Um algoritmo memético para a solução do TSP

Tayná Costa Gonçalves[1], Roberto Leite de M. R. Filho[1], Omar Andres Carmona Cortez[2]

[1]Alunos do Curso de Sistemas de Informação - IFMA e-mail: fulanodetal@ifto.edu.br
[2]. e-mail: omar@ifma.edu.br

Resumo: O problema do caixeiro viajante é um problema clássico de otimização combinatorial, cuja resolução por busca exaustiva é inviável em alguns casos, como por exemplo, se considerarmos um número grande de cidades. Por ser aplicável a muitos problemas da vida real, o TSP é amplamente estudado, buscando-se abordagens mais eficiente para solucioná-lo. Neste trabalho é proposto um algoritmo memético para solução do TSP. O objetivo é mostrar que o uso de tal algoritmo possibilita encontrar uma solução satisfatório para uma instância do TSP em um tempo aceitável. O algoritmo memético se mostra adequado nessa circunstância pois combina operadores genéticos tradicionais e operadores de busca local, aumentando assim a qualidade das soluções. Serão apresentadas comparações entre diferentes combinações de representações de indivíduos e operadores. Ainda, no intuito de comprovar a eficiência do algoritmo proposto, serão apresentados resultados de testes que utilizam uma função de benchmark da conhecida biblioteca TSPLIB.

Palavras–chave: Algoritmo Memetico, Busca Local, Operadores Genéticos, TSP

1. INTRODUÇÃO

Algoritmos evolutivos são bastante difundidos e estudados, e geralmente são aplicados a problemas difíceis de solucionar utilizando outras abordagens. As aplicações são muitas, pois, apesar de contarem com uma estrutura básica definida na teoria, em geral é possível adaptar um algoritmo evolutivo às necessidades de um problema específico, variando apenas os operadores e representações utilizados.

Um exemplo de problema que exige atenção especial é o problema do caixeiro viajante (TSP), onde dado um número *n* de cidades, um caixeiro precisa visitar cada uma delas uma única vez e retornar à cidade de origem. Trata-se de um problema simples de definir, porém de difícil solução, dada a explosão combinatorial gerada pelo aumento do número de cidades a serem visitadas. O TSP é, portanto, um importante problema de otimização combinatorial. Sua aplicabilidade em problemas da vida real estimula as pesquisas no sentido de encontrar melhores abordagens para solucioná-lo.

Em diversos trabalhos a computação evolutiva tem sido aplicada à resolução de problemas de otimização discreta. Rabak & Sichman (2001) modelaram o problema da inserção automática de componentes eletrônicos em placas de circuito impresso como um problema de otimização combinatorial, e propuseram um arranjo de algoritmos que utiliza os conceitos de agentes e memórias, conceitos pertencentes ao mundo dos Algoritmos Meméticos. Becceneri (2012) utiliza uma metaheurística, no caso a colônia de formigas aplicada ao TSP. Merz & Freisleben (1997) revisaram a aplicação ao TSP de heurísticas de busca local associadas a algoritmos genéticos, no intuito de melhorar alguns aspectos dessa combinação.

As heurísticas de busca local associadas a outras técnicas melhoram a qualidade das soluções, porém podem aumentar significativamente o tempo de execução para problemas de maior escala (Arshad 2009). O algoritmo memético proposto busca encontrar o equilíbrio entre soluções próximas ao ótimo e tempo de execução aceitáveis

2. MATERIAL E MÉTODOS

2.1. ALGORITMOS GENÉTICOS

ISBN 978-85-62830-10-5

Os algoritmos genéticos (Holland 1992) possuem características que os tornam especialmente adequados para tratar problemas de otimização, principalmente nos casos em que o uso de métodos determinísticos se mostra inviável computacionalmente. Basicamente o funcionamento de um AG consiste das seguintes etapas: criação de uma população inicial de soluções, avaliação da aptidão desses indivíduos, aplicação dos operadores genéticos (tradicionalmente recombinação e mutação), nova avaliação dos indivíduos e inserção dos melhores indivíduos na nova população.

Espera-se atingir dois objetivos com o uso dos operadores genéticos. Primeiramente explorar o espaço de busca, já que não é desejável que o algoritmo fique preso em um ótimo local. Ao alcançar uma solução de qualidade, é importante que ela seja mantida e melhorada ao longo da evolução. Para que esses objetivos aparentemente conflitantes se harmonizem é necessário escolher, dentre uma gama de opções, a representação da solução, os operadores genéticos e os parâmetros mais adequados a cada problema. Um Algoritmo Genético nem sempre irá encontrar o ótimo global de um problema, porém, se comparado a métodos exatos, que na maioria dos casos tem custo altíssimo para problemas muito complexos, torna-se aceitável uma solução próxima ao ótimo.

2.2 ALGORITMOS MEMÉTICOS

A idéia chave por trás de um algoritmo memético é combinar conceitos e estratégias de diferentes metaheurísticas com o objetivo de aproveitar os pontos positivos das mesmas. Alguns conceitos dos algoritmos genéticos são aproveitados, como por exemplo, a sua natureza populacional. Em ambos é mantida uma população de possíveis soluções, que é atualizada a cada geração.

De acordo com Neri & Cotta (2012) “Algoritmos meméticos são metaheurísticas baseadas em população compostas de um framework evolutivo e um conjunto de algoritmos de busca local que são ativados de acordo com o ciclo de gerações do framework externo." Analogamente aos algoritmos genéticos, que transmitem genes como unidade de informação, os algoritmos meméticos mantém e transmitem memes de geração para geração. Memes podem ser definidos como unidades de informação cultural, que podem ser transmitidos por imitação através das gerações.

O mecanismo que permite a transmissão da informação memética é a introdução de um (ou mais) operador de busca local, que também pode ser considerando um meta-operador dependendo da maneira como é inserido no algoritmo. O uso de operadores de busca local é a característica mais marcante do algoritmo memético (Moscato 2003). Por existirem diferentes maneiras de inserir a busca local no algoritmo, há uma grande variabilidade nos algoritmos meméticos. Dependendo do seu posicionamento, a busca local afeta de maneira diferente os resultados. Por exemplo, em Ghoseiril & Sarhadi (2008) a busca local é aplicada em dois pontos: após a criação da população inicial e após o cruzamento.

2.3 OTIMIZAÇÃO DISCRETA E O PROBLEMA DO CAIXEIRO VIAJANTE

Problemas de otimização consistem em encontrar a melhor combinação de variáveis, dentro de um domínio, para maximizar ou minimizar uma função. Esta função, geralmente chamada de função objetivo ou função de custo, constitui o ponto de partida das abordagens de solução para problemas de otimização. Nos AG's esta função corresponde à função de aptidão.

Nos problemas de otimização as variáreis podem assumir três tipos de valores: contínuos, discretos ou ambos. Associado a cada tipo de valor assumido temos um tipo de otimização: Otimização Contínua, Otimização Combinatória ou Discreta, e Otimização Mista, respectivamente (Becceneri, 2012).Na Otimização Discreta é possível que a solução ótima seja encontrada através de busca exaustiva, método que envolve contar e avaliar todas as possíveis soluções. Porém, por demandar um esforço demasiadamente grande, esta abordagem se torna inviável para a maioria dos problemas. O uso de meta-heurísticas se apresenta como uma opção atraente para este cenário.

Neste trabalho exploramos o TSP, um problema clássico de otimização discreta, amplamente estudado e bastante utilizado como referência para testar a qualidade de soluções propostas para

ISBN 978-85-62830-10-5

problemas de otimização. É um problema de fácil definição, porém difícil de solucionar e possui uma vasta aplicabilidade prática. De maneira geral pode ser definido como segue: Dadas *n* cidades, um caixeiro necessita visitar cada uma delas uma única vez, começando de qualquer uma delas e retornando à cidade inicial. A solução ótima seria uma rota que minimize a distância total percorrida (Shen 1965). Matematicamente, temos: dado um inteiro $n \geq 3$e uma matriz $C\ n\ x\ n = (c_{ij})$, onde cada c_{ij} é um inteiro não negativo. O problema consiste em determinar qual permutação cíclica π dos inteiros de 1 a *n* minimiza a soma:

$$\sum_{i}^{n} = 1^{c_{i\pi(i)}}$$

3. RESULTADOS E DISCUSSÃO

3.1 ALGORITMO PROPOSTO

Neste trabalho é proposto um algoritmo memético aplicado ao TSP. Neste algoritmo é aplicada a busca local em dois pontos. No primeiro ponto ela será executada em todas as iterações enquanto que no segundo ponto ela será aplicada de acordo com uma probabilidade de 10%. O objetivo de condicionar a execução da busca local a uma probabilidade é diminuir o custo computacional do algoritmo como um todo, sem perder o refinamento proporcionado pela metaheurística. A seguir serão detalhados os mecanismos utilizados no algoritmo.

3.1.1 A REPRESENTAÇÃO DOS INDIVÍDUOS

A representação escolhida para a solução é um fator decisivo para a qualidade de um algoritmo aplicado ao TSP. Foram implementados dois tipos de representação, a representação ordinal e a representação por caminho. Iremos explicar brevemente como cada uma delas funciona.

Na representação por caminho o cromossomo é um vetor de inteiros, cada um representando uma cidade dentro de um caminho completo, na ordem em aparecem no vetor. Esta é a representação mais natural para uma solução do TSP, porém apresenta restrições com relação aos operadores genéticos que podem ser diretamente aplicados a ela. Operações genéticas tendem a gerar cromossomos infactíveis nesse tipo de representação. O cruzamento simples, por exemplo, não pode ser aplicado pois tem altas chances de gerar descendentes infactíveis. É possível aplicar um tratamento de reparação a esses descendentes infactíveis, porém é necessário avaliar se o custo computacional agregado compensa. Já estão disponíveis na literatura diversas variações dos operadores genéticos tradicionais que se aplicam à representação de caminho, garantindo uma descendência válida (Larranaga et al 2009). Em geral, todos trazem embutidos alguma maneira de evitar a infactibilidade da prole.

Na representação ordinal o cromossomo é definido através da sua posição relativa dentro de um caminho completo. Para tanto é necessário ter um caminho completo válido de tamanho *n* e um vetor ordenado também de tamanho *n*. A partir destes dois elementos será criado o cromossomo. A primeira posição do cromossomo é o valor do índice, no vetor de referência, da primeira cidade do caminho. Após isso, a cidade em questão é retirada do vetor de referência. A segunda posição do cromossomo é, novamente, o valor do índice, no vetor de referência, da segunda cidade do caminho. Novamente a cidade em questão é retirada do vetor de referência. O procedimento se repete até que o vetor de referência esteja vazio e o cromossomo tenha o mesmo número de elementos que a representação de caminho usada no início.

ISBN 978-85-62830-10-5

VETOR DE REFERÊNCIA: (1 2 3 4 5 6 7 8)

CAMINHO: (1 5 3 2 8 4 7 6)

INDIVÍDUO: (1 4 2 1 4 1 2 1)

3.1.2 SELEÇÃO

Foram testadas a seleção por torneio e a seleção por roleta. Na seleção por torneio, *n* indivíduos são escolhidos aleatoriamente dentro da população atual e entre eles é realizado um torneio, baseado aptidão dos indivíduos. No algoritmo proposto, é realizado um torneio para cada indivíduo a ser submetido ao cruzamento.

Na seleção por roleta (Goldberg 1989) para cada indivíduos é atribuída uma probabilidade de seleção que representa uma fatia de uma roleta. Quanto maior a aptidão maior a fatia atribuída, portanto indivíduos mais aptos têm mais chances de serem selecionados. Para realização dos testes apresentados neste artigo foi usada a seleção por torneio, visto que apresentou um tempo de execução menor e resultados similares à seleção por roleta.

3.1.3 CRUZAMENTO

Em um algoritmo para o TSP há uma forte ligação entre a representação de solução utilizada e seus operadores genéticos. Esta combinação pode determinar a geração de soluções factíveis. Neste trabalho foram testadas duas abordagens: o cruzamento simples e o cruzamento PMX.

Apesar de agregar certa complexidade à criação dos cromossomos, a representação ordinal é atrativa por suportar o cruzamento simples. É escolhido um ponto de corte, que divide cada pai em duas partes. O primeiro filho recebe a primeira parte do pai 1e a segunda parte do pai 2. O segundo filho recebe os elementos restantes dos pais. Este tipo de cruzamento permite manter possíveis subcaminhos promissores.

O cruzamento PMX está associado à representação por caminho e prima por manter a factibilidade da prole. Inicialmente, os genes dos pais que estão no intervalo entre dois pontos de corte são mapeados. O filho 1 recebe o subcaminho intermediário do pai 2 e o filho 2 recebe o subcaminho intermediário do pai 1. Após, o filho 1 recebe o restante dos genes do pai 1, porém, caso alguma cidade já tenha sido transmitida pelo subcaminho do pai 2, esta será substituída pelo mapeamento feito no início. O mesmo processo é feito com o filho 2, recebendo o restante dos genes no pai 2.

PAI 1: (1 2 3 4 5 6 7 8)
PAI 2: (3 7 5 1 6 8 2 4)

PA1 1: (1 2 3 | 4 5 6 | 7 8)
PA1 2: (3 7 5 | 1 6 8 | 2 4)

FILHO 1: (x x x | 1 6 8 | x x)
FILHO 2:(x x x | 4 5 6 | x x)

APÓS A SUBSTITUIÇÃO:

FILHO 1: (4 2 3 | 1 6 8 | 7 5)
FILHO2: (3 7 8 | 4 5 6 | 2 1)

3.1.4 MUTAÇÃO

ISBN 978-85-62830-10-5

Foi utilizada a mutação de troca (Larranaga et al 2009), por ser de simples implementação, baixo custo computacional, e aplicável aos dois tipos de representações utilizadas. O mecanimso consiste em escolher aleatoriamente dois genes no cromossomo e inverter suas posições.

3.1.5 BUSCA LOCAL

A técnica de busca local utilizada no algoritmo memético é o Simulated Annealing, cuja metáfora é o processo físico de recozimento, no qual um material é submetido a variações bruscas de temperatura para melhorar a qualidade de algumas das suas propriedades. Movimentos que resultem em soluções de pior qualidade que a da solução corrente são permitidos, tornando possível sair de mínimos locais, porém sem se afastar demais de uma boa solução já encontrada (Gao & Jia 2009).

A busca local é o elemento que mais agrega custo computacional ao algoritmo memético, porém é o elemento que melhora a qualidade da solução. Torna-se necessário encontrar os parâmetros utilizados que equilibrem a qualidade das soluções e o custo agregado. Para minimizar este custo busca local não é aplicada ao cromossomo todo, somente em parte dele. É escolhido aleatoriamente um ponto *i* dentro do cromossomo. Após este procedimento a busca local tenta substituir a posição *i+1* por outro gene do próprio cromossomo a fim de melhorar o fitness neste subvetor e, possivelmente, em todo o indivíduo.

3.2 TESTES E RESULTADOS

Foram conduzidos dois tipos de testes. O algoritmo memético proposto foi comparado a um algoritmo genético tradicional e o algoritmo memético com representação ordinal foi comparado à representação de caminho. Os experimentos foram conduzidos em um computador Intel Core i3-330M 2.13GHz, 4GB (RAM) e 500GB(HD). O algoritmo foi implementado em Java, utilizando Eclipse IDE versão 3.6.2 e Java versão 1.6.

Para cada teste a função de benchmark foi resolvida 31 vezes, sendo esse número baseado na teoria do limite central, que afirma que, com 30 ou mais execuções, uma amostra apresenta uma distribuição normal, permitindo inferências estatísticas (Moscato 2003), como o teste t que será usado nesse artigo. Na Tabela 1 são apresentados os parâmetros utilizados para os testes.

Tabela 1: Parâmetros utilizados para os testes.

Parâmetro	Valor
Tamanho da população	100
Taxa de Cruzamento	90
Taxa de mutação	15
Número de Iterações	3000

No teste t utilizado a hipótese nula é de que não há diferença entre os grupos, i.e., H_0: $\mu_1 = \mu_2$. Foi utilizado nível de confiança $\alpha = 0.5$ bicaudal, e, pela tabela de distribuição *t* de Student, tem-se o intervalo [-2.042, 2.042]. Portanto, qualquer resultado de *t* calculado pela equação abaixo, deve estar deste intervalo para que a hipótese nula seja aceita, onde μ_i são desvios padrão e *n* é o número de execuções.

ISBN 978-85-62830-10-5

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{n} + \frac{\sigma_2^2}{n}}}$$

Para avaliar o algoritmo proposto, foi utilizada uma das funções de benchmark da biblioteca TSPLIB (Reinelt 1991), o problema de 51 cidades EIL51. O menor caminho conhecido para esse problema custa 426. Este problema é um exemplo de TSP simétrico, pois a distância (custo), de um nó *i* para um nó *j* é a mesma de *j* para *i*, ou seja, $d_{ij} = d_{ji}$ para $\forall$ i,j $\in$ {1, 2,..., n}. No problema eil51 as cidades estão posicionadas em termos do espaço euclidiano, ou seja, cada cidade é representada por um par de coordenadas x e y, e o custo para ir de uma cidade a outra é calculado por:

$$\begin{aligned} xd &= pos_x i - pos_x j; \\ yd &= pos_y i - pos_y j; \\ custo &= round(sqrt(xd^2 + yd^2)); \end{aligned}$$

3.2.1 AG ORDINAL E AG DE CAMINHO

Neste teste foram comparadas as duas representações de indivíduos implementadas. O tempo médio, considerando 31 execuções, cada uma com 3000 iterações é exibido na Tabela 2. Embora demore mais tempo na execução, o algoritmo de representação ordinal não supera a representação de caminho em termos de qualidade da solução, como pode ser observado na Figura 1, que mostra a evolução dos dois algoritmos. Pode-se observar que a convergência do AG de caminho é bem mais rápida e a solução final é melhor que no AG ordinal. Na Tabela 3 podemos ver que, estatisticamente, os resultados são diferentes, sendo que o AG caminho supera o AG ordinal.

Tabela 2: Média do tempo de execução

Implementação	Tempo(s)
Caminho	215,09
Ordinal	439,19

Tabela 3: Média dos resultados AG Ordinal X AG Caminho

	AG Ordinal	AG Caminho	t
Médias	645,16	569	**7,06**
Desvio Padrão	39,09	36,29	

ISBN 978-85-62830-10-5

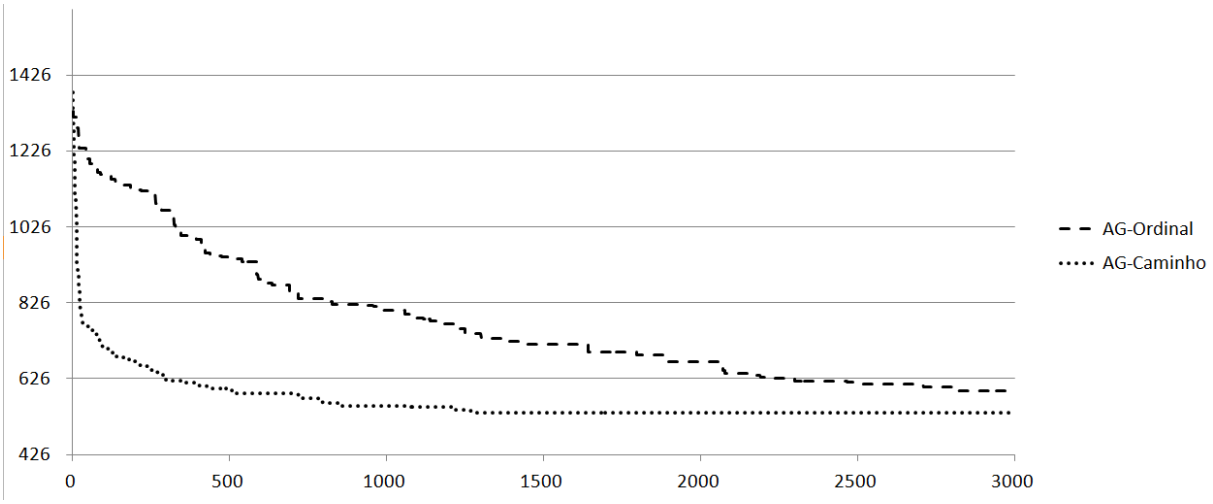

Figura 1: Evolução das representações de caminho e ordinal

3.2.2 AG E AM DE CAMINHO

A Tabela 4 mostra a média dos resultados de execução do AG e do AM, ambos utilizando a representação de caminho. Também é mostrado entre parênteses os seus respectivos desvios padrão. Nas médias o AG se mostra ligeiramente superior ao AM. Para confirmar se há diferença significativa aplicamos o teste estatístico t entre as duas médias. O valor de t calculado se encontra dentro do intervalo de confiança, assim aceitamos a hipótese nula, que diz que não há essa diferença.

Tabela 4: Média dos resultados AG Caminho X AM Caminho

	AG Caminho	AM Caminho	t
Médias	555,45	569	**1,39**
Desvio Padrão	35,17	36,29	

4. CONCLUSÕES

O algoritmo memético apresentado se mostrou ligeiramente inferior, na média, ao algoritmo genético, porém essa diferença não é significativa estatisticamente. Porém também foi constatado que, ao mudar o tipo de representação e os operadores genéticos pode ocorrer uma variação significativa na qualidade das soluções. Portanto, pretende-se testar em trabalhos futuros outros tipos de representação de indivíduo para o TSP, assim como os operadores genéticos relacionados a essa representação, no intuito de comparar a qualidade das soluções produzidas e a performance do algoritmo.

5. AGRADECIMENTOS

Ao CNPq, por ter possibilitado e financiado esta pesquisa

REFERÊNCIAS

ARSHAD, S., YANG, S., LI C. **A Sequence Based Genetic Algorithm with Local Search for the Travelling Salesman Problem.** Department of Computer Science, University of Leicester, Leicester UK, 2009.

ISBN 978-85-62830-10-5

BECCENERI, J. C. **Meta-heurísticas e Otimização Combinatória: Aplicações em Problemas Ambientais** ELAC, 2012

GAO, Y., JIA, S. **Hybrid Differential Evolution Algorithm with Annealing and Chaos.** In: 5th International Conference on Natural Computation: IEEE Computer Society, 2009.

GHOSEIRIL, K., SARHADI, H. **A memetic algorithm for symmetric traveling salesman problem International Journal of Management Science and Engineering Management**. England, UK Vol. 3, 2008.

GOLDBERG, D. E. **Genetic Algorithm in Search, Optimization, and Machine Learning.** Boston: Addison-Wesley Professional, 1989.

HOLLAND, J.H. **Adaptation in Natural and Artificial Systems.** 2 ed., The MIT Press, 1992.

LARRANAGA, P.P., KUIJIPERS, C.M.H., MURGA, R.H., INZA, I. and DIZDAREVIC, S. **Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators.** Artificial Intelligence Review 13, 2009.

LI, Shen. **Computer Solutions of the Traveling Salesman Problem**, manuscript received August 18, 1965.

MERZ, P., FREISLEBEN, B. **Genetic Local Search for the TSP: New Results.** IEEE International Conference on Evolutionary Computation, 1997.

MICHALEWICZ, Z. **Genetic Algorithms + Data Structures = Evolution Programs.** 2 ed, Berlim: Springer Verlag, 1996.

MOSCATO, P., & COTTA, C. **Una Introduccíon a los Algoritmos Meméticos.** Revista Iberoamericana de Inteligencia Artificial, España, 2003.

NERI, F., COTTA, C. **Memetic algorithms and memetic computing optimization: A literature review.** Swarm and Evolutionary Computation, Volume 2, February 2012.

RABAK, C. S., SICHMAN**, J. S. Otimização do processo de inserção automática de componentes eletrônicos empregando a técnica de times assíncronos.** Pesquisa Operacional-39 Vol. 21, No. 1, p. 39-59, junho de 2001.

REINELT, G., **A Traveling Salesman Problem Library,** ORSA Journal on Computer, Vol.3, 1991.

SCHEFLER, W. C. **Statistics: Concept and Applications.** The Benjamin/Cummings Publishing Company, California, 1988.

ISBN 978-85-62830-10-5