ALGORITMOS GENÉTICOS: APLICAÇÃO E MEDIÇÃO DA EFICIÊNCIA DO OPERADOR DE CROSSOVER OX PARA O PROBLEMA DO CAIXEIRO VIAJANTE SIMÉTRICO.

Stenio Anibal Moraes Vital[1]
Felipe Bueno Moreira[2], Roney Lopes Lima[3]
[1]IFG/Jataí/Tecnologia em Análise e Desenvolvimento de Sistemas, anibalstenio@gmail.com
[2] IFG/Jataí/Tecnologia em Análise e Desenvolvimento de Sistemas, felipebn4@gmail.com
[3] IFG/Jataí/Departamento de Áreas Acadêmicas, roney.f10@gmail.com

Resumo

O Problema do Caixeiro Viajante (PCV) é um problema da classe NP-Difícil muito popular no meio científico por modelar vários dos problemas encontrados nas engenharias, biológicas e computação. Por ser extremamente complexo, o PCV para instâncias grandes não pode ser resolvido de forma determinística em tempo computacional viável. Os Algoritmos Evolutivos tais como os Algoritmos Genéticos têm sido aplicados a estes problemas e alcançado resultados interessantes que despertaram o interesse do meio científico. Um dos componentes mais importantes no processo de busca dos Algoritmos Genéticos, vários operadores de cruzamento baseados em permutação estão presentes na literatura. Neste trabalho o operador Crossover de Ordem (OX) tem sua eficiência medida para várias instâncias do PCV da popular biblioteca TSPLIB. Os resultados mostram que o operador possui eficiência muito satisfatória para pequenas instâncias do PCV e relativamente satisfatória para grandes instâncias principalmente quando o número de gerações é aumentado.

Palavras-chave: algoritmos evolutivos, caixeiro viajante, operadores de cruzamento, crossover de ordem, TSPLIB.

INTRODUÇÃO

O Problema do Caixeiro Viajante (PCV ou TSP, do inglês Traveling Salesman Problem) é um problema de otimização combinatória clássico pertencente à classe NP-Difícil que consiste na busca de um trajeto com custo mínimo atravessando todas as n cidades determinadas em uma instância do problema e por fim, retornando à primeira cidade que deu início ao tour. Seu modelo pode ser aplicado à problemas do mundo real como construção de agenda de tarefas, roteamento, aplicações de transporte e logística, dentre outros [10]. Pode ser formalizado como dado um conjunto $A = \{1,2, \dots, n\}$ de cidades, uma matriz de custos $C_{nxn} = [C_{i,j}]$ onde $C_{i,j}$ é o custo para ir da cidade i para a cidade j, consiste em minimizar a função:

$$C_T(n) = \sum_{i=1}^{n} C_{i,i+1} + C_{n,1}$$

Dado seu caráter de complexidade exponencial, algoritmos determinísticos são inviáveis para resolver o problema quando possui grande escala. Algoritmos de aproximação têm sido aplicados ao problema e alcançado resultados relevantes principalmente algoritmos bio-inspirados [10][11][1][5].

Algoritmos Evolutivos (AEs) são algoritmos de busca global apropriados para problemas cujo espaço de busca é vasto, descontínuo ou multimodal, tais como o PCV. Seu mecanismo de funcionamento simula o processo evolutivo natural das espécies em que uma população inicial compete pelos recursos do ambiente e se reproduz. Os mais aptos transmitem suas informações genéticas para gerações posteriores e os indivíduos são submetidos à processos de mutação que consiste em inserir diversidade na população [5]. Essa estratégia proporciona exploração paralela em vários pontos do espaço de busca em problemas de otimização combinatória. Cada indivíduo da população é uma proposta de solução para o problema, sendo avaliado (função objetivo) e classificado (seleção) segundo seu desempenho para resolver o problema.

Neste trabalho, foram realizados experimentos sobre o desempenho computacional e a qualidade das soluções obtidas com o operador de recombinação para algoritmos genéticos OX (Crossover de Ordem, do inglês Order Crossover) com diferentes taxas de mutação nos genes dos cromossomos. Para a realização dos experimentos, instâncias do PCV da popular biblioteca TSPLIB [12] foram utilizadas. Esta biblioteca foi criada em 1990 por Reinelt e possui mais de 100 exemplos com tamanhos que variam de 14 a 85.900 cidades.

Os resultados mostram que o operador de cruzamento OX possui desempenho muito bom para pequenas instâncias inclusive para um número pequeno de interações (gerações). Contudo, quando o número de cidades aumenta, o desempenho do algoritmo com o operador OX tende a cair, ficando mais evidente a necessidade de um número maior de gerações.

ALGORITMOS EVOLUTIVOS

A implementação de algoritmos computacionais para solucionar os mais variados problemas do cotidiano, ciência e engenharias tem proporcionado a otimização de processos reduzindo custos relacionados a tempo e uso de materiais ou esforço humano. De acordo com Garey & Johnson [3], os algoritmos são geralmente, procedimentos passo a passo para a solução de problemas. De forma mais concreta, podemos considerá-los simplesmente como programas de computador, escrito em alguma linguagem de programação precisa. É dito que um algoritmo resolve um problema P se o algoritmo pode ser aplicado a qualquer instancia I de P e é garantido que sempre produzirá uma solução para esta instancia I.

A otimização dos algoritmos já conhecidos para solucionar problemas clássicos constitui um ramo da pesquisa científica em computação, sempre buscando reduzir os tempos de processamento. Dentre os diversos problemas para os quais se busca solução algorítmica, há os conhecidos como Problemas de Otimização Combinatória (POC). Prestes [9] justifica a atenção dispensada por pesquisadores para esta categoria de problema atribuindo a este o fato da grande dificuldade de solução e a presença destes problemas em várias situações do cotidiano que demandam esforços para desenvolver algoritmos cada vez mais eficientes para serem aplicados. Ainda de acordo com Prestes [9], os tempos para solucionar instâncias de grande porte por algoritmos exatos são inviáveis, a opção nesses casos, seria o uso de algoritmos heurísticos.

Algoritmos Genéticos constituem uma estratégia utilizada para a resolução de problemas cuja complexidade é tão alta que não se conhece uma solução determinística [5]. Geralmente são problemas caracterizados pelo número alto de possíveis soluções, porém pela dificuldade em se determinar a melhor solução conhecidos como problemas intratáveis. Linden [5] define um problema intratável como um problema cujo tempo necessário para resolvê-lo é considerado inaceitável para os requerimentos do usuário da solução. Esses problemas têm motivado várias pesquisas e a implementação de diversas estratégias na tentativa de melhorar o tempo necessário para encontrar uma solução boa ou ótima e são usados para modelar vários problemas da ciência

e engenharia. Ainda de acordo com Linden [5] Algoritmos Genéticos (AG) é uma técnica de busca extremamente eficiente no seu objetivo de varrer o espaço de soluções e encontrar soluções próximas da solução ótima, quase sem necessitar interferência humana, sendo uma das várias técnicas da inteligência computacional dignas de estudo.

Inspirados na Teoria da Evolução de Darwin, os algoritmos genéticos consistem em propor uma população de soluções para um determinado problema e através de um ciclo de iterações (gerações), realizar a recombinação dos indivíduos (soluções candidatas) dessas populações submetendo os novos indivíduos gerados às avaliações de aptidão para resolver determinado problema, mantendo os indivíduos que convergem para uma região promissora do espaço de busca e substituindo as soluções menos aptas. Coley [2] cita como exemplos de aplicações de Algoritmos Genéticos os problemas de otimização combinatória em larga escala (tais como redes de tubulações de gás) e estimativas de parâmetros de valor real (como registros de imagem) dentro de espaços de busca complexos crivados de muitos ótimos locais. Outros problemas conhecidos passíveis da aplicação de algoritmos genéticos são o problema da mochila e o problema do caixeiro viajante (PCV).

O PROBLEMA DO CAIXEIRO VIAJANTE

Seja um conjunto de pontos representando n cidades, o problema do Caixeiro Viajante consiste na determinação de uma rota que inicia em uma cidade, passa por cada cidade do conjunto apenas uma vez, e retorna à cidade inicial da rota perfazendo uma distância total mínima. Esta rota é denominada ciclo Hamiltoniano de custo mínimo [7]. Para este problema o número de soluções possíveis é proporcional ao fatorial do número de cidades, o que torna a determinação de uma solução inviável em termos de tempo de execução para um conjunto grande de cidades, sendo assim apropriada a aplicação de estratégias de busca de solução ótima tais como algoritmos genéticos. Sendo um problema de otimização combinatória, o PCV serve de modelo para diversos problemas reais tais como: roteamento de veículos, programação de tarefas em máquinas, perfuração de placas de circuitos integrados, mapeamento do DNA humano, dentre outras aplicações.

O PCV pode ser classificado das seguintes formas: simétricos e assimétricos. São ditos simétricos os PCV's em que para todos os pares de cidades (i, j), o custo de caminhar da cidade i para a cidade j $(c_{i,j})$ é o mesmo do custo de percorrer no sentido j para i $(c_{j,i})$, e assimétrico para o caso de haver distinção entre estes custos. Neste projeto propõe-se a utilização do PCV simétrico através de 10 instâncias de tamanhos diferentes da biblioteca TSPLIB [12].

CROSSOVER DE ORDEM (OX)

Dada a representação por trajeto que consiste em uma permutação das n cidades em um vetor, a ordem em que as cidades aparecem no cromossomo é determinante para a avaliação do mesmo e exige a aplicação de operadores específicos para essa representação. Vários operadores de cruzamento baseados em ordem são conhecidos como Crossover de Ciclo (CX, do inglês Cycle Crossover) [8], Crossover de Mapeamento Parcial (PMX, do inglês Partially Matched Crossover) [4] e Crossover de Ordem (OX, do inglês Order Crossover) [6]. O operador Crossover de Ordem é um dos melhores operadores em termos de qualidade, velocidade e simplicidade de implementação. Seu funcionamento é descrito da seguinte forma:

1. Dados dois pais ($P1$ *e* $P2$), determina-se aleatoriamente dois pontos de corte
2. O subconjunto contido entre os dois pontos de corte dos pais, são copiados para os filhos ($O1$ *e* $O2$) exatamente nas mesmas posições de seus pais.
3. O cromossomo é completado a partir do segundo ponto de corte do filho, com os genes do outro pai que não estão no intervalo de corte do filho.

Para ilustrar o funcionamento do operador OX, considere os cromossomos P_1 e P_2 como os cromossomos pais, e os cromossomos O_1 e O_2 como os cromossomos geradora através da aplicação deste operador na Figura 1.

Figura 1: Aplicação do operador OX a P_1 e P_2

$$P_1 = (1,4 \mid 3,6,5 \mid 2) \quad \rightarrow \quad O_1 = (\,2,1 \mid 3,6,5 \mid 4)$$
$$P_2 = (6,3 \mid 2,4,1 \mid 5) \quad \rightarrow \quad O_2 = (\,6,5 \mid 2,4,1 \mid 3)$$

O Crossover de Ordem possui a vantagem de sempre gerar soluções factíveis. Seu esquema de determinação de substrings e preenchimento sequencial a partir do outro pai garante que as arestas dos filhos sejam predominantemente obtidas dos pais, proporcionando alta hereditariedade.

EXPERIMENTOS COMPUTACIONAIS

Para a realização de testes utilizando o operador Crossover de Ordem, um algoritmo evolutivo foi desenvolvido através da linguagem C e executado em um computador com processador Intel Core i5 com 8GB de memória RAM e Sistema Operacional Ubuntu. O algoritmo gera uma população inicial aleatoriamente com tamanho definido por parâmetro passado ao programa. Assim como o tamanho da população, outros parâmetros recebidos pelo programa são:

- Nome do arquivo contendo os dados da instância TSPLIB
- Número de gerações
- Taxa de Cruzamento
- Taxa de Mutação

Algumas instâncias da biblioteca TSPLIB indicam as distâncias entre as cidades explicitamente. Outras instâncias definem coordenadas das cidades e suas distâncias são calculadas através da distância Euclidiana 2D. O algoritmo foi desenvolvido de forma a trabalhar com essas duas opções. O método de seleção é a roleta viciada e a cada geração, o melhor indivíduo é automaticamente copiado para a população da próxima geração. Como operador de mutação utilizou-se a inversão de posições, em que 2 posições são aleatoriamente selecionadas na solução, e suas posições são invertidas.

Foram definidas três taxas de mutação diferentes: 5%, 10% e 15%. Para cada taxa de mutação o algoritmo foi executado 3 diferentes números de gerações: 500, 1000 e 2000. Foram selecionadas 10 instâncias simétricas da biblioteca TSPLIB com tamanhos que variam de 29 a 127 cidades. Para cada instância, o algoritmo foi executado 10 vezes para cada combinação taxa de mutação/geração, totalizando 90 execuções para cada instância e 900 execuções totais. O tamanho da população foi fixado em 500 para todas as execuções e a taxa de cruzamento foi definida como

80%. Para cada execução, anotou-se a melhor solução encontrada assim como o tempo de execução em que essa solução foi encontrada.

A qualidade da solução encontrada pelo algoritmo, é medida pelo percentual acima da solução ótima determinada para cada instância reportada no site do TSPLIB [12]. Esse cálculo é realizado através da seguinte fórmula:

$$\%\, erro = \frac{solução\ encontrada - solução\ ótima}{solução\ ótima} * 100$$

As Tabelas 1 e 2 mostram as médias das soluções encontradas pelo algoritmo para cada instância (5 em cada Tabela) testada do TSPLIB com suas variações de taxa de mutação e número de gerações. As Tabelas 3 e 4 mostram o erro percentual de cada solução indicada nas Tabela 1 e 2.

Tabela 1 - Média de soluções encontradas pelo algoritmo com operador OX (Parte 1)

		bayg29	eil51	berlin52	brazil58	eil76
	Tamanho	**29**	**51**	**52**	**58**	**76**
	Ótimo	**1610**	**426**	**7542**	**25395**	**538**
500	**5%**	1687	465	9312	32019	684
	10%	1689	468	9260	28349	727
	15%	1620	476	8524	28831	708
1000	**5%**	1615	462	8294	29737	683
	10%	1713	484	8526	29506	734
	15%	1621	468	8451	30712	657
2000	**5%**	1684	488	9038	27617	657
	10%	1624	477	8233	27282	630
	15%	1620	469	8449	29715	658

Tabela 2 - Média de soluções encontradas pelo algoritmo com operador OX (Parte 2)

		pr76	kroA100	eil101	lin105	bier127
	Tamanho	**76**	**100**	**101**	**105**	**127**
	Ótimo	**108159**	**21282**	**629**	**14379**	**118282**
500	**5%**	174947	40713	1082	31823	192186
	10%	160352	34135	954	27381	219598
	15%	140800	33707	973	28826	196454
1000	**5%**	171050	39805	929	24891	181489
	10%	146895	35176	788	26010	172027
	15%	141894	32285	953	26228	166922
2000	**5%**	152156	34996	852	23815	156650
	10%	130923	27277	793	22560	168242
	15%	135507	28371	844	19982	166898

Tabela 3 - Média do % de erro de cada solução encontrada indicada na Tabela 1

		bayg29	eil51	berlin52	brazil58	eil76
	Tamanho	**29**	**51**	**52**	**58**	**76**
	Ótimo	**1610**	**426**	**7542**	**25395**	**538**
500	**5%**	4,8 %	9,2 %	23,5 %	26,1 %	27,1 %
	10%	4,9 %	9,9 %	22,8 %	11,6 %	35,1 %
	15%	0,6 %	11,7 %	13,0 %	13,5 %	31,6 %
1000	**5%**	0,3 %	8,5 %	10,0 %	17,1 %	27,0 %
	10%	6,4 %	13,6 %	13,0 %	16,2 %	36,4 %
	15%	0,7 %	9,9 %	12,1 %	20,9 %	22,1 %
2000	**5%**	4,6 %	14,6 %	19,8 %	8,7 %	22,1 %
	10%	0,9 %	12,0 %	9,2 %	7,4 %	17,1 %
	15%	0,6 %	10,1 %	12,0 %	17,0 %	22,3 %

Tabela 4 - Média do % de erro de cada solução encontrada indicada na Tabela 2

		pr76	kroA100	eil101	lin105	bier127
	Tamanho	**76**	**100**	**101**	**105**	**127**
	Ótimo	**108159**	**21282**	**629**	**14379**	**118282**
500	**5%**	61,7 %	91,3 %	72,0 %	121,3 %	62,5 %
	10%	48,3 %	60,4 %	51,7 %	90,4 %	85,7 %
	15%	30,2 %	58,4 %	54,7 %	100,5 %	66,1 %
1000	**5%**	58,1 %	87,0 %	47,7 %	73,1 %	53,4 %
	10%	35,8 %	65,3 %	25,3 %	80,9 %	45,4 %
	15%	31,2 %	51,7 %	51,5 %	82,4 %	41,1 %
2000	**5%**	40,7 %	64,4 %	35,5 %	65,6 %	32,4 %
	10%	21,0 %	28,2 %	26,1 %	56,9 %	42,2 %
	15%	25,3 %	33,3 %	34,2 %	39,0 %	41,1 %

RESULTADOS

Em uma análise das Tabelas 3 e 4 que mostram mais claramente a qualidade dos resultados atingidos através dos percentuais de erro em relação às soluções ótimas determinadas, é possível perceber que o algoritmo evolutivo desenvolvido neste trabalho com o operador de cruzamento OX, possui muito bom desempenho para instâncias cujas dimensões são menores. Este é o caso da instância bayg29 cujo tamanho é 29 e o percentual de erro ficou em geral abaixo de 5%. Em menor escala as instâncias eil51, berlin52 e brazil58 também tiveram bons índices girando em torno de 15% a 20% em geral. É possível perceber também que os resultados foram melhores quando o número de gerações aumentou. Como se trata de problemas cujos espaços de busca são grandes e complexos, pode se deduzir que um número maior de gerações favorece a busca por melhores soluções. Para todas as instâncias é possível perceber uma grande diferença nos resultados obtidos quando o algoritmo foi executado com 500 gerações e 2000 gerações.

As taxas de mutação variadas não significaram melhores resultados para instâncias pequenas, tendo mais impacto quando as instâncias possuíam tamanho maior (a partir da instância pr76, ver Tabela 4) com a taxa mais agressiva (15%). Com instâncias maiores, a taxa de mutação mais alta possivelmente possibilitou uma exploração melhor dos ótimos locais, dando à busca uma característica mais intensificadora.

CONCLUSÃO

Neste trabalho, foram realizados experimentos com um algoritmo evolutivo implementado utilizando o operador de cruzamento OX (Crossover de Ordem). Os experimentos foram realizados submetendo-se instâncias de problemas do caixeiro viajante (PCV) da biblioteca TSPLIB ao algoritmo desenvolvido. Os resultados mostram que o algoritmo evolutivo com OX possui desempenho relativamente satisfatório para o PCV principalmente para instâncias de problemas menores e quando o número de gerações aumenta.

Pode-se deduzir que outros parâmetros do algoritmo evolutivo tais como tamanho da população, operador de mutação, taxa de mutação, número de gerações, operador de seleção e taxa de cruzamento podem influenciar o desempenho do operador de cruzamento OX, o que demanda uma necessidade de mais pesquisas sobre a relação existente entre esse operador e outros operadores presentes na literatura possíveis para o algoritmo.

Os algoritmos evolutivos como técnica ou como paradigma de desenvolvimento, tem despertado a atenção de pesquisadores em diversas áreas que possuam problemas complexos em que, uma solução aproximada da melhor pode ser satisfatória considerando que a busca pela melhor solução inviabilizaria a solução do problema em tempo factível. Isso foi possível constatar pelo número de trabalhos pesquisados que estão investigando melhor as propriedades dessa classe de algoritmos tais como representações, operadores, estratégias de busca e tendência. Este trabalho contribuiu para este entendimento uma vez que foi possível verificar na prática a eficiência dos algoritmos evolutivos e despertou o interesse e gerou novos questionamentos que poderão se tornar novas pesquisas. Esses questionamentos estão relatados na forma de sugestão de trabalhos futuros na próxima seção.

TRABALHOS FUTUROS

Para trabalhos futuros, sugere-se que outras análises sejam feitas com outros operadores de cruzamento presentes na literatura tais como (CX, PMX e PBX). Pode-se considerar também estudos relacionados às versões modificadas do operador OX, que é o caso dos operadores OX1 e OX2. Além de medir o desempenho desses operadores outra possibilidade de pesquisa é verificar melhores combinações de operadores de cruzamento com outros operadores tais como os de seleção e de mutação. Sabe-se que esses operadores são complementares como estratégias de busca e possivelmente pode-se determinar quais operadores de seleção e mutação favorecem a busca ao serem combinados com o operador de cruzamento OX por exemplo.

Outra sugestão é pesquisar os efeitos causados pela inserção de heurísticas específicas em operadores de cruzamento, mutação e inicialização. Heurísticas que favoreçam a utilização de arestas de menor custo por exemplo podem facilitar a convergência do algoritmo. Outras heurísticas menos aparentes para o PCV devem ser pesquisadas.

AGRADECIMENTOS

Este trabalho foi desenvolvido através do Programa Institucional de Bolsas de Iniciação Científica (PIBIC) do Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG-Câmpus Jataí), instituição à qual agradecemos por todo suporte para a realização deste.

REFERÊNCIAS

[1] – AHMED, Zakir H. Genetic Algorithm for Travelling Salesman Problem using Sequential Constructive Crossover Operator, **International Journal of Biometrics & Bioinformatics**, Vol. 3, Issue 6, p. 96-105, mar, 2010.

[2] - COLEY, David A. **An Introduction to Genetic Algorithms for Scientists and Engineers**. World Scientific Pub Co, Singapure, 1999.

[3] - GAREY, M. R. & JOHNSON, D. S**. Computers and Intractability: a guide to the theory of NP-Completeness**. San Francisco, Freeman, 1979.

[4] – GOLDBERG, D. E.; LINGLE, R. Alleles, loci, and the traveling salesman problem, **Proceedings of an International Conference on Genetic Algorithms and Their Applications**, p. 154–159. 1985.

[5] LINDEN, Ricardo. **Algoritmos Genéticos**. 3ª ed. Rio de Janeiro: Ciência Moderna, 2012.

[6] - MICHALEWICZ, Zbigniew. Genetic Algorithms + Data Structures = Evolution Programs. 3ª ed. London, UK. Springer-Verlag. 1996.

[7] - MURTY, K.G., **Linear and combinatorial programming**. Robert E. Krieger: Florida, 1985.

[8] – OLIVER, I.M.; SMITH, D.J.; HOLLAND, J.R.C. A study of permutation crossover operators on the traveling salesman problem. **Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application**, p. 224–230. 1987

[9] - PRESTES, A. Nunes. **Uma Análise Experimental de Abordagens Heurísticas Aplicadas ao Problema do Caixeiro Viajante**. Universidade Federal do Rio Grande do Norte. Natal. 2006.

[10] - RAY, Shubhra Sankar; BANDYOPADHYAY, Sanghamitra; PAL, Sankar K. Genetic Operators for combinatorial optimization in TSP and microarray gene ordering. **Journal Applied Intelligence**, Vol. 26, Issue 3, p. 183-195, jun, 2007.

[11] – RANI, Kanchan; KUMAR, Vikas. Solving Travelling Salesman Problem using Genetic Algorithm Based on Heuristic Crossover and Mutation Operator. **International Journal of Research in Engineering & Technology**, Vol. 2, Issue 2, p. 27-34, fev, 2014.

[12] – TSPLIB. Disponível em: http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/ Acesso em: 15 fev 2016