ISSN: 1984-3151

PROBLEMA DO CAIXEIRO VIAJANTE: UM ESTUDO COMPARATIVO DE TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL

TRAVELING SALESMAN PROBLEM: A COMPARATIVE APPROACH BY USING ARTIFICIAL INTELLIGENCE TECHNIQUES

Fabiano das Mercês Calado; Ana Paula Ladeira

Centro Universitário de Belo Horizonte, Belo Horizonte, MG

fabianocalado@gmail.com; aladeira.unibh@gmail.com

Recebido em: 22/05/2011 - Aprovado em: 30/06/2011 - Disponibilizado em: 24/07/2011

RESUMO: O Problema do Caixeiro Viajante é tema de pesquisa devido a sua complexidade. Várias técnicas são constantemente pesquisadas para obter soluções aproximadas, mas eficientes. Neste trabalho são comparadas as técnicas de algoritmos genéticos, redes neurais auto-incrementáveis de Kohonen e um algoritmo heurístico proposto na literatura. Os resultados obtidos mostraram que os algoritmos genéticos apresentaram os melhores índices de acerto.
PALAVRAS-CHAVE: Caixeiro Viajante. Algoritmo Genético. Redes Neurais Artificiais. Redes de Kohonen. Heurística.

ABSTRACT: Several techniques for the Traveling Salesman Problem are constantly being researched to obtain approximate solutions, and efficient. This paper compared genetic algorithms, Kohonen neural networks self-organizing and a heuristic proposed in the literature. The results illustrate that genetic algorithms showed the best accuracy rate.
KEYWORDS: TSP. Genetic Algorithms. Kohonen Net.

1 INTRODUÇÃO

Os problemas da classe NP-completo têm sido um grande desafio devido à sua complexidade: geralmente fatorial ou exponencial. Para um número considerável de exemplos não existe recurso computacional para resolvê-los em tempo hábil. Segundo Helsgaun (2000), não há como achar a solução ótima sem o uso da força bruta, ou algum algoritmo exato, que normalmente é de complexa programação, com códigos da ordem de 10.000 linhas e exagerado tempo de execução. Diante disso, é comum utilizar heurísticas para obter uma solução aproximada e consequentemente, mais rápida. Nos últimos anos, tem-se observado também a utilização de técnicas de Inteligência Artificial (IA) como algoritmos genéticos e redes neurais para resolver problemas desse tipo.

Um problema antigo e sabidamente conhecido como NP-difícil é o problema do caixeiro viajante (PCV). O PCV consiste em achar o menor caminho, dado um conjunto *n* de cidades, visitando todas elas uma vez, e retornando à cidade inicial. A solução consiste numa sequência de cidades, na ordem de visitação. Para um grande número de cidades é inviável avaliar todas as possibilidades, pois o número de tentativas cresce exponencialmente. Sendo assim, torna-se essencial utilizar métodos heurísticos que amenizem essa exploração combinatória.

O PCV tem importantes aplicações práticas, tais como identificar itinerários de cobertura de regiões, como, por exemplo, a rota de um carteiro, visto que diminuir o trajeto pode significar economia de tempo e de pessoal. Outra aplicação prática é o processo de furação de placas para circuito impresso. Como

existem vários furos de diâmetros diferentes, e a troca da ferramenta pode demandar tempo, os furos de mesmo tamanho devem ser feitos de maneira sequencial, percorrendo-se a menor distância possível, para diminuir o tempo gasto numa produção em série.

Segundo Helsgaun (2000), esse problema, por ser combinatorial, tem *(n-1)!/2* soluções possíveis para um conjunto de *n* nós, e que por isso é importante escolher uma boa heurística para obter uma solução viável: baixo custo computacional e, se possível, próxima da ótima. Ainda de acordo com o autor, o maior mapa com solução ótima conhecida é de 7.397 cidades, sendo que, a solução ótima conseguida com algoritmos exatos foi obtida por uma rede de computadores com entre três a quatro anos de processamento. Além disso, o autor destaca que o algoritmo heurístico proposto no seu estudo encontrou uma solução ótima em sete das dez tentativas com uma média de 3,6 horas.

Além dos algoritmos heurísticos, outras técnicas da IA podem ser aplicadas neste problema, dentre elas as redes neurais, que são eficientes por causa de sua adaptabilidade, ou seja, capacidade de aprendizado por treinamento; e os algoritmos genéticos, que são eficientes pela grande diversidade de soluções testadas e pela convergência através do cruzamento das soluções mais promissoras (baseado na seleção natural).

Mais especificamente, as redes neurais são usadas para o problema do caixeiro viajante pela sua capacidade de convergir gradativamente aos nós, que representam as cidades, tomando uma configuração similar ao mapa das cidades. O seu desempenho depende do ajuste dos parâmetros de aprendizado da rede e do tipo de rede utilizada. Os algoritmos genéticos, por sua vez, são usados pela capacidade de gerar vários caminhos aleatórios e convergir para os caminhos mais curtos através de seleção e recombinação. O seu desempenho depende do operador de recombinação usado.

Diante do supracitado, o objetivo geral deste estudo é comparar métodos inteligentes para resolver o problema do caixeiro viajante, tais como algoritmos genéticos, redes neurais artificiais e o algoritmo heurístico Lin-Kernighan, apresentado por Helsgaun (2000).

Dentre os objetivos específicos tem-se:

1. Investigar acerca dos métodos utilizados para o problema do caixeiro viajante;

2. Obter bases de dados reconhecidas em experimentos envolvendo o problema do caixeiro viajante;

3. Comparar o desempenho de operadores de recombinação dos algoritmos genéticos;

4. Comparar o desempenho entre as técnicas de IA escolhidas;

5. Verificar possibilidades de melhoria dos algoritmos na resolução desse problema.

Na próxima seção serão descritas as técnicas avaliadas no presente estudo, na seção 3 será exposta a metodologia adotada na implementação do presente trabalho. Na seção 4 serão mostrados e discutidos os resultados obtidos, enquanto que a conclusão é apresentada na seção 5.

2 FUNDAMENTAÇÃO TEÓRICA

Dentre as técnicas aplicadas ao problema do caixeiro viajante, foram escolhidos os algoritmos genéticos, as redes neurais e um algoritmo heurístico Lin-Kernighan, apresentado em Helsgaun (2000), que serão detalhados a seguir. Essa escolha foi baseada nos resultados apresentados em Dorigo e Gambardella (1997).

Para os algoritmos genéticos foram selecionados dois operadores de cruzamento, além do cruzamento

aleatório, para comparação de desempenho. Já a rede neural usada foi a de Kohonen (HAYKIN, 1999), e o algoritmo heurístico Lin-Kernighan foi simulado, usando uma versão simplificada para teste.

2.1 ALGORITMOS GENÉTICOS

Os algoritmos genéticos são baseados no princípio da seleção natural para resolver os problemas. Utiliza-se uma população de indivíduos, onde cada indivíduo é uma possível solução para o problema em questão.

Para o problema do caixeiro viajante, cada indivíduo pode ser uma sequência que representa a ordem em que as cidades são visitadas. Através de seleção e cruzamento entre os indivíduos da população, indivíduos melhores são obtidos, gradativamente.

Os indivíduos são classificados de acordo com uma função de avaliação, e existem várias técnicas para seleção e cruzamento desses indivíduos. Para o PCV, a função de avaliação soma a distância entre cada cidade do indivíduo. Um indivíduo é considerado melhor que outro quando a sua distância total é menor.

Em alguns estudos analisados foi usado um algoritmo otimizador (Lin-Kernighan), cuja versão chamada 2-opt é utilizada como operador de mutação, oferecendo uma boa chance de melhorar um indivíduo através da inversão de parte de seu caminho.

Essa estratégia tende a melhorar significativamente o desempenho dos algoritmos genéticos para o PCV. Tais algoritmos usam um processo de otimização, comumente chamado de busca local, como observado em Merz (2002).

A estrutura geral de um algoritmo genético é mostrada a seguir.

ALGORITMO 1 – ESTRUTURA BÁSICA DO ALGORITMO GENÉTICO.

```
Algoritmo
  Inicializa população
    enquanto (condição de parada não satisfeita)
      realiza Seleção
      realiza Cruzamento
      aplica Mutação
    fim_enquanto
  fim_algoritmo
```

As etapas de Seleção, Cruzamento e Mutação serão apresentadas nas seções seguintes.

2.1.1 SELEÇÃO

A cada iteração, que é chamada de geração, são selecionados os melhores indivíduos. Esse processo de identificar os melhores indivíduos para serem perpetuados para a próxima geração é chamado de seleção. Dentre as estratégias de seleção disponíveis, no presente trabalho foi usada a seleção por torneio, que consiste em escolher dois pais aleatoriamente da população atual e selecionar o que possui melhor função de avaliação para ficar para a próxima geração. Os indivíduos de menor avaliação são considerados perdedores e, portanto são descartados.

Desta forma, parte da população passa para a próxima geração e os melhores indivíduos têm mais chance de sobreviver.

2.1.2 CRUZAMENTO

O algoritmo genético melhora sua população através do cruzamento. O cruzamento consiste em escolher dois indivíduos da população atual e gerar um novo indivíduo (chamado de filho) baseado nos genes dos pais. O PCV tem uma característica peculiar para o cruzamento, que é a permutação. Um indivíduo não

pode conter cidades repetidas, portanto o cruzamento deve manter o controle de quais cidades já foram colocadas no caminho do indivíduo para que não haja repetição.

Outra característica importante do operador de cruzamento é a escolha da sequência de montagem da cadeia de genes do filho. É desejável que as cidades adjacentes presentes no filho sejam adjacentes também nos pais, ou seja, que o indivíduo seja gerado com as arestas dos pais, refletindo a qualidade da estrutura dos mesmos.

Nesse sentido, deve-se obedecer à característica de respeitabilidade: o indivíduo deve conter todas as arestas presentes em ambos os pais. Segundo Merz (2002), essa é uma propriedade importante para operadores de cruzamento no PCV.

No presente trabalho foram implementados dois operadores de cruzamento: o EER (*Enhanced Edge Recombination*), que usa informação de adjacência para a montagem da cadeia genética, e um operador guloso, que compara o tamanho das arestas para a montagem do novo indivíduo. Ambos geram indivíduos baseados nas cadeias dos pais, mas com uma diferença básica: o EER dá preferência à respeitabilidade, ao contrário do guloso. Porém, o operador guloso cria o indivíduo somente com arestas presentes nos pais.

Os algoritmos genéticos tendem a convergir muito rápido quando o cruzamento é feito deliberadamente, devido ao fato de a população ficar homogênea, ou seja, com vários indivíduos idênticos (MERZ, 2000).

Neste sentido, pode-se adotar a restrição de que um indivíduo gerado pelo cruzamento só seja adicionado na população, se não houver outro na população com a mesma sequência genética. Assim, uma grande variedade de indivíduos na população, aumenta as chances de escapar de ótimos locais.

2.1.2.1 ENHANCED EDGE RECOMBINATION OPERATOR

O operador EER supracitado usa uma tabela de arestas para construir um indivíduo que herde o máximo de informação possível dos pais. Esta tabela armazena todas as conexões dos pais que chegam em uma cidade e que saem dela. Como a distância é a mesma nos dois sentidos (considerando o PCV simétrico), cada cidade tem pelo menos duas e no máximo quatro arestas (duas de cada pai).

O processo de construção da tabela de adjacência é simples: para cada cidade, armazenam-se quais cidades são adjacentes a esta nos caminhos dos indivíduos pais. Para identificar as arestas que aparecem em ambos os pais, basta colocar um sinal negativo na cidade correspondente.

Por exemplo, considere um problema com seis cidades (1, 2, 3, 4, 5, 6). Se os pais forem P1=(1-2-3-4-5-6) e P2=(2-1-6-3-4-5), a tabela seria da seguinte forma:

- Adjacentes a 1: (-2, -6)
- Adjacentes a 2: (-1, 3, 5)
- Adjacentes a 3: (2, -4, 6)
- Adjacentes a 4: (-3, -5)
- Adjacentes a 5: (2, -4, 6)
- Adjacentes a 6: (-1, 3, 5)

Com a recombinação de arestas, pode acontecer o problema de cidades ficarem sem uma aresta para continuidade. Estas cidades ficam então isoladas, e uma nova aresta deve ser introduzida, causando uma mutação indesejável. Usando a tabela de arestas é possível escolher as próximas cidades de forma que aquelas que tiverem a menor quantidade de arestas não usadas têm prioridade de ser a próxima escolhida (WHITLEY; STARKWEATHER; SHANER, 1991, p.3).

Desta forma, na hora de escolher a próxima cidade a ser incluída no caminho do novo indivíduo, dentre as possíveis próximas cidades (as que fazem adjacência

com a cidade atual nos pais), é escolhida a que possui menos cidades adjacentes remanescentes, a não ser que haja alguma aresta que é comum aos pais (representada pelo valor negativo).

2.1.2.2 OPERADOR GULOSO

O operador guloso usa conhecimento do problema para realizar o cruzamento. O objetivo é gerar um indivíduo que tenha uma avaliação melhor que os pais. Para isso, o operador escolhe as próximas cidades de forma gulosa, ignorando a quantidade de adjacências e escolhendo a cidade mais próxima dentre as que são adjacentes aos pais.

Esse operador segue o princípio de que o indivíduo deve conter somente arestas que estejam presentes nos pais, mas não atende à respeitabilidade.

2.1.3 MUTAÇÃO

O operador de mutação é responsável por provocar alterações genéticas nos indivíduos com o intuito principal de diversificar a população. Dentre as opções possíveis, o operador de mutação utilizado foi o algoritmo 2-opt, usado em Brocki (2007), que consiste em inverter parte do caminho, o que pode remover ciclos, diminuindo a distância total.

Guedes, Leite e Aloise (2005) sugerem como alternativa uma técnica denominada infecção viral, para substituir a mutação, podendo apresentar bons resultados para os algoritmos genéticos, que será avaliada futuramente.

2.2 REDES NEURAIS

Uma rede neural é uma representação artificial do cérebro humano que tenta simular seu processo de aprendizagem (HAYKIN, 1999). As redes neurais são compostas por neurônios artificiais, que têm um funcionamento similar a um neurônio humano, gerando uma saída para uma determinada entrada e modificando suas sinapses entre os demais neurônios para adaptar aos estímulos recebidos.

As redes neurais são úteis para problemas como associação e classificação de padrões, detecção de regularidades, processamento de imagens, análise de fala, problemas de otimização, processamento de entradas incompletas ou imprecisas, dentre outros (FRÖHLICH, 2004).

As redes neurais possuem camadas de neurônios, que, ao receberem uma informação, geram uma saída e essas camadas são conectadas. Essas conexões são representadas por uma matriz de pesos, que são ajustados durante o treinamento da rede. Normalmente, uma rede é treinada antes de ser utilizada para o problema. Segundo Braga, Carvalho e Ludermir (2007), diversos métodos para treinamento de redes foram desenvolvidos, mas podem ser agrupados em dois paradigmas principais: aprendizado supervisionado e aprendizado não supervisionado.

No treinamento supervisionado, a cada iteração uma entrada, cuja saída desejável seja conhecida, é apresentada à rede e a saída obtida é comparada com essa saída desejada. Em função disso a matriz de pesos da rede é atualizada.

Redes neurais não supervisionadas, que aprendem sem supervisão, não possuem saídas esperadas. Durante o processo de aprendizagem, os valores dos pesos da rede são organizados dentro de certa faixa, dependendo dos valores de entrada apresentados (BRAGA; CARVALHO; LUDERMIR, 2007). O objetivo é agrupar unidades similares próximas em certas áreas da faixa de valores. Esse efeito pode ser usado com eficiência para classificação de padrões (FRÖHLICH, 2004).

As redes neurais são empregadas ao PCV pelo fato de convergirem gradualmente aos nós (cidades), através do treinamento, tendo várias técnicas e tipos de rede que podem ser usadas.

Um tipo interessante de rede neural, que será citado neste trabalho é a rede neural auto-organizável, comumente chamado de mapa de Kohonen (HAYKIN, 1999). Esse tipo de rede usa o aprendizado não supervisionado, pois os pesos dos neurônios são mudados quando uma entrada é apresentada à rede, sem a necessidade de uma função de ajuste.

O desempenho da rede neural depende de certos parâmetros, onde cada rede pode possuir alguns parâmetros específicos. Os mais comuns são a velocidade de aprendizado e a função de ajuste dos pesos dos neurônios.

É comumente empregada alguma versão simplificada do algoritmo Lin-Kernighan (geralmente o 2-opt) para melhorar o resultado final obtido pela rede neural (BROCKI, 2007).

2.2.1 MAPA DE KOHONEN (SOM)

O mapa auto-organizável, do inglês *Self Organizing Map* (SOM), consiste numa rede neural com aprendizagem não supervisionada, onde a função de ajuste dos pesos da rede neural usa a própria entrada para ajustar os pesos dos neurônios.

Em uma rede neural auto-organizável, os neurônios se organizam sem grupos, de acordo com informações que recebem. Essas informações não são recebidas somente por um único neurônio, mas pela influência dos outros que estão próximos.

Essa auto-organização, que acontece no mapa de Kohonen, forma uma espécie de mapa, onde os neurônios com funções similares ficam agrupados (FIG. 1).

Os neurônios da entrada se conectam com todos os neurônios do mapa. A matriz de pesos resultante propaga os valores da entrada para os neurônios do mapa.

Como os neurônios estão interconectados, o neurônio com maior ativação propaga um feedback para os neurônios vizinhos, ajudando a agrupar os neurônios com posições (pesos) similares. A ativação do neurônio é obtida pelo produto escalar entre a entrada e os pesos do neurônio (FRÖHLICH, 2004).

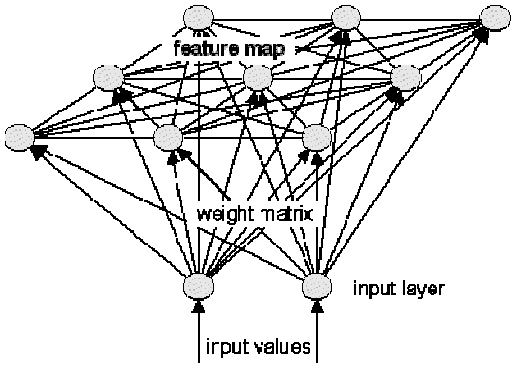

FIGURA 1. Arquitetura do SOM
FONTE: FRÖHLICH, 2004

Brocki (2007) observou que essa rede tenta organizar seus neurônios de uma forma que forme uma rota pequena entre os neurônios, sugerindo que uma rede unidimensional de Kohonen seria útil para resolver o PCV. O estudo proposto por ele consiste numa rede SOM unidimensional, com um neurônio para cada cidade, e os pesos do neurônio representam suas coordenadas, sendo que após o treinamento da rede cada neurônio terá os pesos representando as coordenadas de uma cidade, formando assim um caminho pelo mapa de cidades, ou seja, uma solução para o PCV.

Após o treinamento do SOM, o resultado obtido deve ser melhorado usando-se o algoritmo 2-opt (BROCKI, 2007). Esse algoritmo tem a capacidade de remover ciclos do caminho encontrado por inverter uma parte do circuito, proporcionando boa chance de melhorar a solução encontrada pelo mapa de Kohonen.

Brocki (2007) sugere que esses parâmetros sejam definidos em função do número de cidades em questão.

2.3 LIN-KERNIGHAN

É um método heurístico proposto por Lin e Kernighan em 1971, apresentado em Helsgaun (2000) e que, ao longo dos anos, foi aperfeiçoado por pesquisadores. O algoritmo original tinha ordem exponencial.

O algoritmo usa o conceito da otimalidade, e um caminho é dito λ-ótimo se é impossível melhorá-lo trocando λ arestas do mesmo por outro conjunto de λ arestas (HELSGAUN, 2000).

Através de sucessivas trocas aleatórias de arestas em um caminho, tende-se a melhorar a solução (diminuindo-se a distância). É um algoritmo leve, de baixo custo computacional, comumente usado combinado com as técnicas de IA (como algoritmo de busca local e otimização), que são avaliadas neste trabalho.

Os resultados obtidos pelo algoritmo, que são apresentados por Brocki (2007) inspiraram a implementação do mesmo para comparação neste trabalho.

3 METODOLOGIA

Para o desenvolvimento dos experimentos propostos neste trabalho, foi escolhida a plataforma Java para programação dos algoritmos genéticos e das redes neurais.

Para comparar os algoritmos foi utilizada a biblioteca de mapas TSPLIB (*Traveling Salesman Problem Library*), usada nos estudos referenciados neste artigo. A TSPLIB é uma biblioteca que contém diversos mapas prontos com a solução ótima para cada caso. Cada mapa dessa biblioteca é um arquivo texto com extensão *.tsp*, e contém dados do mapa, como nome, quantidade de cidades, tipo dos dados das cidades. Foram escolhidos mapas 2D, que apresentam cada cidade com um conjunto de coordenadas *x* e *y*, usado para a montagem do mapa. Dentre as dezenas de mapas disponíveis na biblioteca, foram selecionados três mapas de tamanhos distintos.

Dentre as técnicas de IA avaliadas, foram comparados os algoritmos genéticos, as redes neurais auto-organizáveis de Kohonen e o algoritmo heurístico Lin-Kernighan.

3.1 ALGORITMOS GENÉTICOS

A programação da estrutura básica do algoritmo genético foi feita conforme o algoritmo mostrado na seção 2.1.

Inicialmente uma população foi gerada, formando indivíduos com cidades escolhidas aleatoriamente e adicionando na lista da população até que o tamanho máximo (pré-estabelecido) seja atingido. Inicia-se então o ciclo, repetindo as fases de seleção, cruzamento e mutação até que o limite de tempo seja atingido.

Foram implementados três métodos de cruzamento: o operador EER, o operador guloso e um cruzamento aleatório. Os três métodos usam os mesmos parâmetros de entrada (os dois indivíduos pais), fazem o cruzamento e retornam um indivíduo filho.

Para o EER, uma tabela (*Hashtable*) é gerada, percorrendo o caminho dos pais, as cidades adjacentes são adicionadas uma no registro da outra.

Construída a tabela, a recombinação é iniciada pela cidade inicial de um dos pais. São removidas todas as ocorrências da cidade na tabela. Da lista de cidades adjacentes a ela, é escolhida a que tiver a menor lista de adjacentes, dando preferência se alguma cidade estiver representada por valor negativo (obedecendo à respeitabilidade). Repete-se o ciclo até que todas as cidades tenham sido escolhidas.

Caso o programa chegue a alguma cidade que não tem nenhuma cidade adjacente restante, ele continua a partir de outra cidade selecionada aleatoriamente.

Para o operador guloso, o programa também inicia de uma cidade inicial de um dos pais. Para a próxima cidade, procura-se nos pais, quais cidades são adjacentes a atual, e é escolhida a que tiver menor distância à cidade atual, repetindo-se o ciclo.

Para o operador aleatório, a sequência é similar ao do guloso, porém uma das quatro cidades adjacentes à atual nos pais (a da esquerda e da direita em cada indivíduo) é escolhida de forma aleatória.

Após completar a sequência, o indivíduo é criado e retornado para o programa principal. O cruzamento é repetido até que a população tenha atingido o seu tamanho máximo.

Após o cruzamento, a mutação é aplicada, escolhendo-se aleatoriamente indivíduos e aplicando a mutação, o que gera novos indivíduos que são adicionados à população. Os piores serão removidos na próxima seleção. A mutação é aplicada em um percentual da população, pré-estabelecido em 5%, conforme sugerido por Starkweather *et al.* (1991).

Como a seleção inicialmente testada foi simplesmente eliminar uma porção dos piores (por exemplo, eliminar 50% dos indivíduos), a lista de indivíduos da população foi mantida ordenada, inserindo sempre um indivíduo na sua posição pela ordem de *fitness*. Desta forma, o indivíduo na posição zero do vetor sempre será o de melhor avaliação, e é mostrada sua *fitness* a cada iteração para acompanhamento da evolução do algoritmo genético. Ao final das iterações, é mostrado o caminho do melhor indivíduo da população.

3.2 REDE NEURAL

A programação da estrutura básica foi implementada conforme a seção 2.2.1, baseando-se na mesma arquitetura do mapa usada para os algoritmos genéticos, porém, apenas um caminho é criado e melhorado gradativamente através do treinamento da rede neural.

Um mapa auto-organizável (SOM) foi criado, onde cada nó possui dois adjacentes, formando uma linha que representa o caminho do caixeiro viajante.

Após a criação da rede SOM com valores aleatórios, inúmeras iterações de treinamento são realizadas, onde as coordenadas de uma cidade aleatória são apresentadas como entrada, e os nós vão se auto-organizando.

Após o treinamento, os pesos dos neurônios da rede podem não coincidir com as coordenadas das cidades. Diante disso, os pesos são ajustados para a cidade mais próxima, desde que não haja nenhum neurônio associado a ela ainda, visto que a relação de neurônio por cidade é de um para um, não pode haver dois neurônios na mesma cidade.

O mapa de Kohonen foi implementado com treinamento WTA (*Winner Takes All*) onde, a cada iteração, somente o neurônio com maior ativação tem seus pesos ajustados, para evitar a acumulação de mais de um neurônio com os mesmos pesos (representando a mesma cidade).

Ao contrário do estudo de Brocki (2007), não foi utilizado um algoritmo otimizador após o treinamento, pois o objetivo é testar somente o desempenho do mapa de Kohonen.

3.3 LIN-KERNIGHAN

Para comparar os algoritmos genéticos e as redes neurais com o algoritmo Lin-Kernighan sob as mesmas condições, foi implementada uma versão simplificada desse algoritmo.

O algoritmo Lin-Kernighan realiza troca de arestas com o intuito de diminuir a distância do percurso, até um limite de trocas por iteração.

A cada iteração uma nova cidade é escolhida aleatoriamente e a troca começa removendo a aresta adjacente à cidade.

O tempo foi estabelecido como condição de parada, sendo que a cada período determinado de tempo, o limite de trocas é aumentado, permitindo uma otimalidade de maior grau.

4 RESULTADOS

Foram realizados dez experimentos com os principais mapas usados, alterando-se alguns parâmetros. Para os algoritmos genéticos, o tamanho da população variou entre 20 e 1.000, a seleção foi por torneio (seleciona metade para a próxima geração), e fator de mutação entre 5% e 20% da população. Para o algoritmo de Lin-Kernighan, o número de trocas variou entre 3 e a metade do total de cidades.

Todas as simulações foram feitas em uma máquina com processador Intel Core 2 Duo™ de 2.0 GHz, através da IDE do NetBeans em um Mac OS X.

Inicialmente serão apresentados os resultados obtidos pela utilização da rede SOM (Seção 4.1). Em seguida, os resultados alcançados com a utilização dos algoritmos genéticos, variando-se os operadores de cruzamento implementados (Seção 4.2). Finalmente, os melhores resultados obtidos com os algoritmos genéticos serão comparados com o desempenho do algoritmo Lin-Kernighan.

4.1 REDE NEURAL

Durante o treinamento da rede neural foi observado que os neurônios vão coincidindo os pesos a cada iteração, fazendo com que estejam aglomerados em dois grupos ao final do treinamento, inviabilizando a formação de um caminho pelas coordenadas dos mesmos.

Para amenizar o problema, foi descartado o feedback do neurônio de maior ativação para os demais, sendo que a cada iteração somente um neurônio tenha seus pesos alterados. Mesmo assim alguns neurônios continuaram a se agrupar e somente alguns eram ativados quando uma cidade era apresentada à entrada da rede.

Ao final do treinamento, o processo de associação de cada neurônio com uma cidade não gerou uma boa solução, devido ao agrupamento dos neurônios.

4.2 ALGORITMOS GENÉTICOS

Os resultados obtidos com os experimentos envolvendo os algoritmos genéticos são sintetizados na Tabela 1 e discutidos com detalhes a seguir.

Os melhores resultados para o algoritmo genético foram obtidos com uma população de 1.000 indivíduos e taxa de mutação de 10%.

Durante os testes foi possível observar que para populações pequenas, um número maior de gerações não apresentava melhora na solução, no entanto, o algoritmo converge em menos tempo. Em função disso, adotou-se também como critério de parada, o máximo de 20 gerações sem alteração do resultado do melhor indivíduo (além do limite da quantidade de gerações pré-estabelecido).

Para mapas menores, o EER obteve resultados melhores, enquanto que para mapas maiores foi necessário mais tempo de execução para obter resultados melhores.

O operador de cruzamento guloso tende a convergir mais rápido, mas à medida que as gerações aumentam, leva-se mais tempo para gerar a nova população, visto que os indivíduos obtidos tendem a ficar repetidos, conforme a população vai se tornando mais homogênea. Já o operador de cruzamento aleatório apresentou resultados muito ruins, se comparados aos demais.

Na Tabela 1, para cada mapa utilizado são apresentados o caminho ótimo, informado pela biblioteca TSPLIB, e os caminhos médios obtidos pelos algoritmos. Entre parênteses é apresentado o tempo de execução médio, medido em segundos.

TABELA 1

RESULTADOS OBTIDOS NA EXECUÇÃO DOS ALGORITMOS GENÉTICOS

Mapa	Ótimo	EER	Aleatório	Guloso
eil51	426	**438** (63)	982 (105)	445 (43)
tsp225	3916	11327 (122)	35193 (50)	**4221** (60)
pr2392	37803 2	11,8 $.10^6$ (462)	13,9 $.10^6$ (892)	**438991** (161)

É possível observar que para mapas pequenos, o operador EER apresentou o caminho mais próximo do ótimo. No entanto, para os demais mapas, o operador de cruzamento guloso apresentou os melhores resultados, apesar de estarem muito distantes do caminho ótimo. Além disso, vale destacar que o operador guloso tende a convergir mais rápido, como pode ser observado na Tabela 1.

Com isso, o operador de cruzamento guloso mostrou ser mais eficiente que o EER e o aleatório, apresentando boas soluções em tempos muito menores.

4.3 LIN-KERNIGHAN

A versão implementada do algoritmo Lin-Kernighan foi testada com os mesmos mapas usados nos experimentos com os operadores de cruzamento dos algoritmos genéticos.

Foi usado como parâmetro um limite de trocas que, inicialmente, foi estabelecido para 40% do número de cidades do mapa. Além disso, a cada segundo esse limite era aumentado em 1, sendo que o máximo de trocas possíveis foi ajustado para a metade do número de cidades.

Na Tabela 2, são apresentados os caminhos obtidos e os tempos médios de execução (em segundos) tanto do algoritmo Lin-Kernighan, como do algoritmo genético com o operador guloso.

TABELA 2

COMPARAÇÃO LIN-KERNIGHAN X ALGORITMO GENÉTICO

Mapa	Ótimo	LK	AG-Guloso
eil51	426	449 (2)	**445** (43)
tsp225	3916	5236 (10)	**4221** (60)
pr2392	378032	703950 (142)	**438991** (161)

O algoritmo Lin-Kernighan obteve soluções piores que do algoritmo genético, porém com tempo de execução menor.

Apesar de encontrar uma solução razoável em pouco tempo, ele tende a estabilizar em um ótimo local. Ou seja, após encontrar essa solução, ele não consegue escapar desse ótimo local e melhorar a solução, mesmo que lhe sejam dadas mais iterações.

Na Figura 2 são apresentados os erros relativos médios obtidos pelos algoritmos analisados. Nota-se que os algoritmos genéticos com operador de cruzamento guloso e o algoritmo Lin-Kernighan apresentaram erros bem menores que os demais.

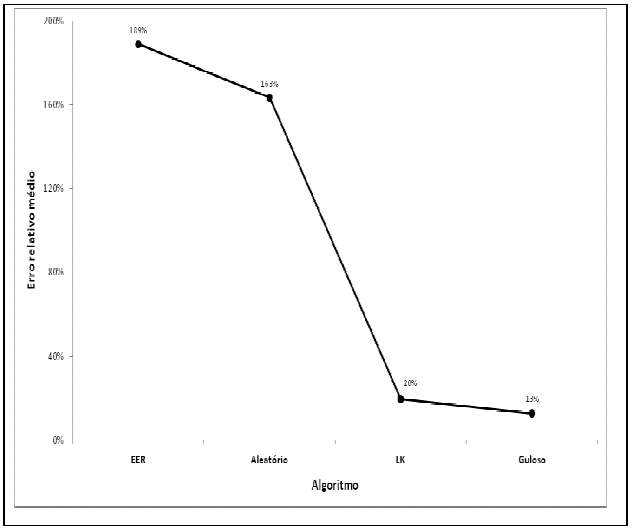

FIGURA 2. Erro relativo médio obtido pelos algoritmos analisados

Diante disso, e com o intuito de avaliar a variabilidade dos resultados obtidos, definiu-se um índice de

eficiência como o produto do tempo de execução e o erro relativo obtido.

TABELA 3

TEMPO DE EXECUÇÃO MÉDIO E ÍNDICE DE EFICIÊNCIA DOS ALGORITMOS AVALIADOS

Algoritmo	Tempo de execução médio	Índice de eficiência
EER	*216*	*4731*
Aleatório	*349*	*10814*
Guloso	*88*	***11***
LK	***51***	*42*

Na Tabela 3 são apresentados os tempos de execução médios, obtidos pelos algoritmos analisados e os respectivos índices médios de eficiência. É possível observar que o algoritmo Lin-Kernighan apresentou os menores tempos de execução para todos os mapas utilizados. No entanto, ao analisar o índice de eficiência, o algoritmo genético com operador de cruzamento guloso continuou apresentando os melhores resultados.

Diante dos resultados obtidos, uma alternativa é usar o algoritmo Lin-Kernighan para melhorar os indivíduos do algoritmo genético, o que não foi avaliado no escopo deste trabalho. Como o algoritmo converge rapidamente para um ótimo local, pode-se executá-lo várias vezes, usando cada resultado como um indivíduo da população inicial do AG, que através do cruzamento entre as soluções encontradas pode sair dos ótimos locais e encontrar a solução ótima para o problema.

5 CONCLUSÃO

A rede auto-organizável não obteve êxito ao gerar uma solução para o PCV. Como os neurônios se acumulam no mesmo ponto, após associar cada neurônio com a cidade mais próxima, muitos teriam que ser associados com uma cidade aleatória, pois a cidade mais próxima já estaria ocupada por outro neurônio coincidente. Isso invalidaria todo o treinamento da rede, gerando ao final do processo uma solução aleatória.

Além disso, e baseando-se nos resultados obtidos, é possível concluir que o algoritmo genético com o operador de cruzamento guloso apresentou desempenho superior ao observado no EER e no algoritmo Lin-Kernighan implementados neste estudo.

Vale ressaltar que, Whitley (1991), em seu trabalho, apresentou resultados melhores que os observados no presente artigo: o EER encontrou solução ótima na maioria dos casos. No entanto, sabe-se que os resultados obtidos pelos algoritmos genéticos dependem dos operadores de seleção e mutação utilizados.

REFERÊNCIAS

BRAGA, A. P., CARVALHO, A. P. L. F., LUDERMIR, T. B., **Redes Neurais Artificiais: Teoria e Aplicações,** Editora LTC, 2. ed., 226 p., 2007.

BROCKI, Lucas. **Kohonen Self-Organizing Map for the Traveling Salesperson Problem.** Berlin: Recent Advances in Mechatronics, Springer, 2007.

DORIGO, Marco. GAMBARDELLA, Luca Maria. **Ant colonies for the traveling salesman problem.** Bruxelas: IRIDIA, Université Libre de Bruxelles, IEEE Transactions on Evolutionary Computation, 1(1):53–66, 1997.

FRÖHLICH, Jochen. **Neural Networks with Java.** Disponível em: http://www.nnwj.de (versão 2004). Acesso em: mar. 2010.

GUEDES, Allison da Costa Batista. LEITE, Jéssica Neiva de Figueiredo. ALOISE, Dario José. **Um algoritmo genético com infecção viral para o problema do caixeiro viajante.** Natal [Brasil]:

Revista On Line de Iniciação Científica, 2005. Disponível em: http://www.propesq.ufrn.br/publica/artigos-1edicao/et/MSIC-ET-011.pdf, Acesso em: fev. 2010.

HAYKIN, S. S., **Neural Networks: A Comprehensive Foundation**, Editora Prentice-Hall, 2. ed., 842 p., 1999.

HELSGAUN, Keld. **An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic.** European Journal of Operational Research 126 (1), p. 106-130, Elsevier, 2000.

MERZ, Peter. **A Comparison of Memetic Recombination Operators for the Traveling Salesman Problem.** GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann Publishers, 2002.

STARKWEATHER, T., et al. **A Comparison Of Genetic Sequencing Operators.** San Diego: International Conference On Gas, 1991.

TSPLIB - ***traveling salesman problem library*****,** Biblioteca de mapas, disponível em comopt.ifi.uni-heidelberg.de/software/ TSPLIB95/, acesso em: mar. 2010.

WHITLEY, Darrell. STARKWEATHER, Timothy. SHANER, **Daniel. Traveling Salesman And Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination.** Handbook Of Genetic Algorithms. New York: Van Nostrand Reinhold, 1991.