5641	961	5758	161	5669	135
ftv44	45	1613	2503	894	1630	156	1736	117
ftv47	48	1776	1912	983	1881	98	1829	158
att48	48	10628	12632	1011	10646	154	12691	420
ry48p	48	14422	17913	1663	15520	195	14488	311
hk48	48	11461	12700	1205	11578	132	11581	123
gr48	48	5046	15001	1028	7133	545	5146	142
eil51	51	426	958	1506	1114	412	453	151
berlin52	52	7542	8742	1346	19564	661	8799	123
ft53	53	6905	8928	1654	10132	590	10993	49

ftv55	55	1608	1650	1455	2321	542	1608	213
ftv64	64	1839	2944	1854	1852	378	1890	176
ft70	70	38673	53636	1981	39677	413	42929	175
st70	70	675	754	2027	699	439	723	414
eil76	76	538	776	2071	593	243	589	546
pr76	76	108159	213355	2130	128168	345	108830	172
gr96	96	55209	69123	2252	65126	1532	55324	1543
kro124	100	36230	56312	2480	57555	1845	66376	2345
kroA100	100	21282	27371	2387	24361	2432	76395	2431
kroB100	100	22141	78754	2242	29795	2753	25679	1840
kroC100	100	20749	966212	2102	23567	2935	22313	2572
kroD100	100	21294	67876	2190	22970	2568	26041	2521
gr229	229	134602	821415	2789	294051	2784	135752	2899

Fonte: Elaborado pelo autor.

Quadro 2 – Resultados Computacionais com RNA.

Instância	Número de cidades	Valor ótimo	Algoritmo Genético		Busca por Vizinhança		Busca Tabu	
			Valor	Tempo (s)	Valor	Tempo (s)	Valor	Tempo (s)
burma14	14	3323	3323	49	3432	21	3323	12
ulysses16	16	6859	6859	69	6859	33	6859	38
br17	17	39	39	44	45	42	39	38
gr17	17	2085	2422	65	2175	65	2085	24
gr21	21	2707	2707	64	2982	75	2707	23
ulysses22	22	7013	7240	42	7125	55	7054	62
FRI26	26	937	1269	76	1472	87	939	77
bayg29	29	1610	1657	143	1620	93	1722	42

bays29	29	2020	2721	159	2241	120	2028	54
ftv33	34	1286	1409	178	1381	124	1885	98
ftv35	36	1473	1532	140	1580	133	1480	78
ftv38	39	1530	1539	125	1596	132	1530	78
dantzig42	42	699	765	102	882	109	795	89
swiss42	42	1273	1492	467	2198	198	1318	120
p43	43	5620	5691	261	6125	161	5719	123
ftv44	45	1613	1892	284	1982	195	1783	115
ftv47	48	1776	1821	287	1982	238	1794	158
att48	48	10628	12732	311	13422	410	10722	435
ry48p	48	14422	18475	663	15520	597	14476	341
hk48	48	11461	11732	304	12478	522	11482	323
gr48	48	5046	14401	278	6120	478	5098	241
eil51	51	426	453	550	714	661	503	762
berlin52	52	7542	7782	322	7564	521	7842	522
ft53	53	6905	8767	354	7016	542	9727	452
ftv55	55	1608	1650	415	2286	378	1608	523
ftv64	64	1839	2944	554	1752	762	1855	782
ft70	70	38673	38909	781	39677	941	39529	673
st70	70	675	784	423	1299	835	1675	1222
eil76	76	538	776	672	593	945	567	1442
pr76	76	108159	287318	730	168168	1245	163883	872
gr96	96	55209	67289	1053	61228	1532	55324	1561
kro124	100	36230	97900	1280	37825	1845	38376	2345
kroA100	100	21282	67772	1887	23676	2432	22109	2443
kroB100	100	22141	47877	1902	27759	2686	23190	1872
kroC100	100	20749	76683	2502	21567	2945	22013	2572
kroD100	100	21294	85076	2790	22780	2768	21341	2575

gr229	229	134602	161415	3789	154051	3984	135722	2954

Fonte: Elaborado pelo Autor.

Quadro 3 - Gap dos resultados das meta-heurísticas

Instâncias	***Gap* dos resultados utilizando Algoritmos Genéticos**	***Gap* dos resultados utilizando Busca por Vizinhança**	**Gap dos resultados utilizando Busca Tabu**
burma14	100	309	0
ulysses16	0	0	0
br17	0	10	0
gr17	338	12	7
gr21	0	0	14
ulysses22	2	102	119
FRI26	37	335	19
bayg29	40	20	102
bays29	437	252	8
ftv33	118	1395	0
ftv35	177	7	17
ftv38	430	155	80
dantzig42	21	63	16
swiss42	1120	1025	106
p43	21	138	49
ftv44	890	17	123
ftv47	136	105	53
att48	2004	18	2063
ry48p	3491	1098	66
hk48	1239	117	120
gr48	9955	2087	100

eil51	532	688	27
berlin52	1200	12022	1257
ft53	2023	3227	4088
ftv55	42	713	0
ftv64	1105	13	51
ft70	14963	1004	4256
st70	79	24	48
eil76	238	55	51
pr76	105196	20009	671
gr96	13914	9917	115
kro124	20082	21325	30146
kroA100	6089	3079	55113
kroB100	56613	7654	3538
kroC100	945463	2818	1564
kroD100	46582	1676	4747
gr229	686813	159449	1150

Fonte: Elaborada pelo autor.

Quadro 4: Gap dos resultados das meta-heurísticas com RNA

Instâncias	***Gap* dos resultados utilizando Algoritmos Genéticos**	***Gap* dos resultados utilizando Busca por Vizinhança**	**Gap dos resultados utilizando Busca Tabu**
burma14	0	109	0
ulysses16	0	0	0
br17	0	6	0
gr17	337	90	0
gr21	0	275	0
ulysses22	227	112	41

FRI26	332	535	2
bayg29	47	10	112
bays29	701	221	8
ftv33	123	95	599
ftv35	59	107	7
ftv38	9	66	0
dantzig42	66	183	96
swiss42	219	925	45
p43	71	505	99
ftv44	279	369	170
ftv47	45	206	18
att48	2104	2794	94
ry48p	4053	1098	54
hk48	271	1017	21
gr48	9355	1074	52
eil51	27	288	77
berlin52	240	22	300
ft53	1862	111	2822
ftv55	42	678	0
ftv64	1105	113	16
ft70	236	1004	856
st70	109	624	1000
eil76	238	55	29
pr76	179159	60009	55724
gr96	12080	6019	115
kro124	61670	1595	2146
kroA100	46490	2394	827
kroB100	25736	5618	1049

kroC100	55934	818	1264
kroD100	63782	1486	47
gr229	26813	19449	1120

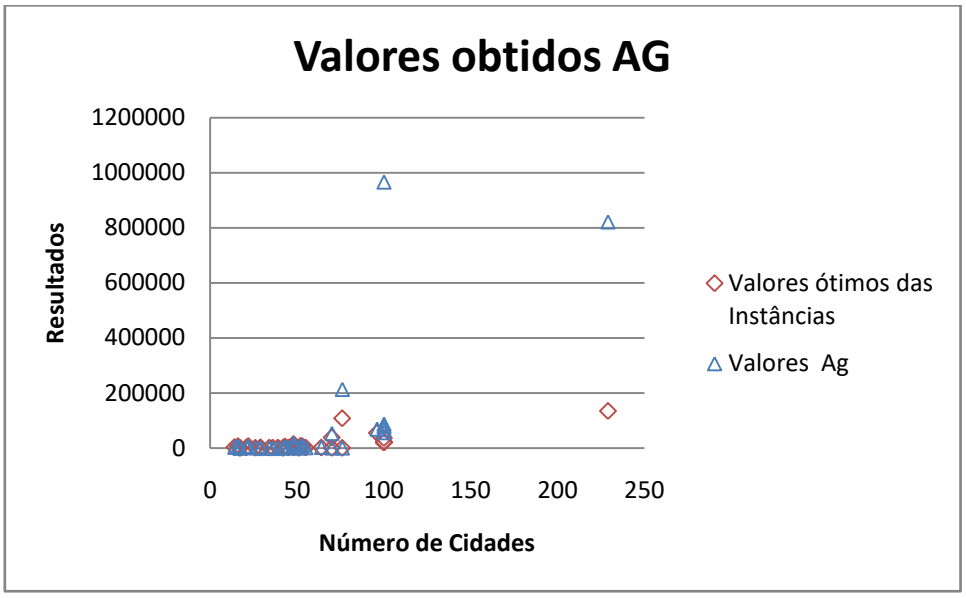
Fonte: Elaborada pelo autor.

Com esses resultados, pode-se chegar as seguintes discussões sobre cada meta-heurística estudada.

6.1 RESULTADOS PARA ALGORITMOS GENÉTICOS

A partir dos testes realizados, pode-se notar que Algoritmos Genéticos não é a melhor meta-heurística a ser utilizada para o Problema do Caixeiro Viajante visto que possui uma grande dificuldade em determinar o ótimo global exato, principalmente quando o número de variáveis (cidades envolvidas) aumenta. Isso ocorre, principalmente, pelo método utilizado, como *crossover* e mutação, que requerem um grande número de iteração para que possa gerar soluções aptas e viáveis a solução final. Além disso, o mal dimensionamento do número de população bem como o número de iterações também pode impactar no resultado final. O Gráfico 1 mostra o valor obtidos com AG comparando com o valor ótimo das instâncias apresentados na literatura.

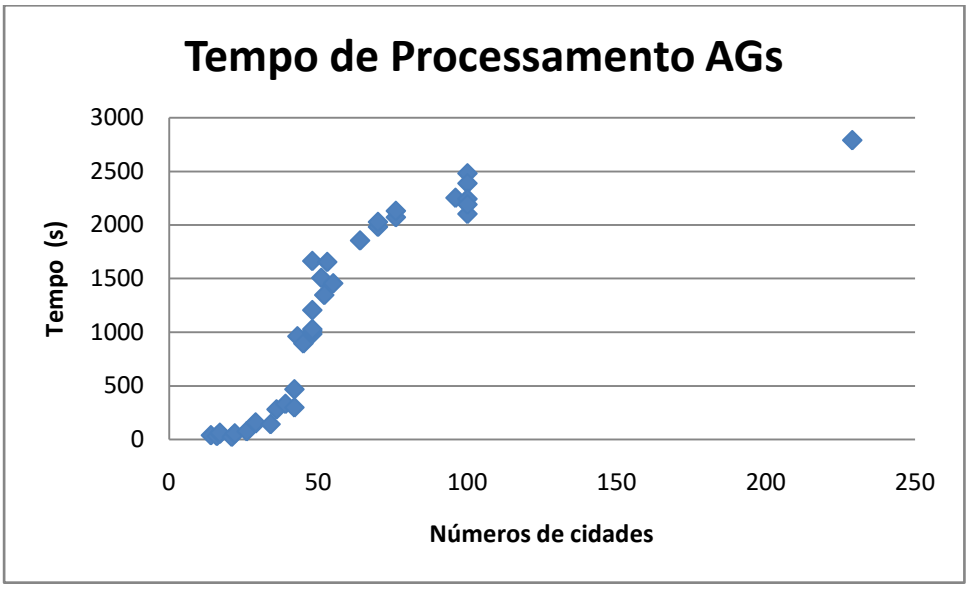
Gráfico 1 – Resultado ótimo comparado com o da meta-heurística AG.



Fonte: Elaborado pelo Autor.

Além disso, o custo computacional é muito elevado, visto que à medida que o número de cidades aumenta, maior o consumo de memória utilizada (por causa da utilização de matrizes para a criação da população). Com isso o tempo de processamento é diretamente proporcional como apresentado no Gráfico 2.

Gráfico 2 – Tempo computacional em relação ao número de cidades.



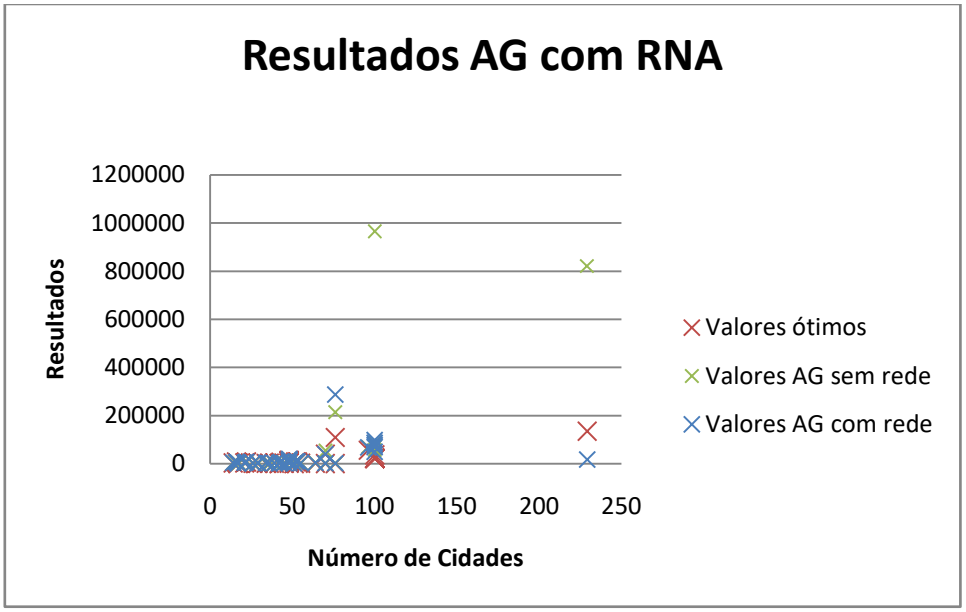
Fonte: Elaborado pelo Autor.

Apesar disso, observa-se que este método possui vantagens, entre as quais pode-se citar:

- É um método robusto e aplicável para uma quantidade muito variada de problemas.
- São de fácil implementação e por isso proporcionam uma maior flexibilidade para tratamento de problemas.
- Consegue convergir rapidamente para um resultado melhor que o inicial

Através do uso da RNA para estimar o número de populações bem como o número de interações houve uma nítida melhora no resultado final como apresentado no Gráfico 3.

Gráfico 3 – Resultado ótimo comparado com o da meta-heurística AG com o uso da RNA.

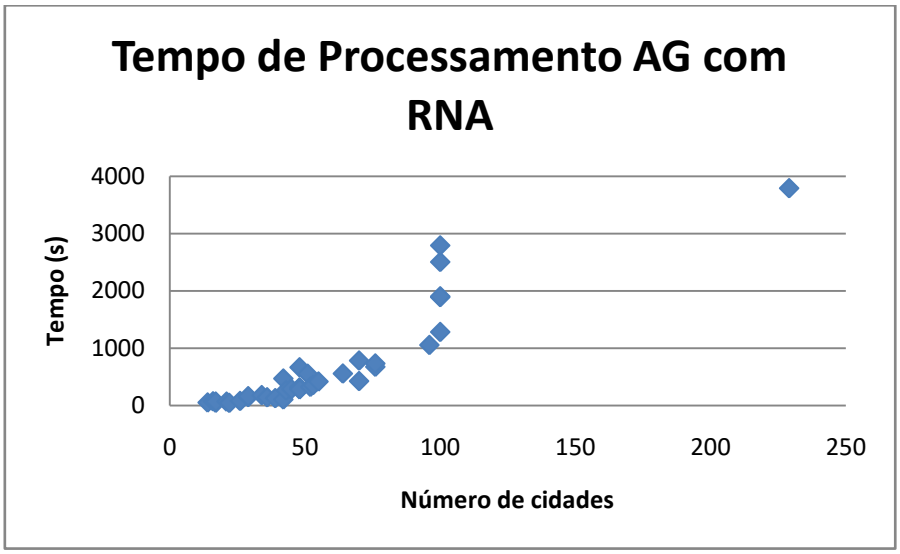


Fonte: Elaborado pelo Autor.

Apesar da melhora no resultado final o tempo aumentou consideravelmente com um número determinado de cidades, tornando-se mais custosa quando comparado às instâncias sem uso da RNA. Isso ocorre devido ao treinamento da RNA, bem como o aumento de interações aumentando assim o número de vezes que era realizado o processo de *crossover* e mutação afetando o tempo final.

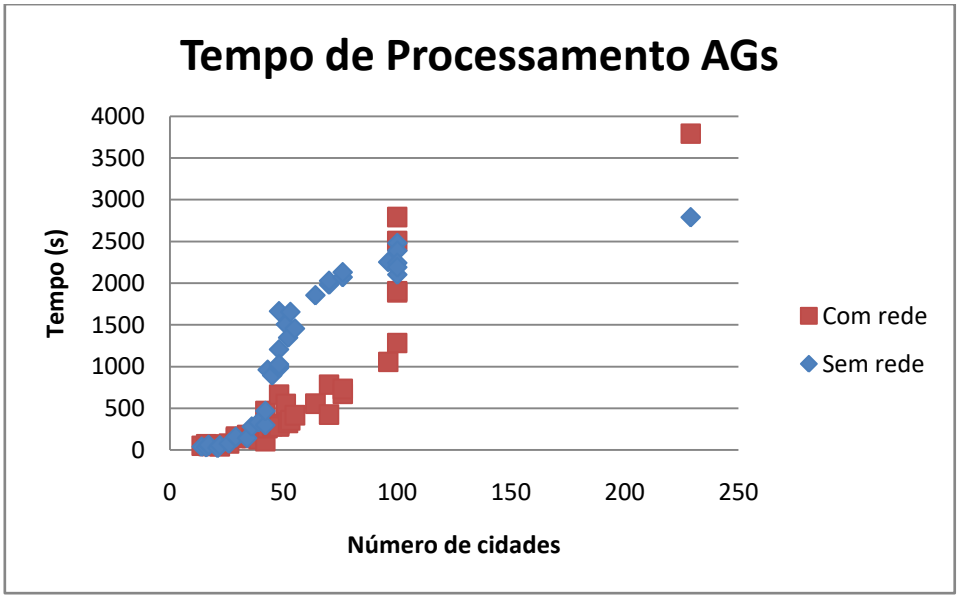
O Gráfico 4 apresenta o tempo de processamento do algoritmo Genético com RNA e o Gráfico 5 apresenta o tempo com RNA comparado com o tempo sem o uso da RNA.

Gráfico 4 – Tempo de Processamento AG com RNA.



Fonte: Elaborado pelo Autor.

Gráfico 5 – Tempo computacional em relação ao número de cidades.

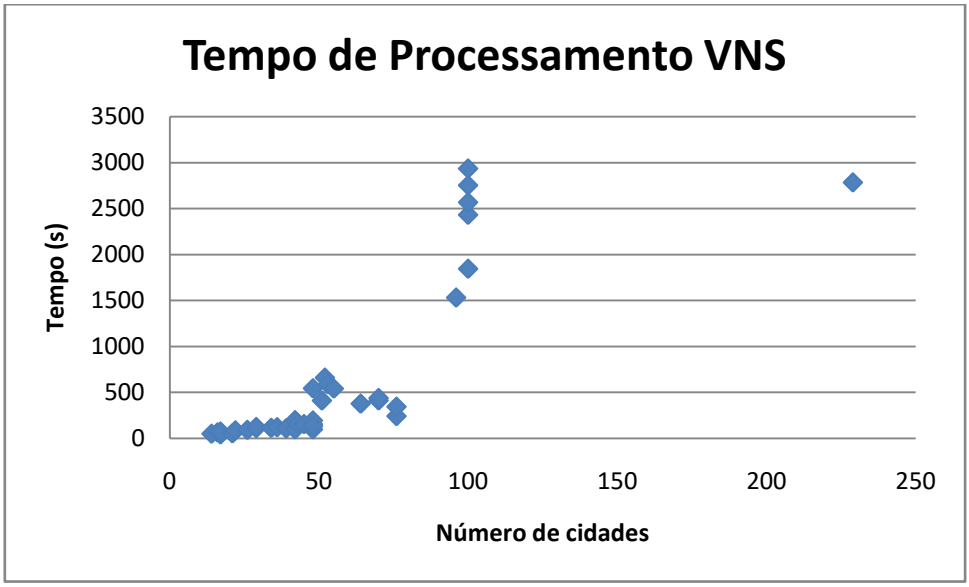


Fonte: Elaborado pelo autor.

6.2 RESULTADOS PARA A BUSCA POR VIZINHANÇA

O método apresentou um bom desempenho em relação a proximidade do ótimo global. Porém, esse método não deve ser utilizado para testes que possuem muitas instâncias, visto que é um algoritmo que lida com trocas sistemáticas (espécie de “força bruta”) e à medida que a vizinhança estudada aumentar ela pode cair em trocas já realizadas anteriormente, que foram descartadas, ou realizar troca já vigentes na solução fazendo assim mudanças desnecessárias ou ficando presos em buscas cíclicas (resultando no aumento de tempo de processamento). O Gráfico 6 mostra o comportamento da meta-heurística em relação ao tempo.

Gráfico 6 – Tempo computacional em relação ao número de cidades.

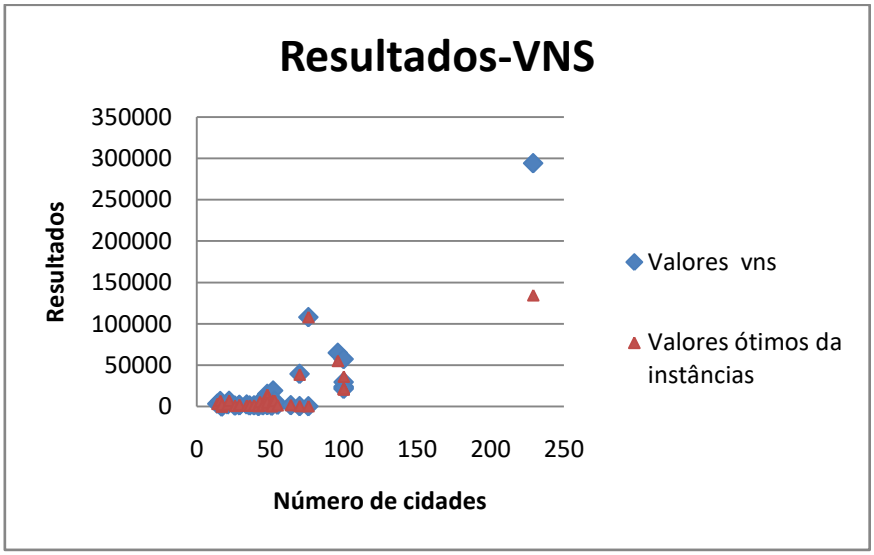


Fonte: Elaborado pelo Autor.

Apesar disso, esse método apresenta fácil implementação e resultados satisfatórios para poucas cidades (tendo em vista a memória e o tempo computacional como análise). Porém, para instâncias muito grandes ele se tornou inviável devido ao tempo e em relação a resultados finais, devido à

realização de trocas feitas anteriormente. O Gráfico 7 mostra o seu desempenho em relação ao valor ótimo das instâncias.

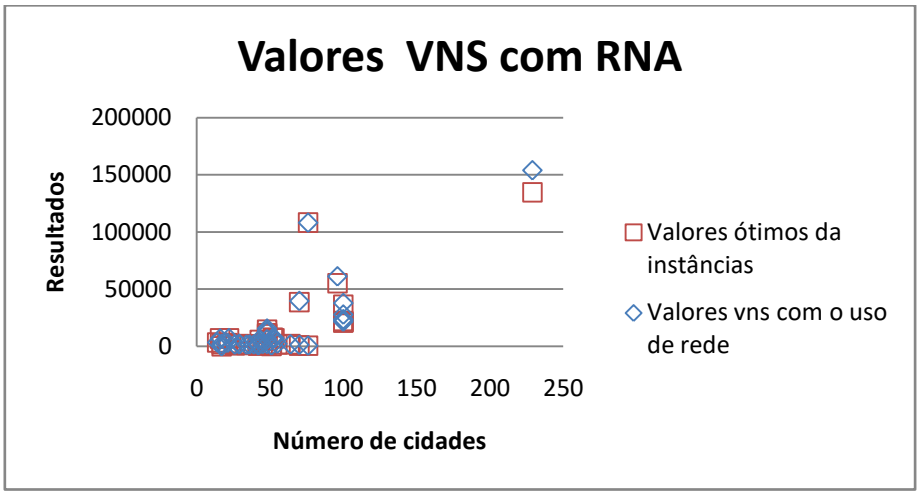
Gráfico 7 – Resultado ótimo comparado com o da meta-heurística VNS.



Fonte: Elaborado pelo Autor.

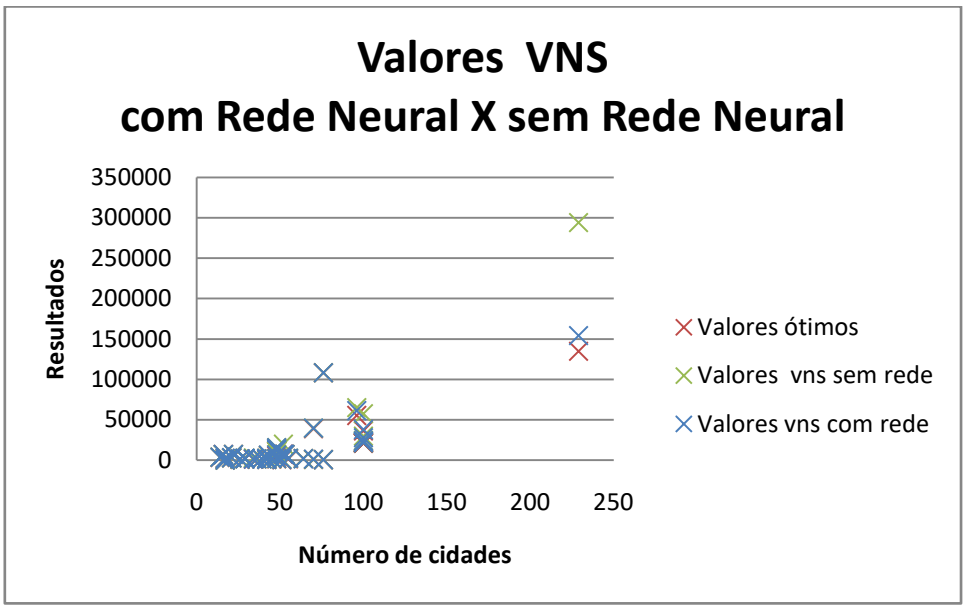
Com o uso da RNA para a VNS, também houve uma melhora dos resultados finais, porém devido o aumento do número de iterações o tempo de processamento aumentou, tornado o uso dessa meta-heuristica inviável. O Gráfico 8 apresenta o resultado com o uso da RNA e o Gráfico 9 apresenta o tempo de processamento da meta-heurística com o uso da RNA em comparação com o tempo sem o uso da RNA.

Gráfico 8 – Resultado ótimo comparado com o da meta-heurística VNS usando RNA.



Fonte: Elaborado pelo Autor.

Gráfico 9 – Tempo de Processamento VNS com RNA X sem RNA.



Fonte: Elaborado pelo Autor

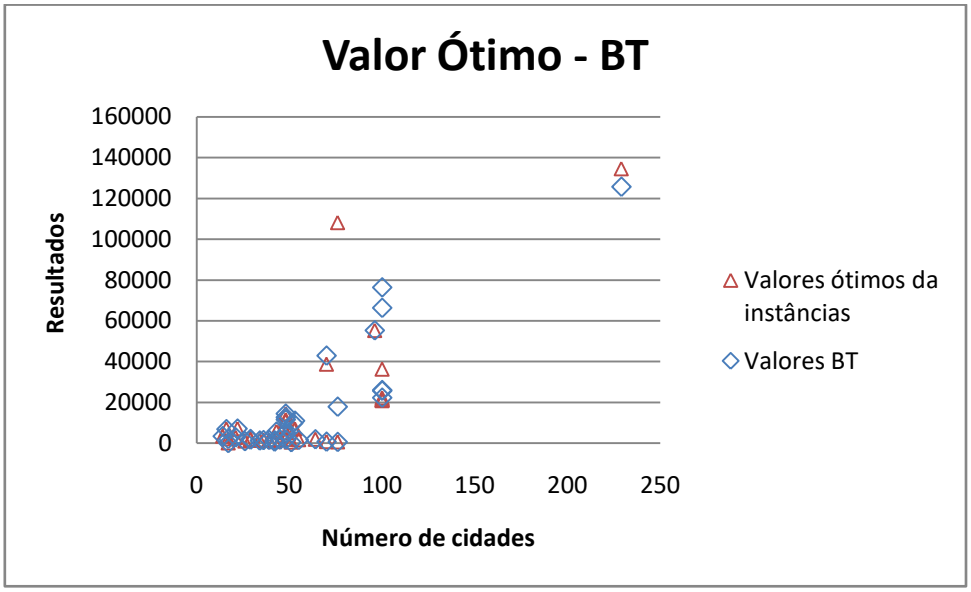
6.3 RESULTADOS PARA A BUSCA TABU

Esta meta-heurística foi a que apresentou melhores resultados em comparação às outras apresentadas neste estudo. Devido ao fato dela ser uma

variação do VNS, facilita em não cair em uma busca cíclica de um ótimo global (devido à criação de suas listas adaptativas). Além disso, o algoritmo considera todas as soluções possíveis. Isto se deve ao fato que ele considera todo o espaço de busca (vizinhança) para determinar o melhor caminho**.**

O Gráfico 10 apresenta o desempenho da Busca Tabu em relação ao valor ótimo das instâncias apresentado nas instâncias da literatura.

Gráfico 10 – Resultado ótimo comparado com o da meta-heurística Busca Tabu.



Fonte: Elaborado pelo Autor.

Apesar da Busca Tabu possuir pontos que podem ser melhorado, como pode-se explicitar:

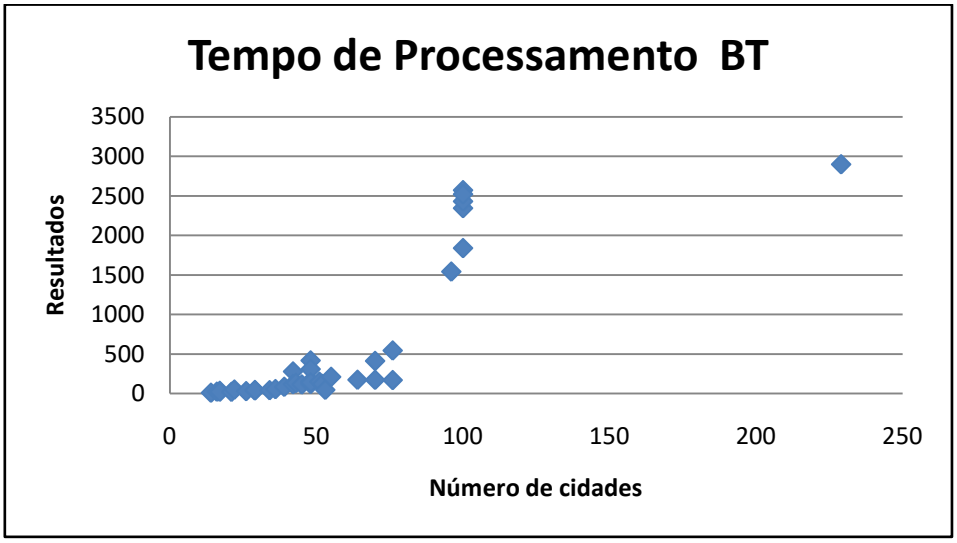
- Se o dimensionamento for muito grande, a Busca Tabu pode ficar presa por um tempo em uma vizinhança, aumentando o tempo de processamento e consequentemente a memória visto que ela também aumentará.

- Com um número muito grande de cidades também aumenta as chances de espaços salvos na lista tabu serem descartados depois, fazendo assim um uso desnecessário de memória.

- O mal dimensionamento do número de iterações pode acarretar em um resultado muito longe do ideal.

O Gráfico 11 mostra o seu desempenho (tempo computacional) em relação ao número de cidades:

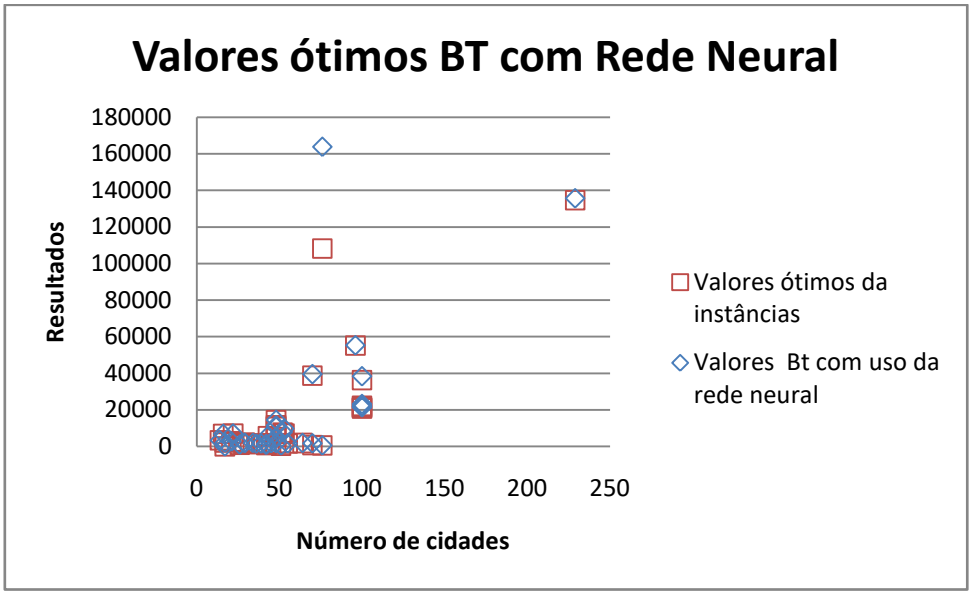
Gráfico 11 – Tempo computacional em relação ao número de cidades.



Fonte: Elaborado pelo Autor.

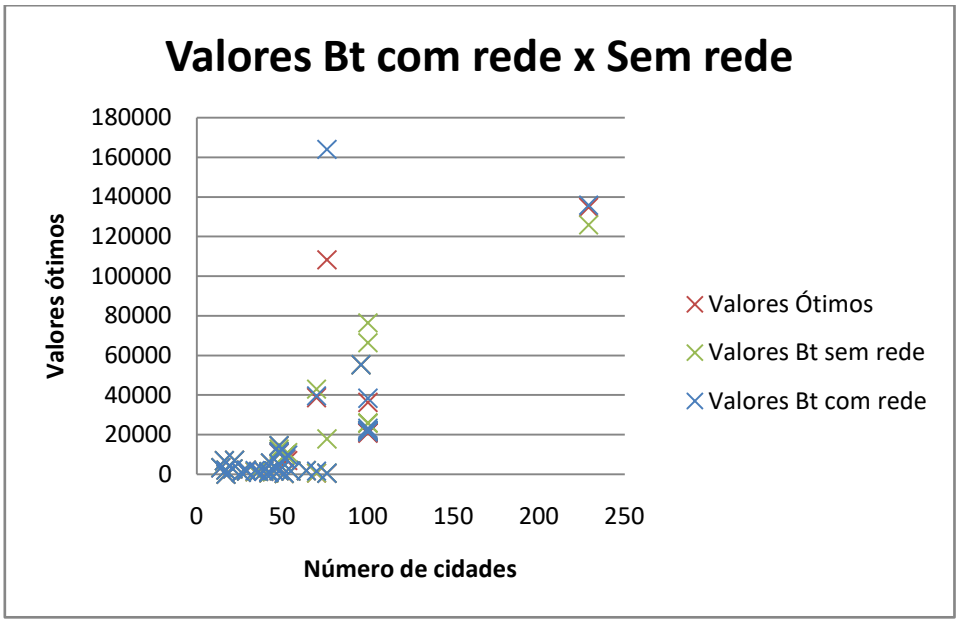
Através do uso da RNA, o mau dimensionamento foi solucionado melhorando os resultados obtidos pela meta-heurística como observado no Gráfico 12. Porém, com o treinamento da RNA também houve um aumento como nas demais meta-heurísticas como observado no Gráfico 13.

Gráfico 12 – Resultado ótimo comparado com o da meta-heurística Busca Tabu com RNA.



Fonte: Elaborado pelo autor

Gráfico 13 – Tempo de processamento Busca Tabu com RNA X sem RNA.



Fonte: Elaborado pelo autor

Como é possível observar com os resultados obtidos e pela discussão apresentada anteriormente, a Busca Tabu foi a meta-heurística que obteve os melhores resultados comparados com o ótimo. Sendo que os resultados de todos os métodos melhoraram com o uso da RNA, que dimensionaram melhor

algumas variáveis que impactavam no bom funcionamento do algoritmo. Os Quadros 5, 6 e 7 apresentam alguns testes das várias meta-heurísticas que mostram como essas variáveis quando submetidas a várias entradas podem gerar resultados melhores.

Quadro 5 – Testes variando o tamanho da população e número de iterações.

Instância	Numero de Cidades	Valor Ótimo	Algoritmos Genéticos		
			Tamanho da população	Numero de iterações	Resultado obtido
gr17	17	2085	4	20	3950
			4	50	3282
			10	20	3215
			10	50	3215
			4	100	3282
			10	100	29773
			10	1000	2449
			10	2000	2085
bays29	29	2020	5	40	4696
			10	40	4579
			10	100	4054
			20	100	4097
			30	2000	2491
			40	3500	2390
			80	6500	2025
Ft53	53	6905	5	50	14696
			50	5000	13212
			250	5000	15472
			150	7000	9735

			250	200000	7120
krA100	100	21282	50	100	149650
			100	1000	124625
			200	10000	104137
			500	100000	67772
			1000	150000	22520
Ftv33	33	1286	4	40	3102
			10	400	2445
			20	4000	1832
			120	90000	1699
FRI26	26	937	4	20	1286
			10	50	1872
			10	200	1325
			15	4000	937

Fonte: Elaborado pelo Autor.

Quadro 6 – Testes variando número de iterações para a Busca Tabu.

Instância	Numero de Cidades	Valor Ótimo	Busca Tabu	
			Numero de iterações	Resultado obtido
gr17	17	2085	100	4982
			500	4870
			1000	3560
			5000	2870
			5500	2270
bays29	29	2020	40	4769
			80	4579
			100	4054

			1000	3597
			2000	2950
			3500	2250
			6500	2025
Eil50	50	6905	50	10696
			5000	8953
			5000	7921
			7000	7120
krA100	100	21282	100	129556
			3000	80329
			4000	53291
			50000	32191
			180000	22191
FRI26	26	937	20	1468
			50	1271
			100	1024
			200	937

Fonte: Elaborado pelo Autor.

Quadro 7 – Testes variando número de iterações para a Busca por Vizinhança.

Instância	Numero de Cidades	Valor Ótimo	Busca Por Vizinhança	
			Numero de iterações	Resultado obtido
gr17	17	2085	100	8790
			500	8270
			10000	3560
			50000	2870
			155000	2270

bays29	29	2020	1000	7953
			100000	6597
			650000	2027
Eil50	50	6905	5000	12696
			50000	8563
			800000	7220
krA100	100	21282	100	177557
			30000	80329
			40000	63231
			150000	35496
			2800000	22191
FRI26	26	937	200	3468
			5000	2218
			10000	1134
			200000	937

Fonte: Elaborado pelo Autor.

7. CONCLUSÕES

Como pode-se notar, Problemas de Roteirização estão presente em nosso cotidiano como o Problema do Caixeiro Viajante (um problema combinatórial).

O objetivo principal deste projeto foi descrever os métodos desenvolvidos, bem como os resultados obtidos para a solução do problema, utilizando diversas meta-heurísticas.

Observou-se que estes algoritmos são suficientemente complexos e muito eficientes para busca de soluções quase-ótimas, pois utilizam mecanismos de busca adaptativo poderosos e robustos, devido ao fato que pode-se encontrar diversos problemas para encontrar o ótimo global.

Observa-se também, pelos resultados dos testes, que a Busca Tabu foi a que mais se aproximou dos resultados ótimos em um tempo de processamento razoavelmente “aceitável”, quando comparadas a tempo de processamento de empresas, sendo a mais indicada dentre as que foram desenvolvidas para a resolução do Problema do Caixeiro Viajante.

Além disso, devido a natureza probabilística, para o caso do Problema do Caixeiro Viajante o algoritmo não conseguiu alcançar seu ótimo, porém conseguem diminuir rapidamente os valores obtidos no inicio de seu funcionamento. Vale ressaltar que, a VNS é um algoritmo que lida com trocas sistemáticas e à medida que a vizinhança estudada aumenta, ela pode cair em trocas já realizadas anteriormente, que foram descartadas, ou realizar troca já vigentes na solução fazendo assim mudanças desnecessárias ou ficando presos em buscas cíclicas, tornando-se inviável utilizar este tipo de algoritmo.

Além disso, o uso da RNA foi de grande ajuda visto que a rede permite realizar analises de dados e tomar decisões bem como aprender simulando um sistema nervoso biológico. Outros pontos a serem destacar são: a capacidade de generalização, visto que podem ser utilizados em diferentes tipos de problemas; adaptabilidade, pois, a medida que mais dados entram ela

consegue aprender sem a necessidade de alterar sua arquitetura; fácil implementação; entre outros motivos.

Com o uso da *backpropagation* foi possível cria uma RNA confiável visto que o erro diminuiu e, sendo assim propagado em sentido oposto ao de propagação do sinal funcional, reajustando os pesos entre as camadas. Além disso, a *backpropagation* é uma RNA muito utilizada devido a suas características de trabalhar com redes multi-camadas e resolver erros não-linearmente separáveis, como foi implementado junto as meta-heurísticas.

Conclui-se ainda que, com uso da RNA todos os métodos melhoraram seus resultados ótimos, pois através dela foi possível dimensionar melhor algumas variáveis que impactavam no bom funcionamento do algoritmo. Apesar disso, o tempo dos métodos aumentaram pelo fato do treinamento da RNA e por realizar mais interações de acordo com os novos valores atribuído pela RNA. Só assim, através desta junção, foi possível criar um sistema híbrido inteligente que busca unir duas ou mais técnicas a fim de obter melhorias nos resultados finais (Melo e Martinhon, 2004). A partir desse sistema híbrido foi possível construir um software como apresentado na Seção 5, que conseguisse resolver o problema com soluções de boa qualidade e que utilizasse os conceitos da RNA e das Meta-Heurísticas.

8 REFERENCIAS

Carvalho B. A.; Barriveira R.; Tavares C.;Rodrigues K.; Losco F**. Busca Tabu Aplicada ao Problema do Caixeiro Viajante**, Acesso em 7 de março de 2018 <http://www.inf.ufpr.br/aurora/disciplinas/topicosia2/downloads/trabalhos/BuscaTabu TSP.pdf>

Cunha, C. B. **Aspectos práticos da aplicação de modelos de roteirização de veículos a problemas reais**. Transportes, v.8 , n.2, p.51-74, 2000.

GHOSH, J. **Network Ensembles and Hybrid Systems**. Tutorial at the IEEE International Conference on Neural Networks, 1997.

Glover, F. **Tabu Search and Adaptive Memory Programing - Advances, Applications and Challenges.** In Barr, R.S., Helgason, R.V., and Kennington, J.L., editors, Interfaces in Computer Science and Operations Research, p,1-75. Kluwer Academic Plublishres, Dordrecht, MA, EUA, 1996.

Goldberg, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning.** Addison Wesley, Reading, New York, 1989.

Haykin, S . **Redes Neurais: Princípios e Prática** , 2ª ed. Editora Bookman, Porto Alegre, RS,2001.

Laporte, G.; Gendreau M.; Potvin J.Y. e Semet F., **Classical and modern heuristics for the vehicle routing problem, International Transactions in Operational Research**, v.7, n. 4/5, p.285-300, 2000.

McCulloch, W. S., Pitts, W., 1943, "**A logical calculus of the ideas immanent in nervous activity**", Mathematical Biophysics, v. 5, pp. 18-27.

Melo, V. A. and Martinhon, C. A. . **Metaheurísticas híbridas para o problema do caixeiro viajante com coleta de prêmios**. In Anais 36 do Simpósio

Brasileiro de Pesquisa Operacional, São João Del Rei, Brazil, 2004,p. 1295–1306.

Mladenovic N. and Hansen. P. **Variable neighborhood search. Computers &Operations Research**,v. 24, ed. Elsevier, p. 1097–1100, 1997.

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.

Plous, S. (1993). **McGraw-Hill series in social psychology. The psychology of judgment and decision making**. New York, NY, England: Mcgraw-Hill Book Company.

Rumelhart D. E., Hinton G. E., Williams R. J., 1986, "**Learning internal representations by error propagation**", In Parallel distributed processig: Exploretimes in the microstructure of the cogitio, v. I, pp. 318-362. Bradford Books, Cambridge, MA.

Silva, A. F.; Oliveira, A. C. de. **Algoritmos Genéticos: alguns experimentos com os operadores de cruzamento ("Crossover") para o problema do caixeiro viajante assimétrico.** In: ASSOCIAÇÃO BRASILEIRA DE ENGENHARIA DE PRODUÇÃO, XXVI. 2006, Fortaleza. Anais. Fortaleza: ABEPRO – Encontro Nacional de Engenharia de Produção (ENEGEP), 2006.